An Empirical Study of Federated Unlearning: Efficiency and Effectiveness

Thai-Hung Nguyen 20hung.nt@vinuni.edu.vn
Hong-Phuc Vu 20phuc.vh@vinuni.edu.vn
College of Engineering & Computer Science, VinUniversity, Hanoi, Vietnam
Dung Thuy Nguyen dung.nt2@vinuni.edu.vn
VinUni-Illinois Smart Health Center, VinUniversity, Hanoi, Vietnam
Tuan Minh Nguyen tuan.nm@vinuni.edu.vn
Khoa D Doan doankhoadang@gmail.com
Kok-Seng Wong wong.ks@vinuni.edu.vn
VinUni-Illinois Smart Health Center, VinUniversity, Hanoi, Vietnam
College of Engineering & Computer Science, VinUniversity, Hanoi, Vietnam

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

The right to be forgotten (RTBF) is a concept that pertains to an individual's right to request the removal or deletion of their personal information when it is no longer necessary, relevant, or accurate for the purposes for which it was initially collected. Machine Learning (ML) models often rely on large, diverse datasets for optimal performance. Hence, when an individual exercises the RTBF, it can impact the ML model's performance and accuracy. In the context of Federated Learning (FL), where a server trains a model across multiple decentralized devices without moving data away from clients, implementing the RTBF in FL presents some unique challenges compared to traditional ML approaches. For instance, the decentralized nature makes it challenging to identify and remove specific user data from the model. Although various unlearning methods have been proposed in the literature, they have not been well investigated from the efficiency perspective. To fill this gap, this paper presents an empirical study to investigate the impacts of various unlearning methods. Our experiments are designed in diverse scenarios involving multiple communication and unlearning rounds using three datasets, MNIST, CIFAR-10, and CIFAR-100. We utilize backdoor attack and Cosine Similarity to assess the effectiveness of each unlearning method. The findings and insights from this research can be integrated into FL systems to enhance their overall performance and effectiveness. Our research codes are available on GitHub at https://github.com/sail-research/fed-unlearn.

Keywords: The Right To Be Forgotten; Federated Learning; Machine Unlearning; Federated Unlearning; Data Privacy Protection

1. Introduction

Federated Learning (FL) (McMahan et al., 2017) has gained attention as a technique for collaborative machine learning (ML) without sharing private data. FL offers advantages like data privacy, reduced transmission and storage needs, and scalability. However, it introduces challenges related to the right to be forgotten (RTBF). RTBF requires unlearning user data while maintaining model accuracy, aligning with privacy regulations like GDPR (Voigt and Von dem Bussche, 2017) and CCPA (Pardau, 2018). Federated unlearning (FedUnlearn) addresses RTBF in FL by removing client data while preserving performance. This essential process ensures compliance with privacy regulations, empowers individuals with control over their data, and enables seamless data removal and reintegration. In real-life scenarios such as healthcare collaborations, financial institutions, social media platforms, and autonomous vehicles, where privacy and data removal are paramount, FedUnlearn proves crucial. By enabling the targeted removal of sensitive data from the shared model, FedUnlearn effectively accommodates situations where institutions or individuals withdraw, exercise RTBF, or request data removal. Through the application of FedUnlearn, organizations can uphold privacy regulations, protect user information, and simultaneously maintain the accuracy and performance of FL systems across diverse domains. FedUnlearn also proves valuable when clients provide low-quality data or launch data poisoning attacks, compromising the FL system's security and reliability (Bagdasaryan et al., 2020; Xie et al., 2020).

Several recent works have focused on the problem of machine unlearning (Graves et al., 2021; Baumhauer et al., 2022; Sekhari et al., 2021). However, these approaches are not directly applicable in the context of FL due to two main challenges. First, FL involves an iterative learning process where participating clients contribute to building the final model, and second, the aggregator (i.e., server) does not have access to the training data held by the clients. A straightforward approach for FedUnlearn is to retrain the model from scratch after removing the relevant data samples from the corresponding client(s). However, this approach is computationally expensive. Recently, some works (Liu et al., 2021; Wu et al., 2022b) have been proposed to speed up this process, but they are either impractical for real-life scenarios (Liu et al., 2021) or require the server to store the history of parameter updates (Wu et al., 2022b). In addition, deciding who should conduct unlearning in FL is not straightforward, and local unlearning may not be sufficient as other checkpoints in the federation may still retain the memory of erased data. Given the above problems, we aim to address three research questions in this study:

***RQ1:* What are the impacts of unlearning methods on different datasets**? We consider several FedUnlearn methods, such as FedEraser (Liu et al., 2021), Projected Gradient Ascent (PGA) (Halimi et al., 2022), and our proposed methods (Continue to Train, Retrain, and Flipping) to investigate the impact of unlearning methods on MNIST, CIFAR-10, and CIFAR-100 datasets.

***RQ2:* What are the effects of early and late unlearning on FL performance**? We evaluate the FL performance when the client request to unlearn at an earlier training round and at a late round where the model is about to converge.

***RQ3:* How effective is each unlearning method to facilitate returning clients**? We assess the effectiveness of the unlearning methods for returning clients. Specifically, how fast can each method facilitate returning clients who return to the FL process?

To evaluate unlearning methods, we adopt the approach described in (Wu et al., 2022a), employing the backdoor samples during client updates. This backdoor attack does not impact the regular input performance of the global model but distorts predictions when specific inputs with the backdoor pattern are triggered. Additionally, we employ Cosine Similarity to compare the effectiveness of each method.

1.1. Our Contributions

Our main contributions can be summarized as follows:

- Experimental analysis: Rigorous experiments compare the performance of FedUnlearn techniques in removing client contributions from the global model.
- Evaluation methodology: The backdoor attack method is employed to assess the effectiveness of unlearning, measuring the distortion of predictions triggered by specific inputs with the backdoor pattern.
- Performance evaluation on diverse datasets: FedUnlearn techniques are evaluated on MNIST (LeCun et al., 1998), CIFAR-10, and CIFAR-100 (Krizhevsky and Hinton, 2009) datasets, offering insights into their efficiency across different data domains.
- Advancing robust FL systems: The findings contribute to developing secure FL systems, aiding practitioners in selecting appropriate FedUnlearn methods to preserve model accuracy while ensuring data privacy and regulatory compliance.

