Subspace based Federated Unlearning

Anonymous authors
Paper under double-blind review

Abstract

Federated learning (FL) enables collaborative machine learning among multiple clients while preserving user data privacy by preventing the exchange of local data. However, when users request to leave the FL system, the trained FL model may still retain information about their contributions. To comply with the right to be forgotten, federated unlearning has been proposed, which aims to remove a designated client's influence from the FL model. Existing federated unlearning methods typically rely on storing historical parameter updates, which may be impractical in resource-constrained FL settings. In this paper, we propose a Subspace-based Federated Unlearning method (SFU) that addresses this challenge without requiring additional storage. SFU updates the model via gradient ascent constrained within a subspace, specifically the orthogonal complement of the gradient descent directions derived from the remaining clients. By projecting the ascending gradient of the target client onto this subspace, SFU can mitigate the contribution of the target client while maintaining model performance on the remaining clients. SFU is communication-efficient, requiring only one round of local training per client to transmit gradient information to the server for model updates. Extensive empirical evaluations on multiple datasets demonstrate that SFU achieves competitive unlearning performance while preserving model utility. Compared to representative baseline methods, SFU consistently shows promising results under various experimental settings.

1 Introduction

The traditional training approach of deep learning typically aggregates data from various participants. However, certain data, such as medical records [30], cannot be relocated from the hospital due to concerns regarding data privacy and individual preferences. In response, Federated Learning (FL) [31; 24; 3; 27] emerges as a prominent decentralized machine learning solution for addressing these challenges. FL facilitates the training of a global model by exchanging model parameters between clients and a central server, effectively bypassing the need to transfer the raw data [22; 23; 27].

Recent privacy legislations [5; 35; 39] provide data owners the right to be forgotten. In the context of machine learning, this right necessitates two key actions: (i) deletion of user data from the storing entity and (ii) removal of the data's influence on the model [15]. Within the realm of Federated Learning (FL), federated unlearning crystallizes the embodiment of the right to be forgotten. However, achieving machine unlearning within the framework of federated learning presents heightened challenges: (1) **Limited Data Access**: In FL, the server lacks direct access to all data and associated operations, rendering forgetfulness techniques reliant on dataset segmentation inapplicable to FL scenarios. (2) **Multi-client Participation**: The initial model of each client in every training round depends on aggregating models from clients engaged in prior-round training, resulting in the gradual propagation of effects from individual data samples across models used for local training at multiple clients [33; 32; 40]. Thus, erasing data samples from one client requires a substantial number of clients to engage in a retraining process.

As mentioned above, retraining in FL demands a significant number of clients to participate in local training, inevitably leading to extended training durations. Some recent endeavors have been focused on addressing this challenge, such as storing the client's historical updated gradient data on the server and utilizing it to revert the trained global model [45; 29]. However, these methods necessitate either the client or the server to retain additional data or gradient information, which may not be practical in FL scenarios with limited storage resources. There are alternative approaches to execute the unlearning process by directly modifying the final model. For instance, directly employing gradient ascent on the target client can achieve the immediate reduction of client data influence in the final model. Nevertheless, this approach may considerably compromise the model's performance.

In this paper, we focus on developing a practical approach for implementing federated unlearning within the final model. We consider unlearning as the inverse process of learning via gradient ascent. However, the loss function is unbounded, requiring constraints on the gradient to preserve model quality [8; 6]. Consequently, we treat the entire process as a constraint-solving problem, aiming to maximize the empirical loss of the target client while maintaining acceptable model performance for other clients. Introducing updates orthogonal to the gradient directions of neural network predictions can induce minimal changes in network output [10]. Building on this idea, we propose a Subspace-based Federated Unlearning method, termed SFU.

In the SFU framework, the server only requires the ascending gradient information from the target client and the descending gradient information from the remaining clients. The server then projects the ascending gradient onto the orthogonal subspace of the descending gradient space. This constrained gradient is used to update the final trained FL global model, aiming to reduce the contribution of the target client while preserving the model's utility. Fig. 1 illustrates the core idea of our approach.

