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*

2023 IEEE International Conference on Data Mining (ICDM) | 979-8-3503-0788-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDM58522.2023.00188

***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 privacy and security in data management. Nevertheless, existing machine unlearning methods fail to perform exactly and efficiently in a 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 the 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

Machine learning models require large amounts of data to perform well, often collected from users or third-party sources. In many real-world applications, users willingly share their data with service providers and platforms in exchange for better services. However, there is a growing emphasis on protecting user privacy, as evidenced by laws such as the General Data Protection Regulation (GDPR) [2] and California Consumer Privacy Act (CCPA) [3]. For example, a user may request to delete part of their search history, or a hospital may be asked to remove certain patient records. These scenarios raise an important and practical question: what should service providers do when users request that their data be deleted from their services or platforms?

One direct approach to address users' data removal requests deleting their data from the databases. However, the memorization capabilities of machine learning models [4] pose a challenge, as the information from the training data becomes embedded within the model parameters and is not easily forgotten. Additionally, the naive deletion of data can introduce security vulnerabilities, potentially enabling malicious actors to exploit various techniques, such as model inversion attacks [5], membership inference attacks [6], reconstruction attacks [7], and more, to infer users' private information.

The initial idea of this paper is discussed in the author's dissertation [1], and some preliminary results are released in Section 5 of the dissertation. In this conference version, we add algorithms, theorems, and experiments to enrich the content.

The process of correctly removing data in the context of machine learning, referred to as "machine unlearning" [8], necessitates the elimination of data from training datasets and its impact on the learned models. While it may seem intuitive to retrain machine learning models from scratch using the remaining databases, excluding the deleted data, this approach can incur a substantial computational cost, particularly for models with millions of parameters [9]. As a result, the primary focus of current machine unlearning research centers on designing methods that are computationally efficient and time-saving. To date, there have been only a limited number of studies on machine unlearning, each with its own set of limitations. These limitations range from simplistic learning methods, such as linear regression [10], [11], to model-dependent techniques, such as decision trees [12] and k-means clustering [13]. Furthermore, within the domain of federated learning (FL), existing research primarily concentrates on enhancing unlearning efficiency through approximate methods. However, these efforts often neglect the evaluation of model utility, such as model accuracy, following the unlearning process. This oversight can detrimentally impact the performance of unlearned models.

Motivated by the limitations inherent in existing unlearning methods, this paper endeavors to develop a model-agnostic, exact, and efficient federated machine unlearning approach. To accomplish exact federated unlearning, we introduce the concept of α-quantization [14] to enhance the stability of the federated model. Building upon the foundation of the quantized federated model, we present the Exact-Fun algorithm, which achieves unlearning in effectiveness and efficiency.

We underscore the key contributions here:

1. This paper studies the exact federated unlearning, which can extend to various machine learning models, rendering our approach model-agnostic.

2. We introduce the Q-FL algorithm, designed to facilitate exact federated unlearning while ensuring model convergence.

3. We introduce the Exact-Fun algorithm, a solution designed to efficiently handle users' data deletion requests, with empirically validated unlearning efficiency.

4. Both the Q-FL and Exact-Fun algorithms undergo extensive experiments. The results affirm the efficacy and efficiency, surpassing the current state-of-the-art approaches.

979-8-3503-0788-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDM58522.2023.00188

II. Related Works

Existing research on unlearning can be classified into two main categories: exact unlearning and approximate unlearning, based on their respective efficiency and effectiveness.

Exact unlearning necessitates that the distribution of unlearned model parameters must exactly match the distribution of model parameters obtained through retraining on the dataset excluding the deleted data. Cao et al. [8] pioneered the development of unlearning algorithms for statistical query-based learning models, such as the Naive Bayesian classifier and Support Vector Machines (SVM). Ginart et al. [13] introduced the first unlearning method for the unsupervised learning k-means clustering algorithm. Their approach leverages stability and divide-and-conquer techniques to enhance unlearning efficiency. In Bourtoule et al.'s work [15], they designed a Sharding, Isolation, Slicing, and Aggregation (SISA) framework to reduce unlearning time. The concept involves splitting a dataset into smaller parts, thereby reducing the retraining time. Similarly, Aldaghri et al. [16] employed ensemble learning to partition the training dataset into disjoint shards using a coding matrix. Following this trend, Schelter et al. [17] and Brophy et al. [12] devised unlearning algorithms for the random forests algorithm. They applied a similar concept to adjust the structure of decision trees to minimize the retraining effort for subtrees.