Roadmap. This paper is organized as follows: Section 2 introduces the problem of FL and provides an overview of the FedUnlearn approach, along with a review of related works and challenges in FL. In Section 3, we describe our experimental methodology. Moving on to Section 4, we present and analyze the results, discussing the pros and cons of various unlearning methods in FL scenarios. Finally, Section 5 concludes the paper by summarizing the main contributions and outlining future research directions.

2. Background and Related Works

2.1. Federated Learning

We consider an FL setting where a central orchestration server $\mathcal{S}$ together with m clients participating in a training task. Each client holds a private dataset $\mathcal{D}_i$. During each round, K clients are dynamically selected (with $1 \leq K \leq m$), ensuring adaptability. The total data points across the K clients participating in one round amount to n, with each client k possessing n_k data points. In general, FL involves the following phases:

***Phase 1 (FL Initialization)*:** The central orchestration server $\mathcal{S}$ initiates the global model's weights and hyperparameters, including the number of FL rounds, local epochs, selected client size per round, and local learning rate.

***Phase 2 (Local Model Training and Update)*:** all selected clients $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K$ receive the current global weights $\mathbf{w}_0$ from $\mathcal{S}$. Next, each $\mathcal{C}_i$ updates its local model parameters $\mathbf{w}_i^t$ using local datasets $\mathcal{D}_i$, where t denotes the current iteration round. Upon completing the local training, all selected clients send the local weights to $\mathcal{S}$ for model aggregation.

***Phase 3 (Global Model Aggregation and Update Phase)*:** $\mathcal{S}$ aggregates the received local weights and sends back the aggregation result to the clients for the next round of training. In Phase 2, the goal of each $\mathcal{C}_i$ is to obtain the optimal local model parameters $\hat{\mathbf{w}}_i^t$ in round t by minimizing the loss function $F_i(\mathbf{w})$ formulated as follows:

$$\hat{\mathbf{w}}_i^t = \arg\min_{\mathbf{w}} F_i(\mathbf{w}) = \arg\min_{\mathbf{w}} \frac{1}{n_i} \sum_{j \in \mathcal{D}_i} \mathcal{L}_j(\mathbf{w}), \tag{1}$$

where $\mathcal{D}_i$ is the set of indexes of data samples on the client i. In Phase 3, the goal of $\mathcal{S}$ is to obtain the optimal global model parameters $\hat{\mathbf{w}}^t$ by minimizing the loss function $F(\mathbf{w}^t)$ formulated as follows:

$$\hat{\mathbf{w}}_g^t = \arg\min_{\mathbf{w}} F(\mathbf{w}) = \arg\min_{\mathbf{w}} \sum_{k=1}^{K} \frac{n_k}{n} F_k(\mathbf{w}), s.t., F_k(\mathbf{w}) = \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \mathcal{L}_i(\mathbf{w}) \tag{2}$$

To achieve this, the server employs a model update aggregation rule, such as FedAvg (McMahan et al., 2016), to merge all updates and derive a new global model. The FL process persists until the maximum round limit is reached, or the global model surpasses the accuracy threshold τ. Subsequently, $\mathcal{S}$ concludes the FL by aggregating local updates and disseminating the final global model to $\mathcal{C}$.

2.2. Federated Unlearning (FedUnlearn)

We now formalize the data erasure problem in FL. In our setting, $\mathcal{S}$ controls the whole training process by collecting and aggregating model updates uploaded by the clients $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K\}$ each client i holds a different local dataset $\mathcal{D}_i$. In particular, we enable clients to have control over data erasure, i.e., a client can request the server to delete its data. First, we explain the concept of the FedUnlearn operation. Here, we use the term "unlearned client" to refer to those clients that perform data erasure, i.e., $\mathcal{C}_{k_u} = \left\{\mathcal{C}'_1, \mathcal{C}'_2, ..., \mathcal{C}'_{k_u}\right\}$ where k_u is the index of the unlearned client set.

The FedUnlearn task involves eliminating the influence of all updates $\mathbf{w}_i^t$ from a target client i belonging to $\mathcal{C}_{k_u}$ from the global model $\mathbf{w}^t$. Essentially, it removes the contribution of any client i to the final global model $\mathbf{w}^t$ and outputs a new final model $\mathbf{w}_u^t$ as if the client i has never participated in the training procedure. The unlearning process can be generalized for a client set, i.e., $\mathcal{C}_{k_u}$. As a result, the new objective function can be expressed as:

$$F'(\mathbf{w}^t) = \frac{1}{n'} \sum_{i=1}^{n'} F_i(\mathbf{w}_i^t), \forall i \in \{\mathcal{C} \setminus \mathcal{C}_{k_u}\}, \tag{3}$$

where n' is the total number of remaining learned clients.

Objectives of FedUnlearn. An effective FedUnlearn method should meet the following objectives:

- *Erasure Efficiency Guarantees.* The method must ensure that deleted data does not impact the unlearned model and preserves the model's parameters. Backdoor triggers (Gu et al., 2017) are used for evaluating unlearning performance, in line with previous research (Halimi et al., 2022; Wu et al., 2022a).
- *Reduced Retraining Time.* An unlearning method should be faster than retraining the entire model from scratch, regardless of the number of data samples to be unlearned.
- *Comparable Accuracy.* The unlearning solution should strive to maintain a moderate accuracy gap compared to the baseline, even if high accuracy is not the primary focus.

