Unlearning through Knowledge Overwriting: Reversible Federated Unlearning via Selective Sparse Adapter

Zhengyi Zhong[1], Weidong Bao[1], Ji Wang[1]*, Shuai Zhang[1], Jingxuan Zhou[1],
Lingjuan Lyu[2], Wei Yang Bryan Lim[3]
[1]Laboratory for Big Data and Decision, National University of Defense Technology, China.
[2] Sony AI, Japan.[3] Nanyang Technological University, Singapore.
{zhongzhengyi20, wdbao, wangji, zhangshuai20, zhoujingxuan}@nudt.edu.cn,
lingjuanlvsmile@gmail.com, bryan.limwy@ntu.edu.sg

https://github.com/Zhong-Zhengyi/FUSED-Code

Abstract

Federated Learning is a promising paradigm for privacy-preserving collaborative model training. In practice, it is essential not only to continuously train the model to acquire new knowledge but also to guarantee old knowledge the right to be forgotten (i.e., federated unlearning), especially for privacy-sensitive information or harmful knowledge. However, current federated unlearning methods face several challenges, including indiscriminate unlearning of cross-client knowledge, irreversibility of unlearning, and significant unlearning costs. To this end, we propose a method named `FUSED`*, which first identifies critical layers by analyzing each layer's sensitivity to knowledge and constructs sparse unlearning adapters for sensitive ones. Then, the adapters are trained without altering the original parameters, overwriting the unlearning knowledge with the remaining knowledge. This knowledge overwriting process enables* `FUSED` *to mitigate the effects of indiscriminate unlearning. Moreover, the introduction of independent adapters makes unlearning reversible and significantly reduces the unlearning costs. Finally, extensive experiments on three datasets across various unlearning scenarios demonstrate that* `FUSED`*'s effectiveness is comparable to Retraining, surpassing all other baselines while greatly reducing unlearning costs.*

1. Introduction

Background. Federated Learning (FL) [25, 30, 46] has emerged as a promising paradigm for privacy-preserving collaborative model training. In practice, FL models need to acquire new knowledge continuously while also ensuring the *"right to be forgotten"* for previously used training data [12, 37]. For example, a year after the launch of ChatGPT, The New York Times accused OpenAI and Microsoft of the unauthorized use of its media data for training, demanding that they delete the acquired knowledge from its models [10]. Furthermore, malicious clients may inject harmful data during training, potentially poisoning the global model. As a result, it is crucial for the global model to eliminate such harmful knowledge. This leads to the concept of *Federated Unlearning* (FU).

Challenges. In the field of FU, two primary categories of methods have emerged: retraining-based methods [29] and model manipulation-based methods [38]. Among these, retraining-based methods are widely regarded as the state-of-the-art (SoTA) for achieving model unlearning. This approach involves removing the data designated for unlearning and retraining the model from scratch until convergence. Conversely, model manipulation methods modify the model directly using techniques such as gradient ascent, knowledge distillation, and setting parameters. However, existing methods still face several challenges:

- *Indiscriminate unlearning*: In scenarios where knowledge overlaps occur among clients, traditional methods indiscriminately remove shared knowledge during the unlearning process, leading to a substantial decline in the performance of other clients.
- *Irreversible unlearning*: In FL systems, clients' unlearning requests may change dynamically. When a client no longer needs to forget certain knowledge, traditional methods cannot recover that memory quickly.
- *Significant unlearning costs*: The retraining-based method requires multiple iterations, resulting in significant computational and communication costs. Even simple adjustments to model parameters can demand a significant amount of storage as a compensatory cost.

*Corresponding Author: Ji Wang

Method. To address these challenges, we propose a reversible **F**ederated **U**nlearning method via **SE**lective sparse a**D**apter (FUSED). To begin, we perform a layer-wise analysis of the model's sensitivity to knowledge changes, identifying the most affected layers. These sensitive layers are then processed into sparse structures known as unlearning adapters. This process, termed Critical Layer Identification (CLI), significantly reduces the number of model parameters, thereby lowering unlearning costs. Subsequently, the unlearning adapters are distributed to clients that do not require unlearning for retraining. During this phase, the original model is frozen, and only the independent unlearning adapters are trained. Ultimately, the unlearning adapters are integrated with the original model to yield a global unlearning model. This method leverages training on the remaining knowledge to effectively overwrite the knowledge that needs to be forgotten (i.e., knowledge overwriting), addressing the issue of indiscriminate unlearning. Moreover, introducing independent adapters facilitates rapid recovery of forgotten knowledge through their removal and significantly reduces unlearning costs by utilizing sparse parameters. In summary, FUSED achieves high performance, reversibility, and cost-efficiency in FU, making it suitable for scenarios involving client unlearning, class unlearning, and sample unlearning scenarios.

Contributions. The contributions are as follows:

- We propose FUSED, a reversible FU approach that retrains independent sparse adapters for unlearning. These adapters effectively mitigate unlearning interference while ensuring that the unlearning is reversible.
- We introduce the CLI method, which accurately identifies model layers sensitive to knowledge changes and constructs sparse unlearning adapters, significantly reducing the parameter scale and unlearning costs.
- We theoretically and experimentally prove the effectiveness of the proposed method across different unlearning scenarios in FL, including client unlearning, class unlearning, and sample unlearning.

2. Related work

Machine unlearning. Currently, most researchers focus on machine unlearning (MU) within centralized scenarios [7, 24, 28, 48]. Mainstream methods can be classified into two categories: data manipulation and model manipulation [31]. Data manipulation includes data mixing and data partitioning. The former fine-tunes the model to forget specific samples by introducing interference data or by replacing existing data [11, 14, 34, 36, 49]. In contrast, the latter divides the training dataset into multiple subsets and retrains only the subset that contains the data to be forgotten [2, 6, 8, 19, 32]. Model manipulation [9, 23] contains three strategies: model transformation, model pruning, and model replacement. Model transformation methods directly update the model parameters to offset the influence of forgotten samples on the model [13, 16, 20, 40]. Model pruning methods involve pruning from the original model [1, 17, 27, 38]. Model replacement methods compute nearly all possible sub-models and store them alongside the deployed model. When an unlearning request is received, only the sub-models affected by the unlearning operation need to be replaced. This method is commonly utilized in machine learning models such as decision trees [3, 33, 43].

