Efficient Federated Unlearning for Privacy-preserving Machine Learning in Decentralized Systems

Liu Jin, Haoda Wang, Jiantao Xu, and Chunhua Su*

University of Aizu, Fukushima, Japan
d8242103@u-aizu.ac.jp
d8242105@u-aizu.ac.jp
d8252108@u-aizu.ac.jp
chsu@u-aizu.ac.jp

Abstract. Recent advances in decentralized systems and the widespread adoption of smart devices have enabled a new era of distributed machine learning, offering real-time data processing and personalized services. However, this paradigm brings significant challenges in preserving data privacy, particularly as regulations emphasize the "right to be forgotten". In this paper, we propose a novel and efficient federated unlearning approach that effectively eliminates the influence of forgotten data while preserving the overall performance of the global model. Additionally, we design a contribution-based client selection algorithm that leverages historical local update information stored on the server, thereby minimizing both computational and communication overhead. We validate our approach through comprehensive experiments on three datasets: MNIST, Fashion-MNIST and CIFAR-10. These results demonstrate that our approach effectively removes the influence of forgotten data, maintains high global model accuracy, and reduces time overhead by over 60% compared to retraining from scratch.

Keywords: Federated Unlearning · Privacy-preserving · Client Selection

1 Introduction

With the rapid growth of artificial intelligence and connected devices, decentralized systems have become essential in fields like finance and healthcare [1]. In these systems, sensitive data remain on user devices, making centralized data aggregation impractical and risky [2]. To address these concerns, Federated Learning (FL) has emerged as an effective paradigm for privacy-preserving data collaboration [3]. FL allows devices to train models locally, sharing only model parameters rather than raw data. A typical FL workflow consists of two key steps: local training and global aggregation. First, devices update their local

* Chunhua Su is the corresponding author.

models using private data; then, they send only the updated parameters to a central server, which combines them to form a global model. This global model is subsequently broadcast back to the devices [4]. Such an iterative process ensures that sensitive data remain on local devices, thereby reducing both privacy risks and communication overhead. Despite its advantages, FL still faces challenges such as limited computational resources and discrepancies in training performance across devices [5]. More importantly, in decentralized systems, users may revoke access to their data or request complete removal of their data's influence from the trained model, introducing the crucial issue of Federated Unlearning (FU) [6].

With the enforcement of data privacy regulations such as the General Data Protection Regulation (GDPR) [7] and Health Insurance Portability and Accountability Act (HIPAA) [8], The "right to be forgotten" has become a fundamental aspect of data management in decentralized systems. In FL, even if a user deletes local data, the global model may still retain knowledge derived from that data. Furthermore, adversaries can exploit Membership Inference Attacks (MIA) or Gradient Inversion Attacks to reconstruct sensitive information from the supposedly forgotten data [9]. Existing FU methods typically rely on model retraining or gradient adjustments to remove the influence of target data. However, these approaches often suffer from high computational costs and may lead to performance degradation of the global model [10].

To enhance the efficiency of federated unlearning while minimizing its impact on global model performance, we propose a novel and efficient federated unlearning approach and design a contribution-based client selection algorithm to optimize the unlearning process. This algorithm leverages historical local update information stored on the server to identify the top k clients who have contributed the most to the global model, allowing selected clients participate in the subsequent unlearning phase. Compared to the conventional approaches such as retraining from scratch, our method reduces computational and communication overhead while maintaining good global model performance.

The main contributions are as follows.

1) We propose a novel and efficient federated unlearning approach that effectively eliminates the influence of forgotten data while preserving the overall performance of the global model.
2) We design a contribution-based client selection algorithm that utilizes historical local update information to improve the efficiency of the unlearning process.
3) We perform the simulation experiments on MNIST, Fashion-MNIST and CIFAR-10 which demonstrate that our method successfully removes the influence of forgotten data and significantly reduces the computational and communication overhead.

2 Related Work

With the increasing importance of privacy regulations such as the General Data Protection Regulation, the concept of machine unlearning has attracted significant attention, as models are required to have the "right to be forgotten" for the specific learned data. Traditional machine unlearning methods, such as the algorithm proposed by Cao et al. [11], leverage the transformation of the learning process into a summation form to enable efficient data deletion. However, these centralized approaches are not directly applicable to FL due to the decentralized nature of FL and its strict privacy constraints.