2.3. Related Work

Machine Unlearning was first introduced by (Cao and Yang, 2015) as the process to erase the impact of a data sample on a trained model. Later, there have been several approaches, proposed in (Ginart et al., 2019; Guo et al., 2019; Wu et al., 2020; Izzo et al., 2021; Baumhauer et al., 2022), to solve the unlearning problem in centralized ML. However, these methods are ineffective regarding FL, mainly due to its decentralized property, which prohibits the central server from accessing the raw data during unlearning.

In recent years, several works have focused on FedUnlearn in the literature. For instance, FedEraser (Liu et al., 2021) was proposed to obtain the unlearned model by adjusting the historical parameter updates of the participating clients through the retraining process. Although this method can significantly reduce the retraining time while eliminating the influences of a federated client's data on the global model, it has several drawbacks. The main disadvantage is that it requires the central server to keep historical updates from clients, which takes up a lot of memory space and might lead to data leakages. Moreover, FedEraser only initiates the unlearning algorithm among the clients that request data removal. In contrast, other clients' models still hold the contributions of the data to be removed, which will later be aggregated in a global model. In addition, their proposed algorithm utilizes the latest rounds of local model updates to approximate the current unlearned model update. However, the data samples to be removed can be used for training from the beginning, and the utilized model updates may still consist of the trace of those samples.

Later, (Wu et al., 2022b) introduced an unlearning method that erases the historical updates from the target client and recovers the model performance through the knowledge distillation process. However, like FedEraser, this method requires all clients to send the parameter updates to the central server. In addition, the knowledge-distillation approach requires the server to possess some extra outsourced unlabeled data, which might not be applicable under strict privacy regulations. Another approach for the unlearning problem, which does not require the server to keep historical parameters, is to reverse the training process based on gradient descent by applying the reverse method of gradient ascent. In 2022, (Wu et al., 2022c) introduced a stochastic gradient ascent (SGA) to eliminate the influence of training data that belongs to the same class. Moreover, the research work also proposed an FU framework that combines SGA with elastic weight consolidation (EWC). The proposed framework performs well in client-unlearning and sample-unlearning problems (Wu et al., 2022c). In the same year, (Halimi et al., 2022) proposed a Projected Gradient Ascent (PGA), in which instead of using gradient ascent arbitrarily over the entire space of model parameters, they add some constraints to ensure that the unlearned model is sufficiently close to a reference model that has effectively learned the other clients' data distributions. This is to avoid obtaining an arbitrary model similar to a random one.

In this paper, we perform experimental studies to assess emerging unlearning methods. We evaluate their effectiveness and efficiency by measuring their unlearning capacity and speed. Our experiments facilitate method comparisons and the identification of strengths and weaknesses. Additionally, we analyze the impact of key factors on unlearning performance, such as unlearning timing and the number of unlearning rounds. Furthermore, we explore scenarios where previously departed clients rejoin the process to understand how unlearning methods accommodate returning clients.

3. Methodology

3.1. Experiment Design

In our experiments, we establish a scenario comprising a central server ($\mathcal{S}$) and five participating clients: client 0 ($\mathcal{C}_0$), client 1 ($\mathcal{C}_1$), client 2 ($\mathcal{C}_2$), client 3 ($\mathcal{C}_3$), and client 4 ($\mathcal{C}_4$). The data distribution across the five clients is independent and identically distributed (IID). Client 0 is removed from the FL training process during unlearning rounds, and backdoor samples are introduced into its training data at a backdoor ratio of 0.9. The trigger pattern used consists of a 3x3 square with fixed pixel values in the bottom right corner. The backdoor label is set to `9` for MNIST, `9` for CIFAR-10, and `99` for CIFAR-100 datasets (the value of the label set is 0-9, 0-9, and 0-99, respectively). The experiment unfolds across four distinct phases: FL training, Unlearning, Post-unlearning, and Returning phase.

In the initial Phase (1), referred to as "FL training", both the central server and all clients adhere to the FL framework detailed in Section 2.1 to collaboratively build the global model $\mathcal{M}$. Subsequently, in Phase (2), denoted as "Unlearning", we examine a scenario where client 0 exits the training process and requests the central server to eliminate its contributions from the global model. In response, the central server executes the unlearning method, resulting in the unlearned model $\mathcal{M}'$ broadcasted to the remaining clients. The experiment then transitions to Phase (3), designated as "Post-unlearning", where the remaining clients, alongside the central server, continue the FL process. Finally, in Phase (4), known as "Returning", client 0 reintegrates into the FL process.

In FedUnlearn, we configure specific factors—namely, the starting stage and duration of unlearning. To evaluate FedUnlearn's performance, we designed three scenarios with variations in these factors:

- **Late Unlearning (Scenario 1)**: Here, we assess the impact of delaying unlearning on existing methods. We extend the FL training phase, with 50 rounds for MNIST and 100 rounds for CIFAR-10 datasets, before initiating unlearning. This scenario helps us understand how delayed unlearning affects method effectiveness.

- **Early Unlearning (Scenario 2)**: In this scenario, we explore unlearning when $\mathcal{C}_0$ requests it within a reduced number of FL communication rounds. We shortened the FL training phase to 10 rounds for MNIST and 20 rounds for CIFAR-10 datasets. This analysis provides insights into early unlearning's impact on method performance.

- **Early Unlearning with small unlearning rounds (Scenario 3)**: Similar to Scenario 2, this scenario involves early unlearning, but we further limit the unlearning rounds to 1 for MNIST and 2 for CIFAR-10 datasets. By examining this setup, we investigate how varying unlearning rounds influence method effectiveness in early unlearning contexts.