Federated unlearning. Unlike centralized unlearning, FU [26] expands the unlearning objectives to client unlearning [35], sample unlearning, and class unlearning [15, 42]. In this context, commonly used unlearning methods can be classified into retraining-based methods [29] and parameter manipulation-based methods [4, 5, 18, 39]. Retraining-based methods means training a new model from scratch without unlearning data. For example, when a particular client needs to be forgotten, [27] have proposed approaches that retrain the remaining clients to obtain corrected gradient directions, which are then used to update the global model stored on the server. [29] utilized an improved quasi-Newton method to accelerate the training process. [35] reduces the time and computational resources required for retraining through clustering. Despite these efforts to mitigate the resource costs associated with retraining, the expenses remain unacceptable in real-world scenarios. Consequently, some researchers have proposed parameter manipulation-based unlearning methods. For instance, [38] focuses on classification tasks using CNN models and achieves unlearning classes by pruning class-related channel parameters. Furthermore, [41] eliminates the contribution of a target client by subtracting the accumulated historical updates from the global model. It then uses the old global model as a teacher model to train the unlearning model, employing knowledge distillation techniques to restore the model's performance. Overall, current research on FU is still limited, primarily focusing on client unlearning. Additionally, the issues of knowledge interference and irreversibility have not been adequately considered.

3. Problem formulation

Centralized machine unlearning. We denote $\mathcal{D}^u$ as the data to be forgotten, and $\mathcal{D}$ as the entire training dataset, $\mathcal{D} = (x_i, y_i)_{i=1}^n$. Then, $\mathcal{D}^r = \mathcal{D} \backslash \mathcal{D}^u$ represents the data to be retained. Let $\mathcal{M}^r$ denote the model before unlearning, $\mathcal{M}^f$ is the model after unlearning, and $\mathcal{FGT}(\cdot)$ denote the unlearning process. The unlearning can be represented as:

$$\mathcal{M}^f = \mathcal{FGT}(\mathcal{M}^r, \mathcal{D}^r, \mathcal{D}^u). \tag{1}$$

The objectives of FU are threefold: (a) minimizing the performance of $\mathcal{M}^f$ on $\mathcal{D}^u$; (b) maximizing the performance on $\mathcal{D}^r$, and (c) minimizing the resources consumed

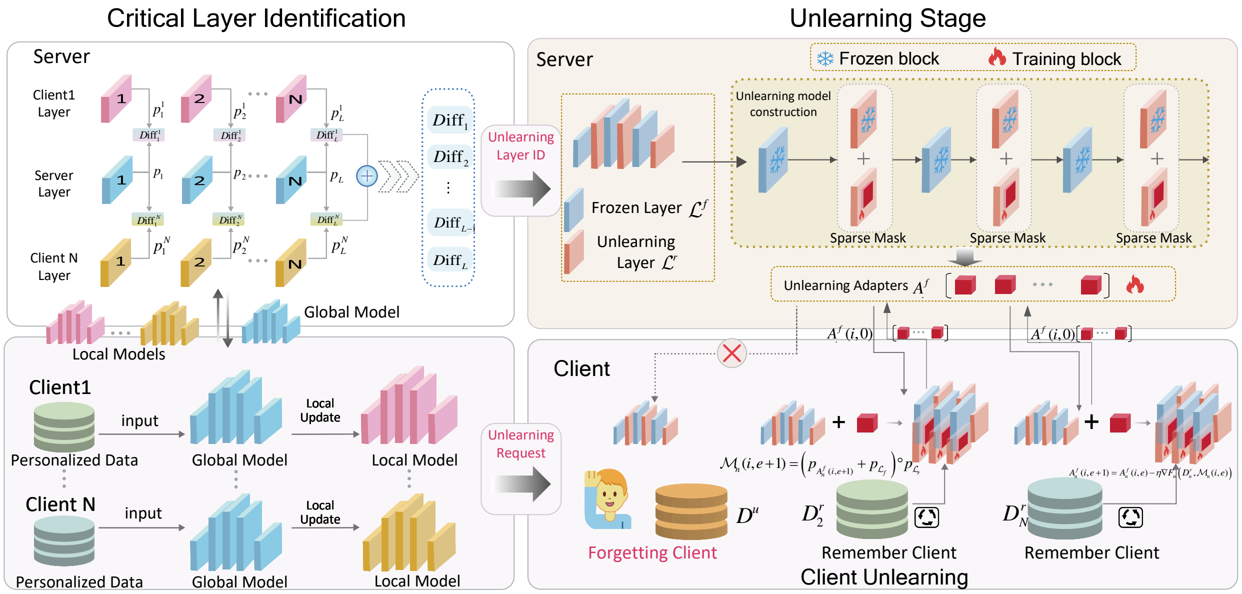


Figure 1. The figure illustrates the process of CLI (left) and unlearning (right). Left: the server computes the difference of each layer between the models uploaded by each client and the distributed one, identifying critical layers that are sensitive to knowledge. Right: a sparse adapter is constructed for each key layer, which is then independently trained on the remaining data.

by the unlearning process. Denoting $\mathcal{F}(\cdot)$ as the model test loss and $\mathcal{RC}(\cdot)$ as resource consumption, the above objectives can be respectively expressed as:

$$\max \mathcal{F}(\mathcal{M}^f, (x_i, y_i)), (x_i, y_i) \in \mathcal{D}^u, \tag{2}$$

$$\min \mathcal{F}(\mathcal{M}^f, (x_i, y_i)), (x_i, y_i) \in \mathcal{D}^r = \mathcal{D} \backslash \mathcal{D}^u, \tag{3}$$

$$\min \mathcal{RC}(\mathcal{FGT}(\mathcal{M}^r, \mathcal{D}^r, \mathcal{D}^u)). \tag{4}$$

Ideally, when a model is considered to have fully forgotten target knowledge, its performance should be equivalent to that of a model trained from scratch without ever seeing the forgotten data $\mathcal{D}^u$. This retraining ensures the worst performance on the forgotten data $\mathcal{D}^u$ and the best performance on the remaining data $\mathcal{D}^r$. However, this approach requires significant computational resources and preserving all historical training data, which is impractical in real-world scenarios. Therefore, we posit that the closer the performance of the model $\mathcal{M}^f$ on $\mathcal{D}^r$ and $\mathcal{D}^u$ is to that of a retrained model, the better the unlearning effect, while also striving to minimize resource expenditure on this basis.