Specifically, participants in the SFU process can be categorized into three roles: the target client to be forgotten, the remaining clients, and the server. The target client performs local gradient ascent based on the global model and sends the gradient to the server. Each remaining client performs gradient descent and transmits their respective gradients to the server. The server aggregates the descending gradients and derives the gradient space through Singular Value Decomposition (SVD) [16; 36]. Finally, the server projects the ascending gradient onto the gradient space and updates the global model. The SFU framework only requires one round of local training per client and a subsequent server update, without the need to access each client's raw data or store historical gradients. Empirical results indicate that SFU achieves competitive unlearning performance across various datasets while maintaining reasonable model accuracy.

In conclusion, our main contributions are as follows:

- We incorporate subspace learning into federated unlearning and propose a novel algorithm named SFU. This algorithm performs gradient ascent within an orthogonal subspace to the gradient space of the remaining clients.
- SFU effectively achieves unlearning while maintaining acceptable model performance, without incurring additional storage costs.
- We conduct comprehensive experiments to evaluate SFU, demonstrating that it achieves competitive performance across multiple datasets, including MNIST, CIFAR10, and CIFAR100.

2 Related Work

Machine unlearning. The concept of "machine unlearning" entails the complete removal of a specific training data instance, necessitating the nullification of its impact on both extracted features and models. The introduction of machine unlearning is attributed to Cao & Yang [7], who reformulate statistical query learning into a summation form and achieve unlearning by selectively updating a subset of the summation. However, this algorithm is confined to transformable traditional machine-learning methods, prompting exploration into machine unlearning for various ML models. Ginart et al. [12] formalize the notion of effective data deletion in machine learning and propose two efficient deletion strategies for the K-means clustering algorithm. In the realm of supervised linear regression, Izzo et al. [17] develop the projective residual update (PRU) for linear and logistic models. While the computational cost of PRU scales linearly with feature dimensions, its suitability is limited for more intricate models such as neural networks. To address the overhead of forgetting, Bourtoule et al. [4] introduce the versatile SISA framework, which trains disjoint sub-models on data shards and only retrains the affected shard after a deletion request. A recent survey [18] provides a comprehensive

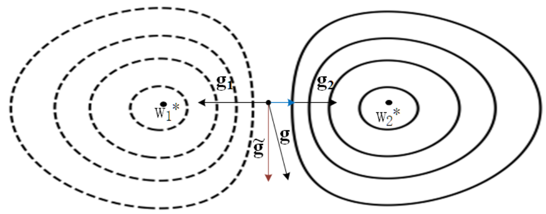


Figure 1: An illustration of SFU with three clients. w_1^* and w_2^* represent the optimal model parameters for client 1 and client 2, respectively. The ascending gradient of the target client is denoted as g. Additionally, g_1 and g_2 correspond to the descending gradients originating from client 1 and client 2. The projection of g onto the orthogonal space spanned by g_1 and g_2 is denoted as $\tilde{g}$. Operating within this orthogonal subspace ensures minimal perturbation to the model's performance on client 1 and client 2.

overview of progress, but most studies still assume centrally accessible data, which is ill-suited for federated settings.

Federated unlearning. Current work can be grouped into two main directions.

Retraining-efficient methods. Liu et al. [29] cache historical gradients to accelerate retraining; Su & Li [41] cluster clients and retrain only within the group containing the target client, which is useful in highly heterogeneous scenarios.

One-shot model modification. Wu et al. [45] subtract stored gradients of the target client from the final model, still requiring extra storage. Wang et al. [44] prune task-specific weights, but only support class-level forgetting. Very recently, NoT [20] introduces a storage-free weight-negation strategy that supports client-, class-, and instance-wise unlearning without retraining. The definition of exact federated unlearning was formalized in FATS [43], which provides a TV-stable FedAvg variant and a provable closed-form unlearning step equivalent to retraining—but it requires server-side batch data and complex aggregation schemes.