Approximate unlearning offers a more time-efficient approach compared to exact unlearning. In the current body of research, there are two primary approaches to processing a model: gradient-based methods [18]–[22] and Hessian-based methods [10], [11], [23]. For instance, Wu et al. [18], Graves et al. [19], and Wu et al. [24] employed a similar approach, updating the trained model with stored gradients when specific data points are removed. Neel et al. [21] and Ullah et al. [22] introduced statistical indistinguishability and algorithm stability, respectively, into gradient descent methods for provable approximation in data unlearning. To address adversarial scenarios where users intentionally delete data with specific distributions, Gupta et al. [20] proposed adaptive machine unlearning capable of handling arbitrary model classes and training methodologies. On another front, Guo et al. [10] devised differentially private data removal mechanisms that unlearn data from learned models using the Hessian matrix. Golatkar and Wang [23], [25] concentrated on unlearning specific class labels from deep networks. To mitigate the computational cost associated with the Hessian matrix, Izzo et al. [11] introduced a sublinear algorithm to expedite unlearning from linear models efficiently. In a separate branch of research, probability-based unlearning methods were utilized to tackle approximate unlearning in federated settings, employing Bayesian [26], [27] and Monte Carlo [28] techniques.

III. Exact Federated Unlearning

A. Problem Formulation

Once an FL model has been trained on a given training dataset, its model parameters remain fixed and can be deployed for various applications. When a client, denoted as j and belonging to the set $\mathcal{K}$, wishes to remove their data $\mathcal{U}_j \subset \mathcal{D}_j$ (with $|\mathcal{U}_j| = m < |\mathcal{D}_j|$) from the trained federated model, they can initiate an unlearning request to the server. This request entails removing the data in $\mathcal{U}_j$ from client j's local training dataset, denoted as $\mathcal{D}_j$. Situations where $\mathcal{U}_j = \mathcal{D}_j$ are treated differently, indicating the departure of client j from the federated learning system, a scenario not addressed in our problem statement. Additionally, alongside the federated model, the federated learning algorithm $\mathscr{A}$ generates a set of meta-data $\mathcal{M}$. In this context, an unlearning algorithm can be formally defined as $\mathscr{A}^u : (\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \to \mathcal{W}$, where it takes as inputs the trained model $\mathscr{A}(\mathcal{D})$, the unlearning dataset $\mathcal{U}_j$, and the meta-data $\mathcal{M}$ to produce an updated unlearned model.

To fulfill an unlearning request, the process involves the removal of $\mathcal{U}_j$ from both $\mathcal{D}_j$ and $\mathcal{D}$. And the influence of $\mathcal{U}_j$ on the trained federated model should be effectively revoked. Moreover, a successful exact unlearning algorithm should ensure two key criteria: (i) The unlearning cost is lower than the cost of retraining the model from scratch using the remaining dataset $\mathcal{D}^u = \mathcal{D} \setminus \mathcal{U}_j$. (ii) The distribution of unlearned model parameters matches that 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 the unlearned model and the retrained model are equal.

B. Quantization of Federated Learning

Within the FL system, when the dataset $\mathcal{U}_j$ is removed from both $\mathcal{D}_j$ and $\mathcal{D}$ per a client j's unlearning request, it can lead to changes in the final trained federated model. Consequently, it becomes challenging to guarantee the exact equivalence in distribution between the unlearned model and the model trained from scratch on $\mathcal{D}^u$, as stipulated in Definition 1. To address this challenge, it becomes imperative to enhance the stability of the FL algorithm, thereby facilitating exact unlearning. This means that minor changes in the local dataset should ideally result in only minimal or no changes in the distribution of trained federated model parameters. In the context of our problem, when a dataset $\mathcal{U}_j$ is earmarked for unlearning, the goal is to ensure that the trained federated model undergoes minimal alterations. If these changes can be efficiently assessed during the unlearning process, it becomes feasible to achieve exact unlearning with efficiency.