Unlearning scenarios in FL. In consideration of the distributed nature of FL, traditional machine unlearning can be extended to client unlearning, class unlearning, and sample unlearning. In the case of client unlearning, we consider N clients, a set of unlearning clients N_u, with the unlearning dataset $\mathcal{D}^u = \{\mathcal{D}_k\}_{k \in N_u}$, and remember dataset $\mathcal{D}^r = \{\mathcal{D}_k\}_{k \in N \backslash N_u}$, where $\mathcal{D}_k$ represents the data of client k. The optimization objectives are:

$$\max \sum_{k \in N_u} \mathcal{F}(\mathcal{M}^f, \mathcal{D}_k), \tag{5}$$

$$\min \sum_{k \in N \backslash N_u} \mathcal{F}(\mathcal{M}^r, \mathcal{D}_k), \tag{6}$$

$$\min \mathcal{RC}(\mathcal{FGT}(\mathcal{M}^r, \{\mathcal{D}_k\}_{k \in N}). \tag{7}$$

Sample unlearning means forgetting a portion of data within a client. It is similar to client unlearning. In the context of class unlearning, let all client data classes be $\mathcal{C}$ and the classes to be unlearned be $\mathcal{C}^u$. The unlearning dataset can be represented as $\mathcal{D}^u = \{(x_i^k, y_i^k = c)\}_{c \in \mathcal{C}^u, (x_i^k, y_i^k) \in \mathcal{D}_k, k \in N}$, and the remember dataset as $\mathcal{D}^r = \{\mathcal{D}_k\}_{k \in N} \backslash \mathcal{D}^u$. The optimization objectives are:

$$\max \sum_{(x_i^{\cdot}, y_i^{\cdot}) \in \{\mathcal{D}_k\}_{k \in N}} \mathcal{F}(\mathcal{M}^f, (x_i^{\cdot}, y_i^{\cdot})|_{y_i^{\cdot} \in \mathcal{C}^u}), \tag{8}$$

$$\min \sum_{(x_i^{\cdot}, y_i^{\cdot}) \in \{\mathcal{D}_k\}_{k \in N}} \mathcal{F}(\mathcal{M}^r, (x_i^{\cdot}, y_i^{\cdot})|_{y_i^{\cdot} \notin \mathcal{C}^u}), \tag{9}$$

$$\min \mathcal{RC}(\mathcal{FGT}(\mathcal{M}^r, (x_i^{\cdot}, y_i^{\cdot})|_{y_i^{\cdot} \in \mathcal{C}})). \tag{10}$$

4. The proposed method: `FUSED`

`FUSED` involves a two-stage unlearning process (as shown in Fig. 1). The first stage is Critical Layer Identification (CLI), and the second stage is Unlearning via Sparse Adapters, which is based on the critical layers identified.

4.1. Critical layer identification

During the CLI phase, each client, coordinated by the server, participates in a federated iteration process. Clients receive the global model distributed by the server and train it using their local data before uploading it back to the server. Subsequently, the distance between the parameters of each layer in the models from different clients and those in the corresponding layers of the initial model is calculated by the server. The layers with the most significant parameter changes are obtained by averaging these distances.

Consider a global model with L layers, and N clients, each with an identical model structure. After local training, the parameters of these models differ across clients. Let p_l^n represent the parameters of the l-th layer of the n-th client, where $n = 1, 2, \cdots, N$ and $l = 1, 2, \cdots, L$. The initial distributed global model is denoted as $\mathcal{M}^r = \{p_1, p_2, \cdots, p_L\}$. After local training by the clients, the variation in the l-th layer of the model can be expressed as:

$$Diff_l = Diff_l^1(p_l^1, p_l) \oplus \cdots \oplus Diff_l^N(p_l^N, p_l), \quad (11)$$

where $Diff_l^n(p_l^n, p_l)$ represents the difference between the l-th layer of the n-th client's model and the l-th layer of the original model (need to be forgotten) distributed by the server. We utilize the Manhattan distance for measurement. Assuming that the dimensions of p_l^n and p_l are $k \times v$. The calculation process is as follows:

$$\mathcal{D}iff(p_l^n, p_l) = \sum\nolimits_{i=1}^{k} \sum\nolimits_{j=1}^{v} \left| p_{l,ij}^n - p_{l,ij} \right|. \quad (12)$$

The aggregation method of $\oplus$ is as follows:

$$\begin{aligned} &Diff_l^1(p_l^1, p_l) \oplus \mathcal{D}iff_l^N(p_l^N, p_l) = \tfrac{|D_1|}{\sum_{n=1}^{N} |D_n|} \\ &Diff_l^1(p_l^1, p_l) + \tfrac{|D_1|}{\sum_{n=1}^{N} |D_n|} \mathcal{D}iff_l^N(p_l^N, p_l), \end{aligned} \quad (13)$$

where $|D_i|$ represents the data volume of client i. Eventually, $LS = \left[\arg\max_l \{Diff_l\}, \cdots, \arg\min_l \{Diff_l\} \right]$ indicating the changes in different model layers is obtained. The first element corresponds to the layer index that is most sensitive to changes in client knowledge, while the last element corresponds to the most robust layer. To minimize resource cost, the subsequent unlearning process prioritizes unlearning in the most sensitive model layers.

4.2. Unlearning via sparse adapters

Based on the list obtained from CLI, given a preset value K for the number of layers to be unlearned, the first K indices in LS are designated for FU. Let $\mathcal{L}^f$ denote the set of layers that need to be unlearned, and $\mathcal{L}^r$ denote the remaining frozen layers. For each unlearning layer in $\mathcal{L}^f$, we discard most of the parameters in a random manner and leave only a small portion, forming a sparse parameter matrix A^f. During training, only A^f is trained while $\mathcal{L}^f$ remains unchanged. In the inference process, the new parameters of the unlearning layer are directly obtained by adding the parameters of A^f (denoted by p_{A^f}) and $\mathcal{L}^f$ (denoted by $p_{\mathcal{L}^f}$).