Wu et al. [12] presented a knowledge distillation–based unlearning method, in which the old global model acts as a teacher model during the unlearning phase, allowing the new model to learn without additional client participation. Wang et al. [13] introduced a channel pruning approach for unlearning, which evaluates the contribution of each CNN channel to different classes and employs Term Frequency Inverse Document Frequency(TF-IDF) scoring to identify and prune the channels that are most discriminative for the target class. This approach facilitates efficient unlearning which does not require retraining from scratch. Liu et al. [14] proposed FedEraser, which removes client contributions by utilizing historical parameter updates stored on the server, however, it incurs extra storage and computation overhead. Halimi et al. [15] proposed a local unlearning method based on projected gradient descent (PGD) to reverse the learning process and maximize the local loss, the server and the remaining clients conduct a small number of FL training rounds on the global model to recover global model performance.

3 Proposed Method

In this section, we propose a novel and efficient federated unlearning approach. We first introduce the overall workflow of the proposed method. Next, we detail the federated unlearning process. Finally, we describe the design of our client selection algorithm.

3.1 Overall Workflow

The overall workflow of efficient federated unlearning is summarized as Algorithm 1.

1) *Initialization:* Initialize the global model M^0 with random parameters.
2) *Client Selection:* Based on the historical update information stored on the server, the absolute values of the updates uploaded by each client are summed and then sorted in descending order for each epoch. The top k clients are selected as the subset C of clients for the current epoch. By selecting clients with larger historical contribution to the global model, the server aims to obtain more potentially effective updates for the current epoch.

3) *Local Model Updates:* In each epoch, for each client i in the selected subset C, the client receives the global model M^{t-1} from the central server. Using their local dataset, the client performs local training to obtain the local model in this epoch and uploads the model parameter updates $\triangle w_i^t(new)$ to the server.
4) *Global Model Aggregation:* After receiving the locally trained and uploaded model parameter updates $\triangle w_i^t(new)$ from the client, the server combines them with the historical model parameter updates $\triangle w_i^t(old)$ stored on the central server. Using $|\triangle w_i^t(old)|$ as the step size and $\frac{\triangle w_i^t(new)}{|\triangle w_i^t(new)|}$ as the direction, the new model parameter updates $\triangle w_i^t(now)$ are generated. The collected local model parameter updates $\triangle w_i^t(now)$ are then aggregated using a weighted averaging algorithm to generate the global model parameter updates $\triangle w^t$. Finally, the new global model M^t is obtained.
5) *Repeat:* Continue steps 2-4 until reaching the maximum number T of epochs.

Algorithm 1 Federated Unlearning in Decentralized Systems

Input: Maximum number of epochs T, client selection parameter k
Output: Global unlearning model M^T
Initialize global model M^0 with random parameters
for epoch $t = 1$ to T **do**
    Select a subset C of clients except for the forgotten client
    $C \longleftarrow ClientSelection(\triangle w_i^t(old), k)$
    for each client i in C **do**
        Receive global model M^{t-1} from the server
        Perform local training on dataset D_i to obtain model parameter updates $\triangle w_i^t(new)$
        Upload $\triangle w_i^t(new)$ to the server
    end for
    The server computes the model parameter updates $\triangle w_i^t(now)$
    $\triangle w_i^t(now) \longleftarrow \left|\triangle w_i^t(old)\right| \frac{\triangle w_i^t(new)}{|\triangle w_i^t(new)|}$
    The server aggregates the model parameter updates $\triangle w_i^t(now)$ using weighted averaging
    $\triangle w^t \longleftarrow \sum_{i=1}^{C} \frac{|D_i|}{|D_C|} \triangle w_i^t(now)$
    $M^t \longleftarrow M^{t-1} + \triangle w^t$
end for
Return M^T

3.2 Federated Unlearning

Federated unlearning refers to the process of effectively removing the influence of specific clients or data samples from the global model during federated learning to protect user privacy and comply with the "right to be forgotten" requirement. The training process in federated unlearning can be summarized as follows.