In all three scenarios, identical model architectures were employed for both the server and the clients. For the MNIST dataset, a neural network configuration featuring two convolutional layers followed by two fully connected layers was utilized. In the case of the CIFAR-10 and CIFAR-100 datasets, the ResNet18 architecture (He et al., 2016) was employed. Details of the hyperparameters utilized in our experiments can be found in

Table 1: Hyperparameter details for Three scenarios in the FedUnlearn setup

Hyperparameter	Scenario 1		Scenario 2		Scenario 3	
	MNIST	CIFAR-10	MNIST	CIFAR-10	MNIST	CIFAR-10
FL communication rounds	50	100	10	20	10	20
Unlearning rounds	5	10	5	10	1	2
Post-unlearning rounds	15	30	15	30	15	30
Returning rounds	15	30	15	30	15	30

Table 1. Additional information pertaining to the experimental setup and result analysis for CIFAR-100 is provided in the supplementary material.

3.2. Datasets

We utilize three popular datasets, including MNIST, CIFAR-10, and CIFAR-100.

- ***MNIST***: MNIST dataset is a collection of handwritten digits. It has a training set of 60000 images and a test set of 10000 images. The images are size-normalized and centered in a fixed size of 28×28. Each image is labeled, ranging from 0 to 9, corresponding to 10 digits.
- ***CIFAR-10***: CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, including airplanes, automobiles, birds, and some other animals and vehicles. The dataset is divided equally, with 6000 images per class. There are 50000 training images and 10000 test images.
- ***CIFAR-100***: CIFAR-100 dataset has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

3.3. FedUnlearn Methods

To achieve the objectives described in Section 2.2, we have selected the following unlearning methods for our study:

***Baseline* (Retrain).** This method involves retraining the model from scratch without the participation of the client who wishes to be removed. While highly effective in ensuring no client-specific information remains in the model, it can be computationally expensive and impractical for real-world scenarios, except in early training rounds. We use this method as our baseline and denote the resulting unlearned model as $\mathcal{M}^*$.

Continue to Train. This approach addresses the catastrophic forgetting phenomenon (French, 1999) by allowing targeted clients to opt out of federation without participating in the unlearning process. The learning process continues without their involvement.

Flipping. We explore the "flipping" method, which leverages the hypothesis that training with the same data but with different label sets can make the model confusing. In the unlearning phase, the client to be unlearned trains a local model with label-flipped data samples, preserving features while generating random labels. After predefined iterations, the local model is submitted to the server and aggregated.

Projected Gradient Ascent (PGA) (Halimi et al., 2022)**.** PGA reverses the learning process for the client to be erased by maximizing the local empirical loss. In particular, instead of arbitrarily optimizing across the entire space of model parameters, the unlearning problem is formulated as a constrained maximization problem by restricting to an ℓ_2-norm ball surrounding a reference model. The reference model is often the average of other clients' local models (excluding the target client) from the previous FL training round.

FedEraser (Liu et al., 2021)**.** FedEraser reconstructs the unlearned model using historical parameter updates of federated clients stored on the central server. Given the retained updates U and the targeted client to be removed, the unlearning process involves (i) calibration training at remaining clients to approximate the direction of the updates without the target client, (ii) the server calibrates the retained update and then (iii) are aggregated for unlearned model updating. After that, (iv) the server can thus renovate the global FL based on calibrated results. This method is proven to yield a significant speed-up to the reconstruction of the unlearned model while maintaining the model's efficacy.

We provide the pseudocode of PGA and FedEraser in the supplementary material.

3.4. Evaluation Metrics

To assess and compare the effectiveness of various unlearning methods, we employ metrics that quantify the degree of model unlearning and assess the similarity between the unlearned and original base models. This study utilizes the backdoor attack (Gu et al., 2017) and Cosine Similarity as evaluation metrics. The backdoor attack assesses the resilience of unlearning methods, while Cosine Similarity offers a quantitative measure of similarity between the unlearned and baseline models. Below, we provide details of these metrics.

Evaluation of FedUnlearn using Backdoor Attacks. In a backdoor attack, the attacker trains a model using inputs with specific triggers, like points, triangles, or crosses, causing the global model to behave anomalously when exposed to trigger-related inputs. For instance, adding a square pattern to the top-left corner of "1" inputs may trick the global model into misclassifying them as "7". Following previous works (Halimi et al., 2022; Wu et al., 2022a), we employ backdoor triggers to assess unlearning methods. An effective unlearning method should remove the influence of backdoor images from the global model. In other words, after unlearning, the global model's accuracy on backdoor images should significantly decrease. It's worth noting that we solely use backdoor triggers for evaluating unlearning methods and do not consider specific backdoor attack scenarios.

Evaluation of FedUnlearn using Cosine Similarity. We employ Cosine Similarity to measure the similarity between the unlearned model $\mathcal{M}$ and the baseline model $\mathcal{M}^*$ by comparing their predictions on the global test dataset $\mathcal{D}$. The similarity is calculated using the following formula:

$$\rho(\mathcal{M}, \mathcal{M}^*) = \frac{1}{N} \sum_{i=1}^{N} \cos \angle(\mathcal{M}(\mathcal{D}_i), \mathcal{M}^*(\mathcal{D}_i)) = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathcal{M}(\mathcal{D}_i)^T \mathcal{M}^*(\mathcal{D}_i)}{||\mathcal{M}(\mathcal{D}_i)||_2 \, ||\mathcal{M}^*(\mathcal{D}_i)||_2} \tag{4}$$