The way to reach stability in federated learning is quantization [14], where the aggregated parameters of the federated model are quantized to a discrete vertex in the hypothesis space of model parameters. The quantization operation

Algorithm 1 Quantized Federated Learning (Q-FL)

Input: the number of iterations T, the number of clients K, learning rate η, the granularity of quantization α
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.

$q(\alpha, w^t) = \hat{w}^t$ can map its continues input value w^t to a discrete value $\hat{w}^t$, which is expressed as follows:

$$\hat{w}^t = \alpha \cdot z^*, \text{ s.t. } z^* = \arg\min_{z\in\mathbf{Z}^d} \|w^t - \alpha \cdot z\|_2, \quad (1)$$

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 $\hat{w}^t = [1.0, 2.5]$.

We introduce the Q-FL algorithm, outlined in Algorithm 1. In the initial phase of Q-FL, the server initializes the model parameter w^0. This initialization is then passed through the quantization function $q(\alpha, \cdot)$ to yield the quantized model $\hat{w}^0$. Subsequently, this quantized model is distributed to all participating clients as their local models for the computation of ClientUpdate$(\cdot)$. The client operations mirror those of the original FL, encompassing tasks such as gradient computation, local model updates, and the transmission of their updated local models to the server. Upon receiving these local updates and executing the aggregation process, the server obtains a new federated model denoted as w^{t+1}. Following this, the quantization function is employed in Line 7, operating on the federated model w^{t+1} to produce 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 retained on the server as part of the meta-data $\mathcal{M}$.

It is noteworthy that, by applying quantization at the server in each iteration t, the quantized federated model tends to exhibit stability as a constant with a high probability (as substantiated in Theorem 2). Furthermore, the Q-FL algorithm we propose not only facilitates exact unlearning but also preserves the model's utility and convergence, even when the quantization operation introduces perturbations to its parameters. To delve deeper into our analysis, we will first present Lemma 1 and subsequently employ it to establish the convergence bounds for the proposed Quantized Federated Learning (Q-FL) algorithm.

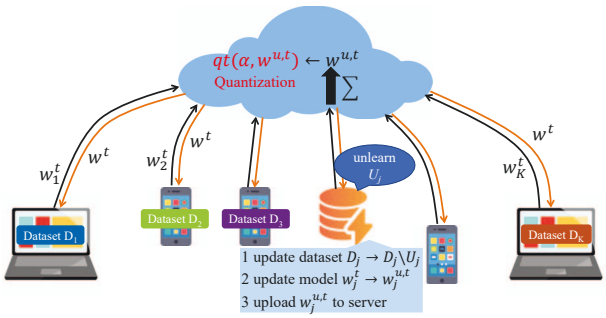


Fig. 1: The framework of proposed Exact-Fun algorithm

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:*

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

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

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.* (3) *when $\eta \in (0, \frac{2}{\mu}]$ and is given by Eq.* (4) *when $\eta \in (\frac{2}{\mu}, \infty)$.*

$$\mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \leq (1 + 2\tau\beta_1)^t C^0 - \frac{\beta_2 \alpha^2 d[1 - (1 + 2\tau\beta_1)^t]}{24\tau\beta_1}, \quad (3)$$

$$\mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \leq (\frac{1}{2\tau} + \beta_1)G^2 + \frac{\beta_2 \alpha^2 d}{12}, \quad (4)$$

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. (3) is reducing along with iteration t. Theorem 1 states that even though our proposed Q-FL algorithm is obfuscated by quantization, the trained federated model can still converge. All proof details can be found in our complete version in this link.

C. Exact and Efficient Federated Unlearning

Upon the completion of our Quantized Federated Learning algorithm after T iterations, a fully trained federated model $\hat{w}^T$ is established. With this quantized federated model $\hat{w}^T$ in place, it becomes feasible to initiate the exact unlearning process to remove $\mathcal{U}_j$ from the trained model $\hat{w}^T$ when requested by client j. This process results in the corresponding unlearned federated model denoted as $\hat{w}^{u,T}$.