For the original model $\mathcal{M}^r$, the entire unlearning process can be divided into four stages: model distribution, local training, model uploading, and model aggregation. In `FUSED`, there are significant differences in the model distribution and local training stages compared to traditional FL. The following sections will primarily focus on these two stages. Firstly, during the model distribution stage, the model is only distributed to clients that contain the remembered dataset $\mathcal{D}^r$ (see Problem Formulation), and only unlearning adapters are transmitted. This means that in client unlearning scenarios, the clients to be forgotten will not receive the adapters (in class/sample unlearning, the classes/samples that need to be forgotten will no longer participate in training with the server). Additionally, the distributed model is a sparse matrix A^f, which significantly reduces communication overhead. Secondly, in the local training stage, for a client n, assuming the total number of local training epochs per federated iteration is E, then in the t-th federated iteration, the parameters of the model $\mathcal{M}_n(i,e)$ are $(p_{A_n^f(i,e)} + p_{\mathcal{L}^f}) \circ p_{\mathcal{L}^r}$. The training process is as follows:

$$A_n^f(i, e+1) = A_n^f(i, e) - \eta \nabla F_n(D_n^r, \mathcal{M}_n(i, e)), \quad (14)$$

$$\mathcal{M}_n(i, e+1) = (p_{A_n^f(i,e+1)} + p_{\mathcal{L}_f}) \circ p_{\mathcal{L}_r}, \quad (15)$$

where $e = 0, \cdots E - 1$, $F_n(D_n^r, \mathcal{M}_n(i, e))$ represents the loss and η denotes the learning rate. In each round of local training, $\mathcal{M}_n(i, e)$ is derived from the fusion of the original model $\mathcal{M}^r$ and the sparse matrix $A_n^f(i, e)$ obtained from the previous round. Each completed training round corresponds to a process of knowledge overwriting, during which the remaining knowledge is progressively enhanced. It is worth noting that during the training process, only $p_{A_n^f(i,e)}$ is updated. The other parameters, $p_{\mathcal{L}^f}$ and $p_{\mathcal{L}^r}$, remain frozen and are only used to compute the loss during inference. After local training is completed, each client uploads $p_{A_n^f(I,E)}$ to the server, which aggregates the updates using the FedAvg [30] method to obtain a new $p_{A_n^f(i+1)}$. After training, we need to concatenate the adapter with the corresponding unlearning layer of the original model to derive the global unlearning model. When the client's knowledge no longer needs to be unlearned, removing the unlearning adapters will effectively restore the original memory, thereby making the unlearning process reversible.

4.3. Algorithm

To elucidate the aforementioned training process more clearly, this section presents it through pseudocode (as

Algorithm 1 Our method `FUSED`

Input: Original model $\mathcal{M}^r$, number of iteration I, number of local training E, number of clients N, unlearning data $\mathcal{D}_u$, all clients' data $\mathcal{D}$

Output: Unlearning model $\mathcal{M}^f$

```
 1: $\mathcal{L}^f \leftarrow Unlearning\ Layer\ Identification$
 2: Adapters $A^f \leftarrow$ Random dropout parameters of $\mathcal{L}_f$
 3: for iteration $i = 0$ to $I$ do
 4:     if i=0 then
 5:         Server distribute $\mathcal{M}^r$ to all clients
 6:     end if
 7:     Server distribute adapters $A_f$ to clients maintaining
        $\mathcal{D}_r = \mathcal{D} \backslash \mathcal{D}_u$
 8:     for client $n = 1$ to $N$ do
 9:         Freezing the original parameter $\mathcal{M}^r$
10:         for local epoch $e = 0$ to $E - 1$ do
11:             Merge the $\mathcal{M}^r$ model with adapters to get
                $\mathcal{M}_n(e)$ (refer to Eq. (15))
12:             Update adapters to get $A^f_n(e + 1)$ (Eq. (14))
13:         end for
14:         Upload $A^f_n(E)$ to the server
15:     end for
16:     Server aggregates adapters to get $A^f$
17: end for
18: Merge $\mathcal{M}^r$ with $A^f$ to get $\mathcal{M}^f$
19: return Unlearning model $\mathcal{M}^f$
```

shown in Algorithm 1). The inputs for Algorithm 1 are the original model to be unlearned $\mathcal{M}^r$, the total number of federated training rounds I, the number of local training epochs E, the number of clients N, the forgetting dataset $\mathcal{D}^u$, and the dataset for all clients $\mathcal{D}$. The output is the model $\mathcal{M}^f$ after the unlearning process. First, based on the original model $\mathcal{M}^r$, a federated iteration is conducted to determine the model layers to be unlearned (details of CLI are displayed in the Supplementary Material). Using the indexes of unlearning layers, a series of unlearning adapters are constructed through random sparsification. Then, the FU process begins: during the first federated iteration, the server distributes both the unlearning adapters and the original model to the clients that contain the remembered dataset (lines 4-7 in Algorithm 1), excluding clients that only contain the forgetting dataset. Subsequently, the clients freeze the original model (line 9 in Algorithm 1) and merge the unlearning adapter with the original model (line 11 in Algorithm 1). The clients then train unlearning adapters using local data and upload it to the server for aggregation (line 16 in Algorithm 1). After I rounds of federated iterations, the final unlearning adapters are merged with the original model (line 18 in Algorithm 1) to obtain the global unlearned model $\mathcal{M}^f$.

4.4. Theoretical analysis

In this section, we will theoretically analyze how knowledge overwriting happens in the training process of different tasks, providing theoretical support for the `FUSED`.

Suppose there are two learning tasks denoted as T_1 and T_2, each task has an input space X and an output space Y. The parameterized model is represented as a mapping $f(\Theta) : X \to Y$, Θ denotes an n-dimensional parameter vector of the neural network. For each task, we use a loss function $\mathcal{L}_T(\Theta)$ to measure the performance of the model.