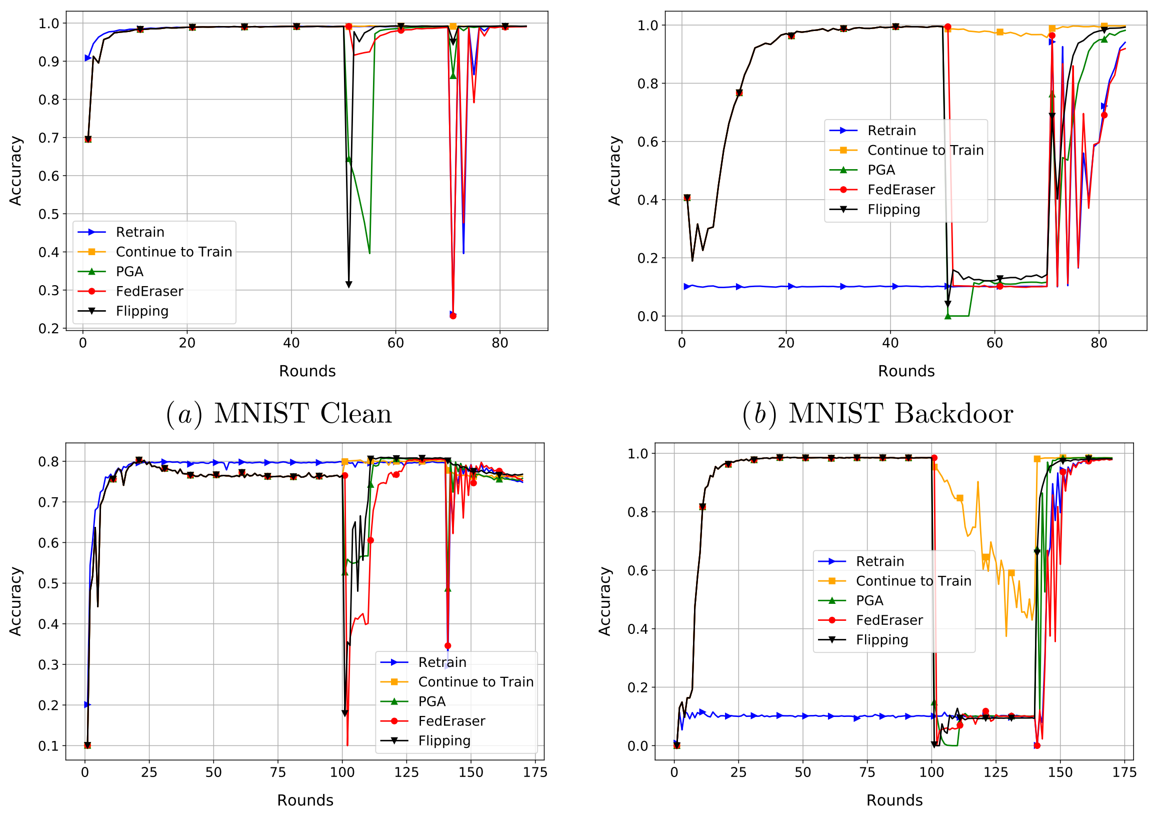
where ρ is the prediction similarity, $\mathcal{D}_i$ $(i = 1, ..., N)$ is batch i of the global test dataset and $\mathcal{M}(\mathcal{D}_i)$ is the prediction of $\mathcal{M}$ on $\mathcal{D}_i$.

4. Experimental Analysis and Discussion

4.1. Effectiveness Analysis

This section provides results and insights from the experimentation to answer our research questions.

RQ1: What are the impacts of different unlearning methods on two datasets, MNIST and CIFAR-10?



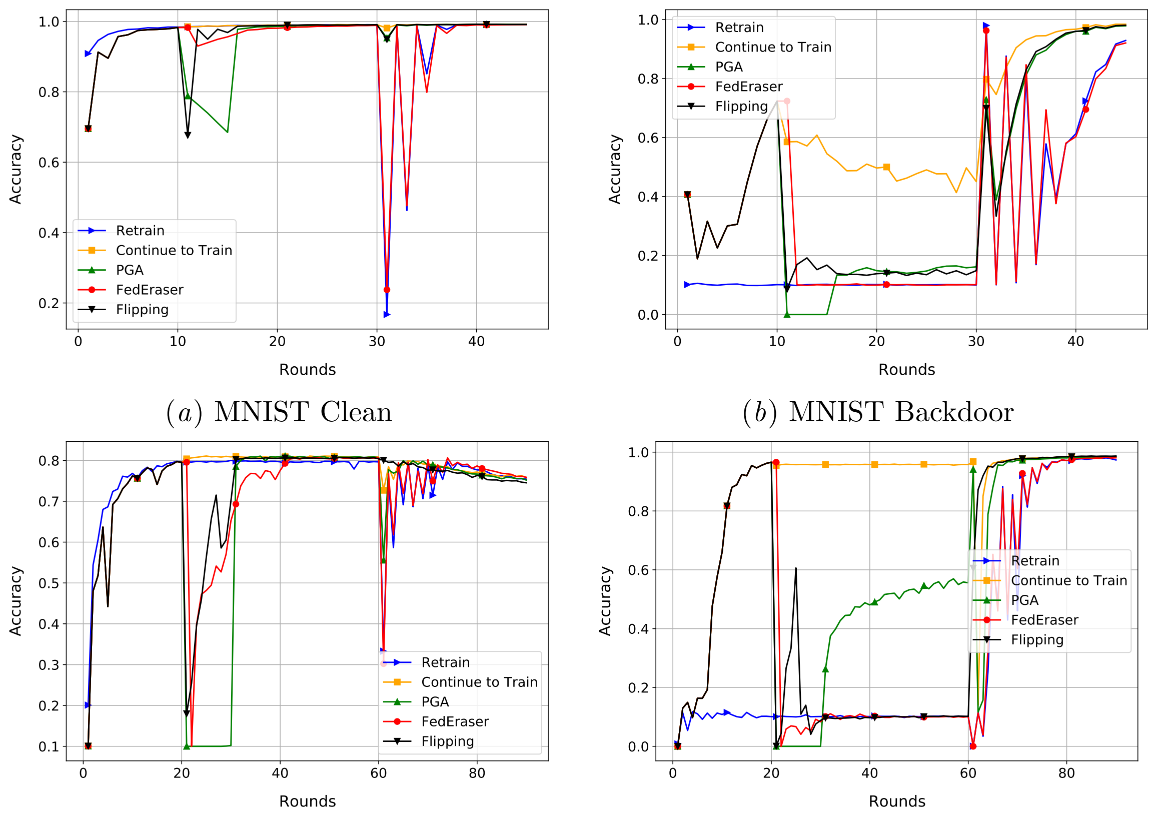
(a) MNIST Clean (b) MNIST Backdoor

(c) CIFAR-10 Clean (d) CIFAR-10 Backdoor

Figure 1: The effectiveness of Continue to Train, PGA, FedEraser, and Flippling unlearning methods compared to the Retrain (baseline) in late unlearning (Scenario 1)

To answer this question, we focus on studying the global model's clean and backdoor accuracy in the unlearning and post-unlearning under the first scenario.