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

$$w_j^{t+1} = w_j^t - \eta \frac{1}{|\mathcal{D}_j|}[\sum_{(x,y)\in\mathcal{D}_j^u} \nabla l(w_j^t, (x, y)) + \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j^t, (x, y))]. \quad (5)$$

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. (6) to unlearn $\mathcal{U}_j$.

$$w_j^{u,t+1} = w_j^t - \eta \frac{1}{|\mathcal{D}_j^u|}[\sum_{(x,y)\in\mathcal{D}_j^u} \nabla l(w_j^t, (x, y))], \quad (6)$$

Algorithm 2 Exact and Efficient Federated Unlearning

Input: the number of iterations T, the number of clients K, the granularity of quantization α, 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. (7)
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** $\hat{w}^{u,T}$

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

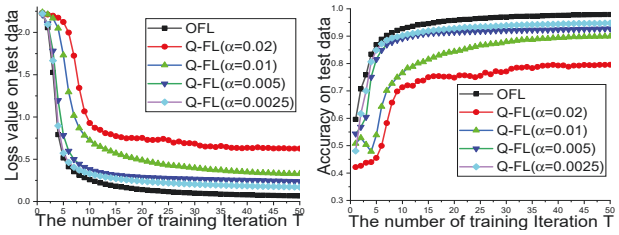
The distinction between the trained local model w_j^{t+1} and the unlearned local model $w_j^{u,t+1}$ primarily lies in the gradients associated with the data in $\mathcal{U}_j$. So, in order to efficiently obtain $w_j^{u,t+1}$ without the need to compute gradients for $\mathcal{D}_j^u$, we can simply subtract the gradient of $\mathcal{U}_j$ from the previously trained local model w_j^{t+1}. By comparing Eq. (5) and Eq. (6), the rule of updating $w_j^{u,t+1}$ from w_j^{t+1} is given as

$$w_j^{u,t+1} = \frac{|\mathcal{D}_j|}{|\mathcal{D}_j^u|} w_j^{t+1} - \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} w_j^t + \frac{\eta}{|\mathcal{D}_j^u|} \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j^t, (x, y))]. \tag{7}$$

In Eq. (7), we already have access to w_j^{t+1} and w_j^t from Q-FL Algorithm 1 and the only computation required pertains to gradients for the data points within $\mathcal{U}_j$. Additionally, thanks to the quantized stability inherent in Q-FL (as demonstrated in Theorem 2), the federated model is likely to maintain stability with a high probability even after the unlearning process.

The process of exact federated unlearning for the deletion of $\mathcal{U}_j$ is illustrated in Fig. 1 and outlined in Algorithm 2. When $\mathcal{U}_j$ is removed from $\mathcal{D}_j$, an updated local model $w_j^{u,t}$ is computed at iteration t, as detailed in Eq. (7). Then, $w_j^{u,t}$ is uploaded to the server to facilitate the aggregation of a new federated model $w^{u,t}$. This newly derived model is then quantized using the quantization function $q(\alpha, \cdot)$ to generate $\hat{w}^{u,t}$. If $\hat{w}^{u,t}$ matches the stored federated model $\hat{w}^t$, the deletion of $\mathcal{U}_j$ has no impact on the previously trained federated model $\hat{w}^t$. This signifies that our unlearning approach is exact. Conversely, if $\hat{w}^{u,t}$ differs from $\hat{w}^t$, it indicates a disruption in the stability of the quantized federated model. In such cases, retraining from the current t-th iteration to T is required to eliminate the influence of $\mathcal{U}_j$ from the learned models.