Learning Objective The main training objectives of federated unlearning are: 1) Removal of the influence of specific data: eliminate the impact of particular clients or data samples from the global model, ensuring that the model no longer retains knowledge of these data. 2) Preservation of model performance: while removing the influence of specific data, the goal is to maintain the model's performance on other data, avoiding degradation in accuracy.

For the first objective, the approach we adopt is to remove the forgotten data from the overall training dataset, resulting in a retrained dataset $D_{retrained} = D_{all} \setminus D_{ulearned}$. The global model is then reconstructed using federated unlearning on this dataset. Additionally, our method accelerates the reconstruction process of the global model by leveraging the historical clients updates stored on the server.

For the second objective, our approach is achieved by minimizing the global loss function $Loss$, which quantifies the difference between the global model's predicted output and the ground truth labels or values. The global loss function can be defined as:

$$Loss = \sum_{i=1}^{N} \frac{|D_i|}{\sum_{i=1}^{N} |D_i|} \cdot Loss_i. \tag{1}$$

Here, $|D_i|$ represents the size of the dataset of the i-th client, N is the total number of retrained clients, and $Loss_i$ denotes the local loss function of the i-th client, which measures the model's performance on its local dataset.

We assign each client a weight proportional to the size of their dataset, reflecting each client's contribution. This weighted scheme ensures that clients with larger datasets have a greater influence on the global model, while still considering the contributions of all clients in the federated unlearning process.

Client-Side Training Each selected client trains a local model using its own local dataset. The goal of local model training is to update the client's model parameters w_i through gradient descent or other optimization methods, in order to minimize the local loss function $Loss_i$. The local model training process can be described as follows:

$$Loss_i = \frac{\sum_{j=1}^{|D_i|} loss_j}{|D_i|}, \text{ for } \ j \in |D_i| \tag{2}$$

$$w_i^t = w_i^{t-1} - \eta \nabla Loss_i. \tag{3}$$

Here, w_i^t represents the model parameters of the i-th client at the training epoch t, η is the learning rate, and $\nabla Loss_i$ is the gradient of the local loss function.

After local model training is completed, each client uploads their model parameter updates to the server,

$$\triangle w_i^t(new) = w_i^t - w_i^{t-1} \tag{4}$$

where $\triangle w_i^t(new)$ denotes the local model parameter updates of each client after several local training epochs.

Server-Side Aggregation After the client-side training, the central server firstly combines the model parameter updates of the current global epoch $\triangle w_i^t$(new) uploaded by the client, with the historical model parameter updates $\triangle w_i^t$(old) stored on the server. This generates the new model parameter updates $\triangle w_i^t$(now) as follows:

$$\triangle w_i^t(now) = \left|\triangle w_i^t(old)\right| \frac{\triangle w_i^t(new)}{|\triangle w_i^t(new)|} \tag{5}$$

where $|\triangle w_i^t(old)|$ denotes the update step of the model parameter updates $\triangle w_i^t(now)$, $\frac{\triangle w_i^t(new)}{|\triangle w_i^t(new)|}$ denotes the update direction of $\triangle w_i^t(now)$.

And then, the new model parameter updates $\triangle w_i^t$(now) are aggregated using a weighted averaging algorithm to generate the global model parameter updates $\triangle w^t$. The aggregation process can be described as follows:

$$\triangle w^t = \sum_{i=1}^{C} \frac{|D_i|}{|D_C|} \triangle w_i^t(now) \tag{6}$$

where C is the total number of the selected clients in this global epoch.

Finally, the global model M^t based on the global model parameter updates $\triangle w^t$ for the current global epoch is generated.

$$M^t = M^{t-1} + \triangle w^t \tag{7}$$

The training process in federated unlearning can be executed iteratively, consisting of multiple rounds of client-side training and server-side aggregation. By repeatedly performing this process, the global model performance gradually improves over time as the participating clients collectively contribute their knowledge and insights.

3.3 Client Selection

Client selection is a technique used in federated learning to improve training efficiency by selecting appropriate clients to participate in each training epoch. Since federated learning involves distributed devices or clients for training, and these clients often vary in terms of data distribution, computational capabilities, and network conditions, it is essential to select the clients for each epoch based on a certain strategy. By choosing suitable clients for local training according to specific evaluation criteria, the overall federated learning process can significantly reduce communication overhead while still achieving a high global model performance. In this work, we employ the contribution-based client selection algorithm, which can be described as Algorithm 2. This algorithm aims to select clients that can contribute more to the global model, thereby enhancing training efficiency and improving the final model's accuracy.