Our approach is closely related to UPGA [15] and EWC-SAG [45]. All three perform gradient ascent on the final model to remove a client's influence, but differ in their constraints: UPGA constrains the *magnitude* of the update with an ℓ_2 ball; EWC-SAG employs an elastic weight penalty; in contrast, **SFU** constrains the *direction* of the update to the orthogonal complement of the remaining clients' descent subspace, eliminating the need for historical checkpoints while better preserving accuracy.

3 Method

We introduce a novel federated unlearning approach, termed SFU, outlined in Algorithm 1. SFU leverages constrained gradient information from the target client to adapt the final trained model, effectively removing client contributions while upholding model performance across other clients. Notably, this method eliminates the need for the server to retain a historical record of parameter updates from individual clients and obviates the necessity for extensive retraining.

3.1 Problem Setup

Suppose there are K clients, denoted as $C_1, \ldots, C_N$, respectively. Client C_i possesses a local dataset $\mathcal{D}^i$. The objective of conventional Federated Learning (FL) is to collaboratively train a machine learning model w over the combined dataset $\mathcal{D} \triangleq \bigcup_{i\in[N]} \mathcal{D}^i$:

$$w^* \in \arg\min_{w} L(w) = \sum_{i=1}^{N} \frac{|\mathcal{D}^i|}{|\mathcal{D}|} L_i(w), \tag{1}$$

where $L_i(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}^i}[\ell_i(w;(x,y))]$ represents the empirical loss of client C_i. Throughout the federated training process, each client minimizes their respective empirical risk $L_i(w)$. The model w^* obtained is the final outcome of the FL procedure.

Algorithm 1 Subspace-based Federated Unlearning (SFU)

Input: FL global model w^*, local dataset $\mathcal{D}^i$ of client i, learning rate η.

Target client C_I**:**
$g_I \leftarrow \eta \nabla L_I^{ul}(w^*; \mathcal{D}^I)$
Send g_I to the server
Other clients:
for each client $i \neq I$ **do**
    $g_i \leftarrow \eta \nabla L_i(w^*; \mathcal{D}^i)$
    Send g_i to the server
Server:
for each net's layer $l = 1, 2, 3, \dots$ **do**
    $\boldsymbol{R}^l \leftarrow [\mathbf{g}_1^l, ..., \mathbf{g}_{I-1}^l, \mathbf{g}_{I+1}^l, ..., \mathbf{g}_S^l]$
    Perform SVD on $\boldsymbol{R}^l = \boldsymbol{U}^l \boldsymbol{\Sigma}^l (\boldsymbol{V}^l)^T$
    Select the first k vectors of $\boldsymbol{U}^l$
    $S^l \leftarrow \text{span}\{\boldsymbol{u}_1^l, \boldsymbol{u}_2^l, \dots, \boldsymbol{u}_k^l\}$
    Projection matrix $P^l \leftarrow S^l (S^l)^T$
$P \leftarrow [P^1, P^2, \dots]$
$\tilde{g_I} \leftarrow P g_I$
$g_P \leftarrow (g_I - \tilde{g_I})$
$w^{ul} \leftarrow w^* - g_P$

Now let's delve into the strategy for eliminating the contribution of the target client C_I. An intuitive approach is to escalate the empirical risk $L_I(w)$ associated with the target client C_I, essentially reversing the learning process. However, a straightforward maximization of the loss will impact the model's performance on other clients. Federated unlearning needs to forget the contribution of the target client C_I while ensuring the overall model performance. Consequently, federated unlearning can be formulated as follows:

$$\begin{aligned} &\arg\max_{w} L_I(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}^i}[\ell_I(w; (x,y))] \\ &\text{s.t.} \qquad \mathcal{E}(w) - \mathcal{E}(w^*) \leq \delta \end{aligned} \tag{2}$$

Here, δ represents a tolerable difference in model performance, while $\mathcal{E}(w)$ signifies the accuracy of model w evaluated on the remaining clients within the FL system. Prior work by Halimi et al. [15] utilized the parameter distance between w and w^* as a constraint, although this parameter distance might not fully capture the performance disparity among different models. Conversely, the constraint presented in Eq. 2 effectively addresses this concern.