The model learns on tasks T_1 and T_2 and obtains the optimized parameters Θ_1^* and Θ_2^*, which are, respectively, ϵ_1 and ϵ_2 away from the minimum value, where ϵ_1 and ϵ_2 are arbitrarily small positive numbers:

$$\Theta_1^* = \arg\min_{\Theta}(\mathcal{L}_{T_1}(\Theta) + \epsilon_1),\ \epsilon_1 > 0. \tag{16}$$

$$\Theta_2^* = \arg\min_{\Theta}(\mathcal{L}_{T_2}(\Theta) + \epsilon_2),\ \epsilon_2 > 0. \tag{17}$$

We have the following assumptions:

Assumption 1: *The loss function $\mathcal{L}_T(\Theta)$ is continuous and differentiable with respect to the parameters Θ.*

Assumption 2: *Near the optimized parameters Θ_1^*, the loss function $\mathcal{L}_{T_1}(\Theta)$ can be approximated using a first-order Taylor expansion.*

From the work of [22], we notice the agreement between the predictions of the original network and those of the linear model obtained from the first-order Taylor expansion of the network. Consider the linear model of the loss function $\mathcal{L}_{T_1}(\Theta_2^*)$ for the old task T_1 at the optimized parameters Θ_1^*:

$$\mathcal{L}_{T_1}(\Theta_2^*) \approx \mathcal{L}_{T_1}(\Theta_1^*) + \nabla_\Theta \mathcal{L}_{T_1}(\Theta_1^*)^{\mathrm{T}} \cdot (\Theta_2^* - \Theta_1^*). \tag{18}$$

Performance degradation $\Delta\mathcal{L}_{T_1}$ is defined as:

$$\Delta\mathcal{L}_{T_1} = \mathcal{L}_{T_1}(\Theta_2^*) – \mathcal{L}_{T_1}(\Theta_1^*) \approx \nabla_\Theta \mathcal{L}_{T_1}(\Theta_1^*)^{\mathrm{T}}(\Theta_2^* – \Theta_1^*). \tag{19}$$

Defining the task difference $\Phi(\Theta, T_1, T_2)$ as the cosine similarity between the gradients of two tasks.At the optimized parameters Θ_1^*, we have:

$$\Phi(\Theta_1^*, T_1, T_2) = \frac{\nabla_\Theta \mathcal{L}_{T_1}(\Theta_1^*)^{\mathrm{T}} \cdot \nabla_\Theta \mathcal{L}_{T_2}(\Theta_1^*)}{||\nabla_\Theta \mathcal{L}_{T_1}(\Theta_1^*)|| \cdot ||\nabla_\Theta \mathcal{L}_{T_2}(\Theta_1^*)||}. \tag{20}$$

The changing direction of the parameters Θ is influenced by the gradient of the new task T_2, that is:

$$\Theta_2^* - \Theta_1^* = -\eta \nabla_\Theta \mathcal{L}_{T_2}(\Theta_1^*), \tag{21}$$

where η represents the learning rate.

Substituting into the performance degradation approximation formula, we can obtain:

$$\Delta\mathcal{L}_{T_1} \approx -\eta ||\nabla_\Theta \mathcal{L}_{T_1}(\Theta_1^*)|| \, ||\nabla_\Theta \mathcal{L}_{T_2}(\Theta_1^*)|| \, \Phi(\Theta_1^*, T_1, T_2). \tag{22}$$

When the cosine similarity of the gradients of the loss function at tasks T_1 and T_2 is less than 0, the performance degradation of the old task is greater than 0. That is, when the knowledge of the tasks learned sequentially is opposite, the learning of new knowledge will overwrite the old knowledge that the model has mastered.

The aforementioned analysis is based on full training of model parameters. Specifically, in `FUSED`, only partial parameters of critical layers are updated with new data. Then, the result will be:

$$\begin{aligned} &\Theta_2^* - \Theta_1^* = -\eta \cdot (\mathbf{v} \circ \nabla_\Theta \mathcal{L}_{T_2}(\Theta_1^*)), \\ &\{\mathbf{v} \in \{0,1\}^n | P(v_i = 1) = \mathrm{p}, i \in \{1, \ldots, n\}\}, \end{aligned} \tag{23}$$

$\mathbf{v}$ is an n-dimensional binary vector. Each element of $\mathbf{v}$ takes the value of 1 with probability p. Then, we can get:

$$\begin{aligned} \mathbb{E}(\Delta\mathcal{L}_{T_1}) = &- \eta \cdot \mathrm{p} \cdot ||\nabla_\Theta \mathcal{L}_{T_1}(\Theta_1^*)|| \cdot \\ & ||\nabla_\Theta \mathcal{L}_{T_2}(\Theta_1^*)|| \cdot \Phi(\Theta_1^*, T_1, T_2). \end{aligned} \tag{24}$$

Therefore, it can be inferred from the above formula that when the angle between the gradients of the new task and the old task is greater than 90 degrees, the old knowledge learned by the model can also be covered by training some parameters on the new task.

5. Experiments

5.1. Experimental setting

The experiments are built on PyTorch 2.2.0, developing an FL framework comprising one server and 50 clients. The hardware environment uses an NVIDIA RTX 4090. Optimizers consisting of SGD and Adam are employed with a batch size of 128. The result is listed in Tab. 1.

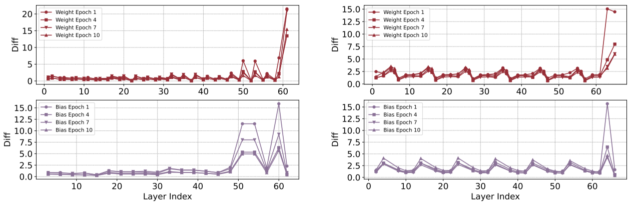
The datasets include FashionMNIST [44], Cifar10 and Cifar100 [21]. We use the Dirichlet function to partition the dataset and conduct tests under two conditions: $\alpha = 1.0$ and $\alpha = 0.1$. In Tab. 1, we primarily present the results for $\alpha = 1.0$; for the results under non-independent and identically distributed, please refer to the appendix. The model used for FashionMNIST is LeNet, while ResNet18 is employed for training on Cifar100. For Cifar10, training is conducted using both ResNet18 and a vision-based Transformer model called SimpleViT [47]. Baselines include Retraining, Federaser [27], Exact-Fun [45], and EraseClient [18]. Among these, Retraining is the upper bound of unlearning; it can achieve the effect of the model having never encountered the forgotten data.