Scenario 1. We observe the same pattern for PGA, FedEraser, and Flipping methods in both datasets, in which the backdoor lines drop significantly in the unlearning phase and



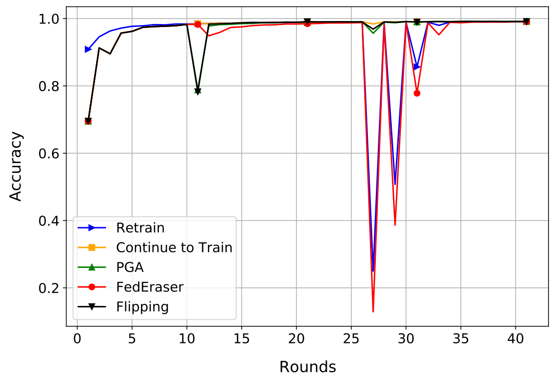
(a) MNIST Clean (b) MNIST Backdoor

(c) CIFAR-10 Clean (d) CIFAR-10 Backdoor

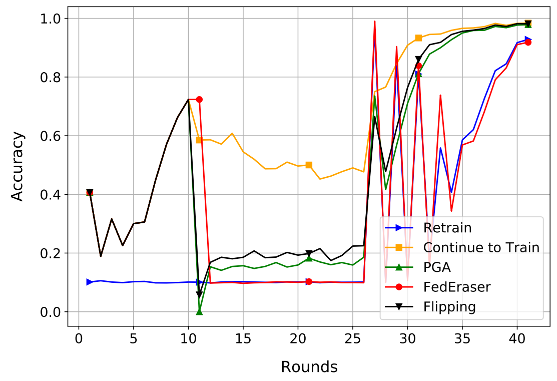
Figure 2: The effectiveness of Continue to Train, PGA, FedEraser and Flippling unlearning methods compared to the Retrain (baseline) in early unlearning (Scenario 2)

remain converged to the baseline in the post-unlearning phase. More importantly, in the unlearning phase, the accuracy of the global model on the clean test set remains high. In other words, the unlearning methods do not affect other clients but those who request the unlearning. This shows the effectiveness of these three unlearning methods in this scenario.

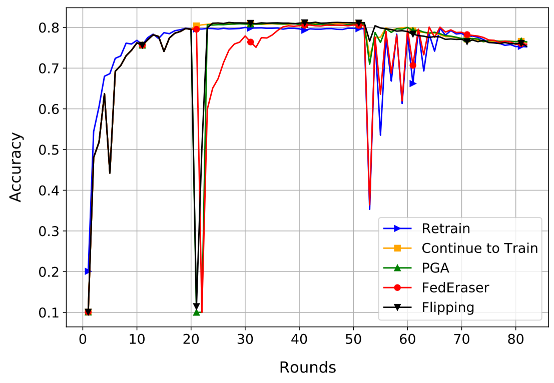
To observe better the impact of each unlearning method on the global model in each phase, in the first case of our experiments, we also highlight in Table 2 the detailed accuracy of the global model on both clean and backdoor test sets in four phases. We observe the change in the accuracy of the global model on a clean test set to verify if the unlearning methods affect other clients' contributions. At the beginning of the post-unlearning phase, the clean accuracies of the unlearned model using both PGA, FedEraser, and Flipping are as high as those before unlearning. This shows that eventually, these methods do not affect the contributions from other clients to the global model. In terms of unlearning effectiveness, both PGA and Flipping methods can unlearn effectively after the first round of unlearning, eliminating the impact of backdoor images on the global model, while for FedEraser, it takes more unlearning rounds to unlearn effectively. However, it is essential to notice that the backdoor accuracy remains high after unlearning with the Continue to Train method.



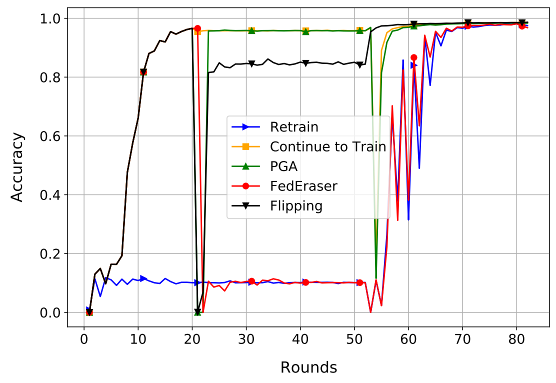
(a) MNIST Clean



(b) MNIST Backdoor



(c) CIFAR-10 Clean



(d) CIFAR-10 Backdoor

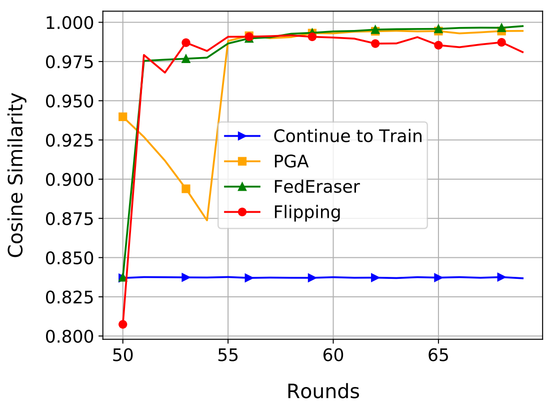
Figure 3: The effectiveness of Continue to Train, PGA, FedEraser and Flippling unlearning methods compared to the Retrain (baseline) in early unlearning with small unlearning rounds (Scenario 3)

Table 2: The effectiveness of four unlearning methods compared to Retrain (baseline).