Within our unlearning algorithm, Exact-Fun, the primary computational burden resides in the retraining process, specifically, as indicated in Line 12 of Algorithm 2. This computational aspect can be managed by adjusting the quantization parameter α based on system requirements. A larger value for α enhances stability, reduces the probability of retraining, and lowers retraining costs. However, it also leads to a decrease in model utility due to the increased noise perturbations. Hence, in Theorem 2, we establish that the retraining probability in Algorithm 2 is a function of α, and it becomes evident that selecting an appropriate value for α plays a crucial role in guiding efficient unlearning in practical applications.



(a) loss on F-MNIST dataset (b) accu on F-MNIST dataset

Fig. 2: The loss and accuracy of FL models with different α (K=50).

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., $\|w^t - w^{u,t}\| \leq B$ with $t \in [0, T]$. The probability that Algorithm 2 needs retraining is given by Eq.* (8)*.*

$$\Pr(\hat{w}^{u,t} \neq \hat{w}^t) = \begin{cases} 1 - (\frac{\alpha}{2B})^d, & B \in [\alpha, \infty) \\ 1 - (1 - \frac{B}{2\alpha})^d, & B \in (0, \alpha) \end{cases} \tag{8}$$

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

Theorem 2 shows that a larger quantization parameter α has the potential to decrease the retraining probability, albeit at the cost of potentially inferior convergence bounds. Therefore, the balance between efficiency and convergence must be thoughtfully considered and designed to meet specific requirements.

IV. Experiments

In this section, we evaluate the performance of the Q-FL algorithm and the Exact-Fun algorithm. Experiment settings can be found in our complete version in the link.

A. Q-FL Performance

To empirically assess the impact of α on our Q-FL algorithm, we have configured α accordingly for both the Fashion-MNIST and CIFAR-10 datasets. The loss values of federated models on their respective test datasets across various iterations are presented in Fig. 2. First and foremost, it is evident that the loss values for all compared federated models exhibit a decreasing trend as T increases, eventually stabilizing after a certain number of iterations (e.g., at T=35 in Fig. 2(a)). This observation lends support to the convergence capabilities of our quantized federated learning (Q-FL), aligning with our analysis in Theorem 1. Similar conclusions can be drawn for the CIFAR-10 dataset in its entirety. Next, we examine the impact of α on the testing accuracy of federated models in Fig. 2(b). It is evident from the figure that, in the case of our Q-FL model, employing a smaller α value results in higher

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 I: Training time comparison between OFL and Q-FL

accuracy and greater stability in training accuracy. This is because a smaller α corresponds to less noise being introduced into the model parameters. Furthermore, we have conducted a comparison of the training times between our Q-FL and the baseline OFL. Table I presents the average training time required for a single iteration, measured in seconds, for both our Q-FL and the baseline OFL. In summary, the quantization function $q(\alpha, \cdot)$ employed in our Q-FL algorithm does not significantly contribute to the overall time consumption and remains conducive to achieving the desired model accuracy.

B. Unlearning Effectiveness and Efficiency

In this part, we evaluate the effectiveness and efficiency of our Exact-Fun algorithm.

Unlearning Effectiveness (Accuracy). To assess the effectiveness of unlearning, we compare the accuracy differences between various unlearning algorithms, such as Exact-Fun and the INFOCOM22 algorithm. We employ the Symmetric Absolute Percentage Error (SAPE) metric, a commonly used measure in unlearning studies [18], [24], [29].

Figure 3(a) illustrates the impact of the quantization parameter α on the effectiveness of Exact-Fun, with $p = 0.1$. It's worth noting that the baseline (INFOCOM22) remains unaffected by changes in α, maintaining a constant SAPE as α varies. Evidently, the SAPE value on the test data increases as α becomes larger. This occurs because a larger α introduces more noise perturbation during the quantized federated learning process, resulting in lower accuracy in our unlearned federated model. In comparison to the INFOCOM22 baseline, our Exact-Fun algorithm outperforms it when α is small (e.g., α values of 0.0025, 0.005, and 0.01). Conversely, the SAPE on the unlearned data increases gradually for smaller α values but rises sharply when α reaches 0.02. With smaller α values, our Exact-Fun algorithm is more inclined to retrain the quantized federated model on the remaining dataset, akin to the retraining method, resulting in a smaller accuracy difference between our unlearned model and the retrained model. In contrast, larger α values reduce the probability of retraining in our Exact-Fun, leading to a larger accuracy difference between our unlearned model and the retrained model on the unlearned data. Subsequently, we explore the impact of the unlearning portion p on unlearning effectiveness, considering values of p such as 0.05, 0.1, 0.15, and 0.2. Figures 3(c) and 3(d) reveal that as the unlearning portion p increases, the SAPE value of our Exact-Fun decreases. This trend can be explained through the lens of model retraining. When p is smaller, the probability of retraining the quantized federated model in Exact-Fun is lower, resulting in a larger difference between our quantized model and the retrained model. Conversely, as p increases, our Exact-Fun algorithm requires retraining on the remaining dataset, consequently reducing the accuracy difference. 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. In summary, our Exact-Fun outperforms the baseline approximate unlearning algorithm in effectiveness, especially when unlearning more data.