Evaluations. We evaluate `FUSED` from multiple perspectives:

- ***RA & FA***: *RA* is the testing accuracy on the remaining dataset, which should be as high as possible to minimize knowledge interference. *FA* is the testing accuracy on the unlearned dataset, which should be as low as possible.
- ***Comp & Comm***: *Comp* is the time to complete predefined unlearning iterations; *Comm* denotes the data volume transmitted between a single client and the server.
- ***MIA***: the privacy leakage rate after unlearning, which is assessed in the context of membership inference attacks. A lower inference accuracy of the attack model indicates less privacy leakage.
- ***ReA***: the accuracy of relearning, which refers to the accuracy that can be achieved after a specified number of iterations to relearn the unlearned knowledge. If the unlearned knowledge is effectively erased, the accuracy achieved during the subsequent relearning will be lower.

5.2. Critical layer identification

Before conducting unlearning, it is essential to identify the layers that are sensitive to knowledge. We segment the client data using a Dirichlet distribution with a parameter α of 0.1 to enhance knowledge disparity among clients. For the Cifar10 and Cifar100 datasets, we employ the ResNet18 and SimpleViT models, while the LeNet model is utilized for FashionMNIST. After obtaining locally trained models from different clients, we can observe the average change in each layer. In Fig. 2, we present the Diff values for each layer of the ResNet18 and SimpleViT across different training iterations. We can see the last, second-to-last, sixth-to-last, and eighth-to-last layers of the ResNet18 model, and the last several layers of the Transformer model demonstrate heightened sensitivity to data variations across clients. Therefore, these layers will be designated as unlearning layers for sparse training in the subsequent unlearning process. In fact, with the increasing number of federated iterations, the global model's knowledge generalization ability improves, leading to a gradual reduction in the gap of Diff values between layers. As illustrated in Fig. 2, the gap is most pronounced when Epoch=1. Therefore, by comparing the model after a single federated iteration, it is possible to more precisely identify the critical layers.



(a) ResNet18 (b) Transformer

Figure 2. The average difference between local models and the server model across different models.

5.3. Results Analysis

Unlearning performance. In the client unlearning scenario, we use clients affected by Byzantine attacks as test

	Client Unlearning					*Class Unlearning*		*Sample Unlearning*		
	Retrain	Federaser	E-F	FUSED	E-C	Retrain	FUSED		Retrain	FUSED
	FashionMNIST-LeNet									
RA(↑)	**0.99**	0.99	0.99	**0.99**	0.09	0.99	**0.99**	*0A*(↑)	0.99	**1.00**
FA(↓)	**0.00**	0.00	0.00	**0.00**	0.11	0.00	**0.00**	*PS*(↑)	0.75	**1.00**
ReA(↓)	**0.77**	0.94	0.96	0.97	0.96	1.00	**1.00**	*ReA*	**0.15**	0.70
MIA(↓)	0.85	**0.47**	0.70	0.68	0.70	**0.27**	0.99	*MIA*	**0.53**	0.94
Comp(↓)	210.15	178.66	298.28	158.96	**26.31**	213.60	**81.94**	*Comp*	873.04	**872.65**
Comm(↓)	177K	177K	177K	**11K**	177K	177K	**11K**	*Comm*	177K	**11K**
	Cifar10-ResNet18									
RA(↑)	**0.71**	0.67	0.65	0.67	0.64	0.73	**0.73**	*0A*(↑)	**0.56**	0.52
FA(↓)	**0.04**	0.04	0.05	0.05	0.06	0.00	**0.00**	*PS*(↑)	**0.55**	0.54
ReA(↓)	0.49	0.48	**0.41**	0.42	0.56	1.00	**1.00**	*ReA*	**1.00**	1.00
MIA(↓)	0.78	0.67	**0.43**	0.65	0.78	**0.86**	0.96	*MIA*	**0.98**	0.99
Comp(↓)	434.39	990.91	1211.74	262.20	**233.45**	735.07	**183.02**	*Comp*	4619.82	**1253.98**
Comm(↓)	42.73M	42.73M	42.73M	**0.98M**	42.73M	42.73M	**0.98M**	*Comm*	42.73M	**0.98M**
	Cifar10-Transformer									
RA(↑)	0.33	0.21	0.33	**0.41**	0.35	0.27	**0.44**	*0A*(↑)	0.05	**0.33**
FA(↓)	0.08	0.09	0.07	**0.07**	0.07	**0.00**	0.00	*PS*(↑)	0.20	**0.54**
ReA(↓)	0.40	**0.25**	0.40	0.41	0.40	1.00	**0.80**	*ReA*	**0.03**	0.50
MIA(↓)	0.62	**0.41**	0.76	0.62	0.63	**0.64**	0.88	*MIA*	**0.85**	1.00
Comp(↓)	5388.83	1867.36	4929.45	**502.60**	3240.05	1207.56	**218.30**	*Comp*	628.44	**342.14**
Comm(↓)	36.21M	36.21M	36.21M	**0.71M**	36.21M	36.21M	**0.71M**	*Comm*	36.21M	**0.71M**
	Cifar100-ResNet18									
RA(↑)	**0.39**	0.19	0.25	0.36	0.35	**0.34**	0.30	*0A*(↑)	0.57	**0.79**
FA(↓)	0.01	0.01	**0.00**	0.01	0.01	0.00	**0.00**	*PS*(↑)	0.54	**0.66**
ReA(↓)	0.18	**0.13**	0.17	0.14	0.20	1.00	**1.00**	*ReA*	0.25	**0.00**
MIA(↓)	0.22	0.28	**0.08**	0.48	0.36	0.66	**0.10**	*MIA*	**0.98**	1.00
Comp(↓)	443.86	1000.75	1598.55	276.59	**235.65**	820.43	**188.00**	*Comp*	4121.99	**1580.91**
Comm(↓)	42.91M	42.91M	42.91M	**0.98M**	42.91M	42.91M	**0.98M**	*Comm*	42.91M	**0.98M**

Table 1. Main Results. "0A" represents the accuracy of class 0, while "PS" refers to the precision of the predicted class 0. The symbol ↑ indicates higher values are better, while ↓ indicates the opposite. "E-F" is the short for Exact-Fun, and "E-C" is EraseClient.

cases. The mode of attack is label flipping, where one client's label is maliciously manipulated, resulting in a demand for unlearning. From Tab. 1, it can be observed that among the three metrics that directly measure forgetting effects—*RA*, *FA*, and *ReA*—only FUSED is nearly on par with Rrtraining. This approach maintains a low accuracy on unlearned data while achieving a high accuracy on others, and even demonstrates overall superiority compared to Rrtraining, particularly in the Transformer model. It can be concluded that FUSED effectively unlearns the specified client knowledge while minimizing the impact on the original knowledge. This is attributable to freezing the parameters of the original model and only training the unlearning adapter, thereby avoiding direct modifications to the old knowledge and effectively reducing interference with the existing knowledge. Similarly, the same results are observed in class and sample unlearning.