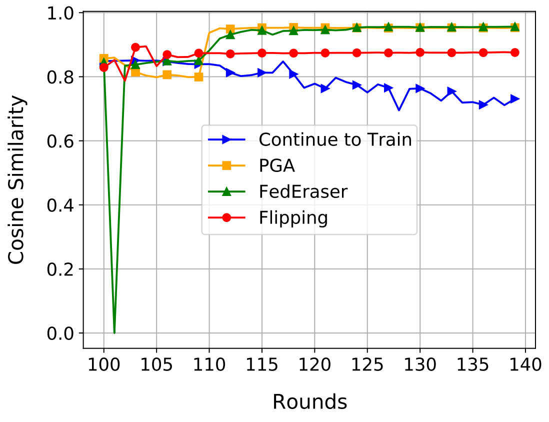
Dataset	Method	BU		SU		FU		SPU		FPU		SR		FR	
		CA	BA	CA	BA	CA	BA	CA	BA	CA	BA	CA	BA	CA	BA
MNIST	Retrain	**0.991**	0.101	0.991	0.102	0.991	0.102	0.991	0.102	0.991	0.101	0.236	0.942	0.991	0.940
	ConTrain	0.991	0.994	0.990	0.986	0.993	**0.979**	0.992	0.977	0.992	0.957	0.991	0.988	0.991	0.997
	PGA	0.991	0.994	0.644	**0.000**	0.395	0.000	**0.971**	0.114	0.990	0.116	0.862	0.762	0.991	**0.981**
	FedEraser	0.991	0.994	0.991	0.994	0.924	0.102	**0.955**	0.101	0.988	0.101	0.231	0.964	0.990	**0.919**
	Flipping	0.991	0.994	0.314	**0.040**	0.981	0.134	**0.990**	0.124	0.992	0.142	0.950	0.686	0.992	**0.992**
CIFAR-10	Retrain	**0.798**	0.101	0.797	0.101	0.796	0.095	0.795	0.097	0.796	0.101	0.295	0.000	0.748	0.979
	ConTrain	0.765	0.984	0.799	0.953	0.799	**0.845**	0.801	0.847	0.800	0.488	0.777	0.981	0.763	0.984
	PGA	0.765	0.984	0.527	**0.148**	0.567	0.000	**0.743**	0.081	0.805	0.100	0.487	0.672	0.758	**0.984**
	FedEraser	0.765	0.984	0.765	0.984	0.401	0.059	**0.605**	0.069	0.803	0.102	0.346	0.000	0.752	**0.978**
	Flipping	0.765	0.984	0.179	**0.003**	0.703	0.127	**0.805**	0.094	0.806	0.093	0.801	0.659	0.768	**0.981**

ConTrain: Continue to Train
CA: Clean Accuracy
BA: Backdoored Accuracy
BU: Before Unlearning
SU: Start Unlearning
FU: Finish Unlearning
SPU: Start Post Unlearning
FPU: Finish Post Unlearning
SR: Start Returning
FR: Finish Returning

This concludes that Continue to Train does not help eliminate the influence of backdoor images on the global model.



(a) MNIST



(b) CIFAR-10

Figure 4: Cosine Similarity between the unlearned and baseline model in Scenario 1

In addition to clean and backdoor accuracy, we use Eq. 4 to measure the Cosine Similarity between two models' predictions on the same backdoor test set. The measurement results are plotted in Fig. 4. In general, we can see there are two main patterns. Using PGA, FedEraser, and Flipping, we observe that after unlearning, the Cosine Similarity between the two models' prediction increases and gradually converges to 1.0. This means that after unlearning using these three methods, the behavior of the unlearned model on the backdoor test set becomes more similar to the behavior of the baseline model. On the other hand, using the Continue to Train, the Cosine Similarity is low and tends to decrease in the later phases. This confirms that the Continue to Train method does not make the unlearned model's behavior similar to the baseline model's.

RQ2: How early and late unlearning affect the general performance of the unlearning methods?

Besides studying different methods under a standard setting (i.e., late unlearning scenario), we evaluate the performance of the selected methods when the unlearned client $\mathcal{C}_0$ requests to unlearn early to answer this research question.

Scenario 2. Firstly, Figs. $2(a)$ and $2(d)$ show that in the unlearning phase, the accuracy of the global model on the clean test set drops significantly. This implies that when client $\mathcal{C}_0$ requests unlearning too early, the unlearning methods can affect other clients' contribution to the global model. However, we can observe that in the post-unlearning phase, the clean lines immediately increase back to before the unlearning phase. Therefore, the unlearning methods generally do not affect other clients. Secondly, Fig. $2(d)$ shows that the PGA method is ineffective on CIFAR-10. This is because in the post-unlearning phase, without the contribution from client $\mathcal{C}_0$, the PGA backdoor line still increases to almost 1.0 instead of converging to the Retrain backdoor line. Meanwhile, in this scenario, FedEraser and Flipping are still effective as they behave similarly to the first scenario.

Scenario 3. In this scenario, our objective is to observe the performance of the studied methods when the number of unlearning rounds is reduced. The behaviors of these methods are mostly aligned with the previous scenario except for PGA and Flipping. In particular, Fig. 3(*d*) shows that after unlearning, without dataset from client $\mathcal{C}_0$, the accuracy of the unlearned model using PGA and Flipping methods on the backdoor test set remains high.

Observing the two scenarios, we conclude that the unlearning moment does affect the performance of the unlearning methods. Particularly, PGA performs more effectively in the late unlearning scenario. On the other hand, FedEraser works effectively in both early and late unlearning scenarios.

RQ3: How effectively does each method facilitate returning clients?