Unlearning Effectiveness (Privacy). Membership inference attack (MIA) is a metric to evaluate the unlearning effectiveness in many related works [30]–[32], 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 II, original model means we only delete data but do not change the trained model, so high MIA accuracy remains on both datasets. From Table II, 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. The higher the speed-up ratio, the better efficiency.

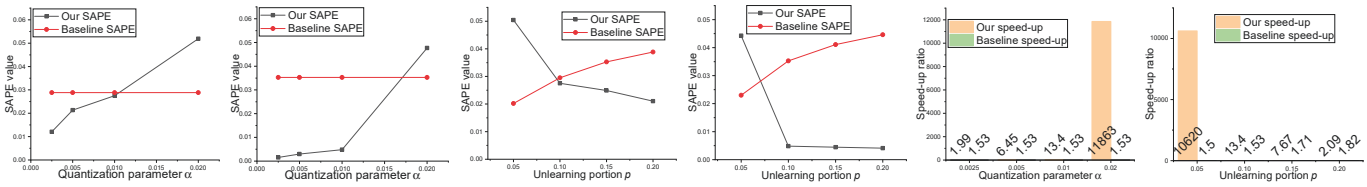
The impact of the quantization parameter α on the efficiency of Exact-Fun is depicted in Figure 3(e). Clearly, the speed-up ratio increases with higher values of α because a greater α signifies stronger stability in our quantized federated model and a reduced likelihood of retraining, ultimately resulting in a higher speed-up ratio. In comparison to the baseline (with a fixed speed-up ratio of 1.53), our Exact-Fun proves to be more efficient for every α value. Next, Figure 3(f) illustrates the influence of the unlearning portion p on unlearning efficiency. As p increases, the speed-up ratio of Exact-Fun diminishes. This decrease occurs because unlearning a larger portion of the data may disrupt the stability of the quantized federated model, necessitating more retraining time. In contrast, the baseline experiences an increase in its speed-up ratio as p grows larger. Nevertheless, even with this trend, our Exact-Fun consistently outperforms it and can achieve a speed-up ratio exceeding 10,000 times when p equals 0.05.

V. Conclusion & Future Work

In this research, we delve into the novel challenge of federated unlearning. As a novel solution for exact federated unlearning, we introduce the Q-FL algorithm. Subsequently, we present the Exact-Fun algorithm to facilitate the unlearning process. Besides, we provide an analysis of the upper bound for convergence in the Q-FL algorithm and offer insights into the analytical retraining probability associated with the Exact-Fun algorithm. Our results unequivocally demonstrate that our Exact-Fun algorithm surpasses the baseline method in both efficacy and efficiency.

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 II: 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. 3: SAPE on Fashion-MNIST test data and unlearned data with different α and p in Fig. 3(a), 3(b), 3(c), 3(d); speed-up in 3(e), 3(f).

REFERENCES

[1] Z. Xiong, "Towards privacy preservation of federated learning in artificial intelligence of things," Ph.D. dissertation, Georgia State University, 2023, doi:https://doi.org/10.57709/35862659.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[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] 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.

[11] 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*, vol. 130. PMLR, 13–15 Apr 2021, pp. 2008–2016.

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

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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

[31] 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.

[32] 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.