In our experiments, the contribution-based client selection process computes the absolute values of the historical local updates stored by the server for each client and sums them. This is represented as $\sum|\triangle w_i^t(old)|$, where $\triangle w_i^t(old)$ denote the historical updates stored on the server. This summed value serves as an evaluation criterion to predict the potential contribution of each client's model update to the global model in the current training epoch. Then, the k largest summed values of these clients are selected, which represents as a subset C. These top k clients that predicted to make the greatest contributions are selected for local training, and it can greatly reduce the overall computational and communication overhead.

Algorithm 2 Client Selection

1: **Input:** Historical updates stored on the server $\triangle w_i^t(old)$, client selection parameter k
2: **Output:** Subset C of clients except for the forgotten client
3: Compute the absolute values of model parameter updates $\left|\triangle w_i^t(old)\right|$
4: Calculate the values of the sum of these numbers $\sum\left|\triangle w_i^t(old)\right|$for each client i
5: Sort clients in descending order based on $\sum\left|\triangle w_i^t(old)\right|$
6: Select top k clients as subset C
7: **Return** C

4 Experiment Evaluation

In this section, we first introduce the datasets required for the experiments, the corresponding network models, and other comparison methods. Then, we conduct experiments on the MNIST, Fashion-MNIST and CIFAR-10 to compare the accuracy of our method and other methods on the test datasets, the time cost for forgetting the target data, and prove the effectiveness of forgetting through membership inference attacks.

4.1 Experimental Setup

Datasets For our experiments, we use three datasets: MNIST, Fashion-MNIST and CIFAR-10.

MNIST It contains 70,000 grayscale images of handwritten digits (0-9), with 60,000 images in the training set and 10,000 images in the test set. Each image is 28x28 pixels in size, and the dataset is preprocessed so that the digits are centered and normalized. The pixel values range from 0 to 255, representing the intensity of grayscale. Along with the images, the dataset provides labels corresponding to the digits in the images, ranging from 0 to 9.

Fashion-MNIST It contains 70,000 grayscale images of fashion items, divided into 60,000 training images and 10,000 test images. Each image is 28x28 pixels in size, with the fashion items centered and normalized. The pixel values range from 0 to 255, representing the grayscale intensity. The dataset includes 10 classes of fashion products, such as T-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots, with each image labeled according to its corresponding category.

CIFAR-10 It consists of 60,000 color images, divided into 10 distinct classes, with 6,000 images per class. These classes represent various objects, including airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The images in the dataset are 32x32 pixels in size, and each image is labeled with the corresponding class. The CIFAR-10 dataset is split into 50,000 training images and 10,000 test images.

Model Architectures In our experiments, we use a simple CNN model with two convolutional layers. For the CNN model used with MNIST, the first convolutional layer has a 5×5 kernel with 20 output channels, followed by a 2×2 max pooling layer. The second convolutional layer also has a 5×5 kernel with 50 output channels, followed by another 2×2 max pooling layer. For the CNN model used with Fashion-MNIST, the first convolutional layer has a 3×3 kernel with 32 output channels, followed by a 2×2 max pooling layer. The second convolutional layer also has a 3×3 kernel with 64 output channels, followed by another 2×2 max pooling layer. For the CNN model used with CIFAR-10, the first convolutional layer has a 5×5 kernel with 32 output channels, followed by a 2×2 max pooling layer. The second convolutional layer also has a 5×5 kernel with 64 output channels, followed by a 2×2 max pooling layer.

Comparison Methods In our experiments, we compare our approach with two different methods: Federated Averaging (FedAvg), Federated Retrain (FedRetrain). FedAvg is the most widely used and classic federated learning algorithm without unlearning. It allows multiple clients to collaboratively train a shared machine learning model while keeping their data locally, thus ensuring privacy. FedRetrain is a classic federated unlearning algorithm that involves retraining the model from scratch after removing the target data. FedRetrain starts with a training set containing the remaining data and completely retrains the model to ensure that the influence of the deleted data is entirely eliminated.