To answer this question, we analyze the performance of four unlearning methods at the *Returning Phase*. As shown in Figs. 1, 2 and 3 that in all scenarios, when client $\mathcal{C}_0$ returns, the backdoor accuracy increases to the same level as before unlearning quickly. This indicates that most unlearning methods can effectively unlearn and enable client $\mathcal{C}_0$ to return to the FL process. However, compared to the baseline method, only PGA and Flipping facilitate rapid returning. On the other hand, as observed during the returning phase, FedEraser requires more training rounds than PGA and Flipping methods to obtain comparable backdoor accuracy.

4.2. Efficiency Analysis

In Table 3, we present the unlearning times for each method and their speedup compared to the baseline. Two separate runs were conducted for each experiment, and the recorded runtime is the lowest among the two runs, although there was minimal variation between them. It is evident that PGA is the fastest method across all scenarios among the four methods. It achieves the highest speedup, ranging from 10.94 times to 54.29 times faster compared to the baseline. Although FedEraser and Flipping demonstrate significant speedup in the first and last scenarios, their efficiency is less pronounced in the second scenario. For instance, with CIFAR-10, FedEraser only accelerates unlearning by 1.10 times. In contrast, unlearning using Flipping is slower than the baseline method when applied to MNIST.

Table 3: Comparative Analysis of Unlearning Time (in seconds) among Five Methods across Three Scenarios

Scenario	1		2		3	
	MNIST	CIFAR-10	MNIST	CIFAR-10	MNIST	CIFAR-10
Retrain	524.78(1.00x)	10892.97(1.00x)	101.88(1.00x)	2185.81(1.00x)	97.77(1.00x)	2166.39(1.00x)
Continue to Train	70.15(7.48x)	1619.28(6.73x)	42.43(2.40x)	1971.36(1.11x)	5.77(16.95x)	316.74(6.84x)
PGA	**9.67(54.29x)**	**330.59(32.95x)**	**9.31(10.94x)**	**164.08(13.32x)**	**4.14(23.59x)**	**66.01(32.82x)**
FedEraser	85.25(6.16x)	1494.40(7.29x)	72.93(1.40x)	1983.95(1.10x)	4.64(21.06x)	119.32(18.16x)
Flipping	131.15(4.00x)	1972.31(5.52x)	**131.76(0.77x)**	1304.93(1.68x)	29.59(3.30x)	264.11(8.20x)

1: Late unlearning 2: Early unlearning 3: Early unlearning with small unlearning rounds

Table 4: Comparison of Unlearning Methods based on Three Criteria

Criterion	Scenario	Retrain (baseline)	Continue to Train	PGA	FedEraser	Flipping
Unlearn effectively	1	✓	×	✓	✓	✓
	2	✓	×	×	✓	✓
	3	✓	×	×	✓	×
Unlearn fast	1	×	✓	✓	✓	✓
	2	×	×	✓	×	×
	3	×	✓	✓	✓	✓
Support fast returning	1	×	✓	✓	×	✓
	2	×	✓	✓	×	✓
	3	×	✓	✓	×	✓

1: Late unlearning 2: Early unlearning 3: Early unlearning with small unlearning rounds

4.3. Discussion

We summarize our result analysis by discussing the comparison of each unlearning method in Table 4 using three different criteria as follows:

- Unlearn effectively: an unlearning method **unlearns effectively** if it can unlearn the global model as effectively as the baseline method.
- Unlearn fast: an unlearning method **unlearns fast** if it can unlearn the global model faster than the baseline method.
- Support fast returning: an unlearning method **supports fast returning** if the backdoor accuracy increases to the same level as before unlearning in fewer rounds compared to the baseline method.

As mentioned, the baseline method does not unlearn fast despite unlearning effectively in all scenarios. The Continue to Train method cannot unlearn effectively in all scenarios. Therefore, we conclude that it cannot be used to unlearn. Next, while PGA is speedy in all scenarios and supports fast returning, it only unlearns effectively in the first scenario. In contrast, although FedEraser can unlearn effectively in all scenarios, it cannot unlearn fast in the second scenario and does not support fast returning. Finally, although Flipping supports fast returning in all scenarios, it does not unlearn fast in the second scenario and does not unlearn effectively in the third scenario. Based on the pros and cons of each method, we suggest that PGA is the most suitable method for the first scenario. However, for the second scenario, instead of using unlearning methods, we should consider training from scratch. And in the last scenario, we should unlearn using FedEraser.

5. Conclusion

This paper comprehensively evaluated four emerging FedUnlearn methods: FedEraser, PGA, Flipping, and Continuing to Train. Our study provides valuable insights into the current state of research and offers directions for future investigations. It is essential to acknowledge the limitations of this study, particularly in terms of the scenarios considered. While the study primarily focuses on significant scenarios in FedUnlearn, it is worth noting that these scenarios may not fully capture the complexity and realism of cross-silo cases. By recognizing these limitations, future research can aim to address and incorporate a broader range of scenarios, e.g., the variation of the total number of clients, to provide a more comprehensive understanding of FedUnlearn and its applications in diverse contexts. Moreover, we plan to incorporate more real-world datasets to evaluate these methods in practice better. Finally, the existing literature lacks an in-depth examination of scenarios involving clients opting out and subsequently rejoining the FL process. Therefore, future research should dedicate more attention to comprehending and tackling the challenges linked to the return of clients who had initially opted out. This is a practical situation that may occur in real-world FL implementations and warrants further investigation.

References

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

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

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. *2015 IEEE Symposium on Security and Privacy*, pages 463–480, 2015.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. Federated unlearning: How to efficiently erase a client in fl? *ArXiv*, abs/2207.05521, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

Yann LeCun, Corinna Cortes, and CJ Burges. The mnist database of handwritten digits. *www.cs.nyu.edu/ ylclab/data/nips2003-raw-digits.ps.gz*, 1998.

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.

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.

H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016.

Stuart L Pardau. The california consumer privacy act: Towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y*, 23:68, 2018.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.

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.

Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation. *ArXiv*, abs/2201.09441, 2022a.

Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441*, 2022b.

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, 2022c.

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.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International conference on learning representations*, 2020.