Knowledge interference. To investigate the impact of unlearning on the overlapping knowledge across clients, we use the Cifar10 dataset, distributing 90% of the data labeled as 0 and all data labeled as 1 to a client that needs to be forgotten. The remaining data, labeled from 2 to 9, and 10% of the data labeled as 0, are randomly assigned to other clients. After unlearning, we evaluate the accuracy of the knowledge unique to the unlearning client (data labeled as 1), the accuracy of the overlapping knowledge (data labeled as 0 from the remaining clients), and the accuracy of the knowledge unique to the remaining clients (data labeled from 2 to 9). The final results are shown in Tab. 2. It can be observed that all methods completely forget the knowledge unique to the forgetting client, while only the FUSED method demonstrates improved performance on overlapping knowledge compared to Retraining. Therefore, FUSED can reduce knowledge interference.

Method	Federaser	Retrain	E-F	FUSED	E-C
F-Acc	0.00	0.00	0.00	0.00	0.00
C-Acc	0.14	**0.38**	0.07	0.37	0.06
R-Acc	0.27	0.65	0.64	0.65	**0.66**

Table 2. "F-Acc" is the accuracy of the knowledge unique to the unlearning client, "C-Acc" is for overlapping knowledge, and "R-Acc" is for the knowledge unique to the remaining clients

Unlearning cost. In the unlearning process, resource overhead is an inevitable problem. Tab. 1 primarily illustrates the consumption of computational and communication resources. Since the FUSED trains and transmits only the sparse adapters, it consistently demonstrates a significant advantage across nearly all unlearning scenarios and datasets. Additionally, in terms of storage resources, both Federaser and EraseClient require the retention of all client models and the global model during each round, which presents significant challenges regarding storage capacity. This demand increases exponentially with the number of clients and iterations, rendering it impractical in real-world applications. In contrast, FUSED only requires the storage of a complete global model and its adapters. Moreover, when compared to the retraining method, the retraining method achieves *RA/FA* values of 0.71/0.04 when data is complete, and FUSED achieves *RA/FA* values of 0.67/0.05. When we reduce the number of retraining data by half, FUSED maintains *RA/FA* values of 0.65/0.03, indicating no significant decline in unlearning performance. This suggests that FUSED can achieve results comparable to retraining with less data, thereby conserving storage resources.

Privacy protection. When unlearned data is users' privacy, even if the model shows great unlearning performance, an attacker may still be able to discern which data corresponds to unlearned private information and which does not, particularly in the context of member inference attacks. Therefore, it is crucial to evaluate the privacy leakage rate of the model after unlearning. The *MIA* values for FUSED are generally comparable to those of the Retraining method, and in most instances, they remain at a relatively low level. This indicates that FUSED's capability to mitigate privacy leakage is on par with that of other methods.

Ablation study. To illustrate the necessity of CLI, we conduct an ablation study using the Cifar10 dataset, with the experimental results presented in Fig. 3. In Fig. 3, "W/O CLI" denotes the effect of FUSED achieved by randomly selected layers. It is evident that, with the implementation of CLI, the accuracy of remaining knowledge is higher in both client unlearning and class unlearning scenarios. Although the disparity is smaller in sample unlearning, it still maintains a comparable level. This indicates that CLI can more accurately identify the model layers that are more sensitive to knowledge, thereby enhancing the unlearning effect.

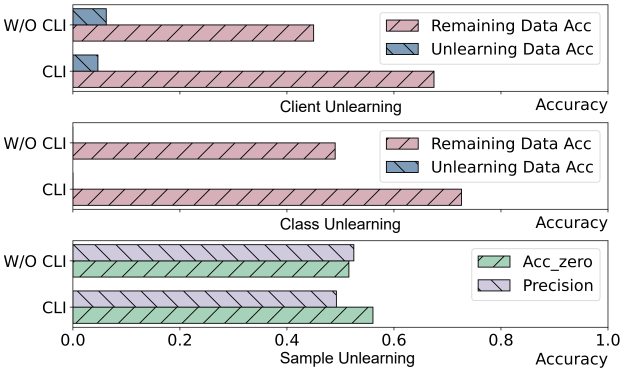


Figure 3. Ablation study of CLI.

6. Conclusion and Discussion

Conclusion. This paper focuses on the unlearning problem in FL. To address the challenges of indiscriminate unlearning, irreversible unlearning, and significant unlearning costs, we propose a reversible federated unlearning method via selective sparse adapters (FUSED). Firstly, by comparing the client model with the server model, we identify critical layers to unlearn. Then, independent sparse unlearning adapters are constructed for each unlearning layer. After that, only the sparse adapters are retrained, achieving efficient resource utilization. In this way, FUSED greatly reduces knowledge interference. Furthermore, independent adapters are easy to remove to facilitate memory recovery. Finally, we validate FUSED in client, sample, and class unlearning scenarios. The results show that FUSED's unlearning effectiveness matches that of Retraining, surpassing other baselines while significantly reducing costs.

Discussion. The proposed adapters can also serve as knowledge editors, adjusting the model's knowledge on different occasions. For instance, they can help unlearn private information and overcome catastrophic forgetting simultaneously. Moreover, when the knowledge editing requirements vary among clients, combinations of adapters can enhance global generalization. However, there are some limitations of FUSED we can not overlook, for example, it still requires a great number of remaining data to train the adapters. Some techniques like data compression are expected to solve this problem.

Acknowledgement. This work was supported by National Natural Science Foundation of China (62002369) and Postgraduate Scientific Research Innovation Project of Hunan (CX20230075).