4.2 Results and Analysis

Comparison of accuracy on test dataset Figure 1 demonstrates that our approach remains highly competitive across all datasets, achieving accuracy levels comparable to FedRetrain. On MNIST, it reaches 0.9826, only slightly below FedAvg (0.9837) and FedRetrain (0.9838), suggesting minimal performance loss due to unlearning. For the more complex Fashion-MNIST dataset, our accuracy

is 0.8608, slightly lower than FedAvg (0.8704) and FedRetrain (0.8678). Importantly, on CIFAR-10, our method achieves 0.5194 accuracy, performing better than FedAvg (0.5167) and FedRetrain (0.5109). Overall, these results show that our approach provides performance close to the baseline FedRetrain method, highlighting its practical usefulness for generating global unlearned models.

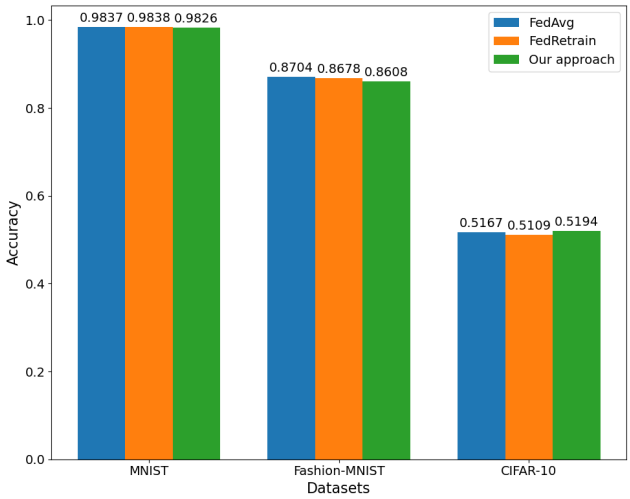

Fig. 1: Comparison of accuracy on test dataset

Comparison of membership inference attack (MIA) accuracy To address privacy concerns, we assess how effectively the unlearning global model forgets the removed data—i.e., how much information about the removed data remains. In our experiments, we employ a MIA to measure the degree of forgetting. MIA determines whether a specific data sample was part of the model's training set by querying the model and analyzing its outputs. Lower MIA accuracy thus indicates a smaller residual influence of the forgotten data on the model.

Figure 2 shows that our approach demonstrates significant improvements in privacy protection. FedAvg exhibits high attack accuracy across all datasets (0.9738, 0.9757, 0.9753), indicating severe vulnerability to MIA. FedRetrain effectively reduces attack accuracy to 0.486 (MNIST), 0.5485 (Fashion-MNIST), and 0.4833 (CIFAR-10), showcasing its ability to mitigate privacy risks through complete retraining. Our approach achieves comparable results, with attack accuracies of 0.513 (MNIST), 0.5015 (Fashion-MNIST), and 0.474 (CIFAR-10). The experimental results clearly show that, like the models retrained from scratch with FedRetrain, our global unlearning model also contains very little

information about the forgotten data. Our approach efficiently eliminates the impact of forgotten data from the original global model.

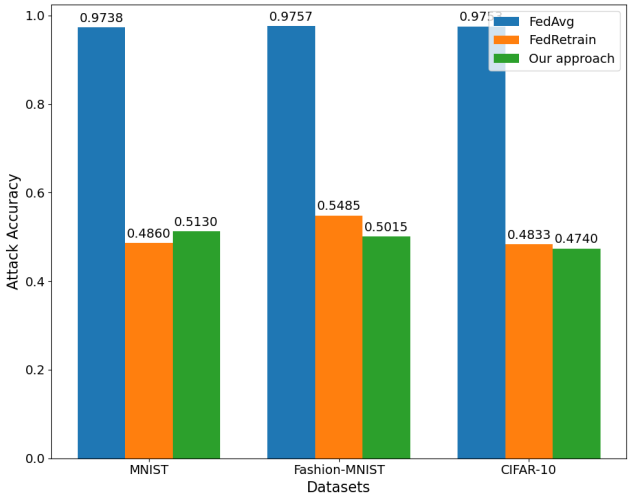

Fig. 2: Comparison of membership inference attack accuracy

Comparison of time cost Figure 3 shows that our approach significantly reduces time overhead compared to both baseline methods. FedAvg incurs the highest overhead, with 780.21s (MNIST), 795.33s (Fashion-MNIST), and 1006.77s (CIFAR-10). FedRetrain also requires substantial computation time at 708.31s, 699.14s, and 850.64s, respectively. In contrast, our approach greatly reduces the time overhead, requiring only 241.27s (MNIST), 255.47s (Fashion-MNIST), and 303.55s (CIFAR-10). This substantial reduction, exceeding 60% in time savings, highlights the efficiency of our unlearning mechanism based on client selection, making it highly suitable for real-world federated learning applications where computational resources and response times are critical constraints.

5 Conclusion

In this paper, we propose a novel and efficient federated unlearning approach for privacy-preserving machine learning in decentralized systems. Our approach effectively removes the influence of forgotten data while maintaining the performance of the global model. To optimize the unlearning process, we design a contribution-based client selection algorithm that leverages historical local updates stored on the server to identify the most influential clients, thereby reducing both computational and communication overhead. We validate our proposed

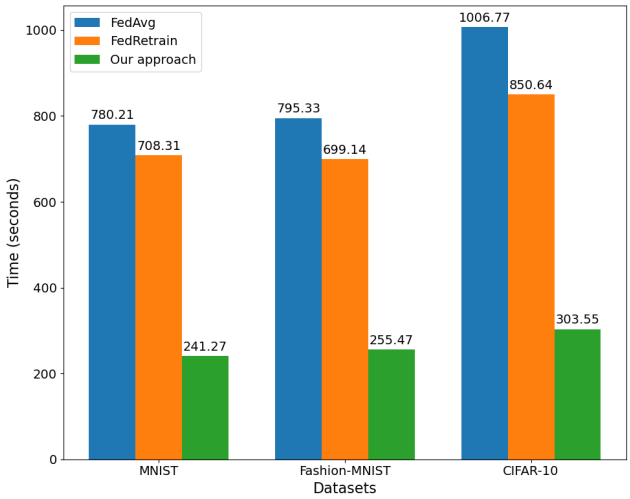

Fig. 3: Comparison of time cost

method through extensive experiments on datasets, including MNIST, Fashion-MNIST and CIFAR-10. The results demonstrate that our approach not only successfully eliminates the contributions of forgotten data but also preserves high global model accuracy. Furthermore, compared to conventional unlearning approaches, our method significantly reduces the time overhead by more than 60%. In summary, our work provides an effective and scalable solution for federated unlearning in decentralized systems. Future research will focus on extending this approach to more complex and heterogeneous environments, improving robustness against adversarial attacks, and exploring adaptive client selection mechanisms to further enhance efficiency and privacy protection.

Acknowledgment

This work was partially supported by JSPS Grant-in-Aid for Scientific Research (C) 23K11103.

References

1. Harsh Kasyap and Somanath Tripathy. Privacy-preserving decentralized learning framework for healthcare system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(2s):1–24, 2021.
2. Meng Sun and Wee Peng Tay. On the relationship between inference and data privacy in decentralized iot networks. *IEEE Transactions on Information Forensics and Security*, 15:852–866, 2019.
3. Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

4. Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.
5. Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. A survey on federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 9(1):1–24, 2021.
6. Fei Wang, Baochun Li, and Bo Li. Federated unlearning and its privacy threats. *IEEE Network*, 38(2):294–300, 2023.
7. Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.
8. Lawrence O Gostin, Laura A Levit, and Sharyl J Nass. Beyond the hipaa privacy rule: enhancing privacy, improving health through research. 2009.
9. Kongyang Chen, Yaping Chai, Weibin Zhang, Shaowei Wang, Jiaxing Shen, et al. Federated unlearning for human activity recognition. *arXiv preprint arXiv:2404.03659*, 2024.
10. Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, Xingliang Yuan, and Xiaoning Liu. A survey on federated unlearning: Challenges, methods, and future directions. *ACM Computing Surveys*, 57(1):1–38, 2024.
11. Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE, 2015.
12. Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441*, 2022.
13. 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.
14. Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th international symposium on quality of service (IWQOS)*, pages 1–10. IEEE, 2021.
15. Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. Federated unlearning: How to efficiently erase a client in fl? *arXiv preprint arXiv:2207.05521*, 2022.