References

[1] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. *Machine Learning*, 111(9):3203–3226, 2022. 2

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

[3] Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *International Conference on Machine Learning*, pages 1092–1104. PMLR, 2021. 2

[4] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *2023 IEEE Symposium on Security and Privacy*, pages 1366–1383. IEEE, 2023. 2

[5] Tianshi Che, Yang Zhou, Zijie Zhang, Lingjuan Lyu, Ji Liu, Da Yan, Dejing Dou, and Jun Huan. Fast federated machine unlearning with nonlinear functional theory. In *International conference on machine learning*, pages 4241–4268. PMLR, 2023. 2

[6] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*, pages 2768–2777, 2022. 2

[7] Kongyang Chen, Zixin Wang, Bing Mi, Waixi Liu, Shaowei Wang, Xiaojun Ren, and Jiaxing Shen. Machine unlearning in large language models, 2024. 2

[8] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 499–513, 2022. 2

[9] Somnath Basu Roy Chowdhury, Krzysztof Choromanski, Arijit Sehanobish, Avinava Dubey, and Snigdha Chaturvedi. Towards scalable exact machine unlearning using parameter-efficient fine-tuning. *arXiv preprint arXiv:2406.16257*, 2024. 2

[10] CNN. New york times sues openai and microsoft, 2023. [Online; accessed: 2024-08-02]. 1

[11] Daniel L. Felps, Amelia D. Schwickerath, Joyce D. Williams, Trung N. Vuong, Alan Briggs, Matthew Hunt, Evan Sakmar, David D. Saranchak, and Tyler Shumaker. Class clown: Data redaction in machine unlearning at enterprise scale, 2020. 2

[12] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. Formalizing data deletion in the context of the right to be forgotten. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 373–402. Springer, 2020. 1

[13] Aditya Golatkar, Alessandro Achille, and Stefano 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*, pages 9304–9312, 2020. 2

[14] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11516–11524, 2021. 2

[15] Hanlin Gu, Gongxi Zhu, Jie Zhang, Xinyuan Zhao, Yuxing Han, Lixin Fan, and Qiang Yang. Unlearning during learning: An efficient federated machine unlearning method, 2024. 2

[16] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models, 2023. 2

[17] Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1539–1547, 2020. 2

[18] Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. Federated unlearning: How to efficiently erase a client in fl? In *International Conference on Machine Learning*. PMLR, 2022. 2, 6

[19] Dongxiao He, Youyou Wang, Jinxin Cao, Weiping Ding, Shizhan Chen, Zhiyong Feng, Bo Wang, and Yuxiao Huang. A network embedding-enhanced bayesian model for generalized community detection in complex networks. *Information Sciences*, 575:306–322, 2021. 2

[20] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021. 2

[21] Alex Krizhevsky. Cifar-10 and cifar-100 datasets. http://www.cs.toronto.edu/ kriz/cifar.html, 2009. Accessed: 2024-08-14. 6

[22] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *NeurIPS*, pages 8570–8581, 2019. 5

[23] Guihong Li, Hsiang Hsu, Chun-Fu Chen, and Radu Marculescu. Fast-ntk: Parameter-efficient unlearning for large-scale models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 227–234, 2024. 2

[24] Meng Li and Haochen Sui. Causal recommendation via machine unlearning with a few unbiased data. In *AAAI 2025 Workshop on Artificial Intelligence with Causal Techniques*, 2025. 2

[25] Yichen Li, Qunwei Li, Haozhao Wang, Ruixuan Li, Wenliang Zhong, and Guannan Zhang. Towards efficient replay in federated incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12820–12829, 2024. 1

[26] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federated unlearning. *arXiv preprint arXiv:2012.13891*, 2020. 2

[27] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service*, pages 1–10. IEEE, 2021. 2, 6

[28] Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, Kush R. Varshney, Mohit Bansal, Sanmi Koyejo, and Yang Liu. Rethinking machine unlearning for large language models, 2024. 2

[29] Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1749–1758. IEEE, 2022. 1, 2

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017. 1, 4

[31] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning, 2022. 2

[32] Zhenwen Ren, Quansen Sun, and Dong Wei. Multiple kernel clustering with kernel k-means coupled graph tensor learning. In *Proceedings Of the AAAI Conference on Artificial Intelligence*, pages 9411–9418, 2021. 2

[33] Sebastian Schelter, Stefan Grafberger, and Ted Dunning. Hedgecut: Maintaining randomised trees for low-latency machine unlearning. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1545–1557, 2021. 2

[34] Rituparna Sinha, Rajat K Pal, and Rajat K De. Genseg and mr-genseg: A novel segmentation algorithm and its parallel mapreduce based approach for identifying genomic regions with copy number variations. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(1):443–454, 2020. 2

[35] Ningxin Su and Baochun Li. Asynchronous federated unlearning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2023. 2

[36] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 2

[37] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. *Computer Law & Security Review*, 34(2):304–313, 2018. 1

[38] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM Web Conference 2022*, pages 622–632, 2022. 1, 2

[39] Weiqi Wang, Zhiyi Tian, Chenhan Zhang, An Liu, and Shui Yu. Bfu: Bayesian federated unlearning with parameter self-sharing. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 567–578, 2023. 2

[40] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels, 2023. 2

[41] Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation, 2022. 2

[42] Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Yaohong Ding. Federated unlearning: Guarantee the right of clients to forget. *IEEE Network*, 36(5):129–135, 2022. 2

[43] Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pages 10355–10366. PMLR, 2020. 2

[44] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 6

[45] Zuobin Xiong, Wei Li, Yingshu Li, and Zhipeng Cai. Exact-fun: An exact and efficient federated unlearning approach. In *2023 IEEE International Conference on Data Mining*, pages 1439–1444. IEEE, 2023. 6

[46] Xin Yang, Hao Yu, Xin Gao, Hao Wang, Junbo Zhang, and Tianrui Li. Federated continual learning via knowledge fusion: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(8):3832–3850, 2024. 1

[47] Kentaro Yoshioka. vision-transformers-cifar10: Training vision transformers (vit) and related models on cifar-10. `https://github.com/kentaroy47/vision-transformers-cifar10`, 2024. 6

[48] Haibo Zhang, Toru Nakamura, Takamasa Isohara, and Kouichi Sakurai. A review on machine unlearning. *SN Computer Science*, 4(4):337, 2023. 2

[49] Peng-Fei Zhang, Guangdong Bai, Zi Huang, and Xin-Shun Xu. Machine unlearning for image retrieval: A generative scrubbing approach. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 237–245, 2022. 2