3.2 Subspace-based Federated Unlearning (SFU)

When applying the ascending gradient update of the target client to the global model without considering other clients, there is a high probability that the neural network will turn into a stochastic model. SFU restricts model updates to orthogonal subspaces aligned with the gradients of other clients to mitigate this issue. This approach achieves the goal of forgetting the target client's contribution while minimizing potential disruptions to model performance for other clients. The training process of SFU is depicted in Fig. 2. Participants in SFU encompass the target client, remaining clients, and the server. The target client employs gradient ascent and transmits the resulting gradient to the server. Other clients engage in gradient descent and forward their descending gradients to the server. The server calculates the gradient space of the other clients and performs unlearning updates on the global model. Subsequently, we will provide a more detailed explanation of the SFU process.

3.2.1 Local training on clients

Satisfying the constraints outlined in Eq.(2) necessitates imposing restrictions on the ascending gradient. Drawing inspiration from Fig. 1, it becomes evident that moving orthogonally to the gradient space yields the least impact (or even negligible change locally) on the FL model's performance for clients. This valuable insight prompts us to project the updated gradient of the target client C_I onto the orthogonal space of the gradient subspace associated with other clients [10]. This process requires both the ascending gradient from the target client and the descending gradients from the other clients. Hence, upon receiving a forget request from the target client, the training procedures for the target client and the other clients are as follows:

- **Target Client:** The target client trains the FL model using the gradient ascent algorithm. The empirical loss of client C_i in FL is denoted as $L_i(w)$. To implement gradient ascent, we modify the empirical loss of the target client C_I to $\frac{1}{L_I(w)}$, denoted as $L_I^{ul}(w)$. Consequently, the local optimization objective for the target client becomes $\arg\min_{L_I^{ul}}(w)$. Upon completing local training, the target client transmits the gradient information $\nabla L_I^{ul}(w)$ to the server.
- **Other Clients:** The remaining clients with availability undergo the same gradient descent training as in the FL process. Hence, the local optimization objective for each remaining client remains $L_i(w)$, and they also forward the gradient information $\nabla L_i(w)$ to the server upon training completion.

3.2.2 Computation of projection matrix

When the server aggregates the gradient information sent by each client, it proceeds to compute the projection matrix of descending gradient space for each layer of the model. This process involves two key steps:

- The gradient information for each client in every layer of the network can be represented as a matrix. Let $\mathbf{g}_i^l \in \mathbb{R}^{m\times n}$ denote the gradient matrix for layer l of client C_i. To calculate the gradient space, the server needs to merge the gradient information from different clients. In SFU, we concatenate the gradient matrices from available clients other than the target client C_I and assume that the number of available clients is S, resulting in the merged matrix $\boldsymbol{R}^l = [\mathbf{g}_1^l, ..., \mathbf{g}_{I-1}^l, \mathbf{g}_{I+1}^l, ..., \mathbf{g}_S^l]$, where $\boldsymbol{R}^l \in \mathbb{R}^{m\times(n\times S)}$.
- Next, we perform SVD on $\boldsymbol{R}^l$, yielding $\boldsymbol{R}^l = \boldsymbol{U}^l\boldsymbol{\Sigma}^l(\boldsymbol{V}^l)^T$. Subsequently, we obtain the k-rank approximation $(\boldsymbol{R}^l)_t$ of $\boldsymbol{R}^l$ based on a given coefficient ϵ^l, satisfying the following criterion:

$$||(\boldsymbol{R}^l)_k||_F^2 \geq \epsilon^l||\boldsymbol{R}^l||_F^2. \tag{3}$$

  We then span the first k vectors in $\boldsymbol{U}^l$ to represent the significant space at layer l, resulting in $S^l = span\{\boldsymbol{u}_1^l, \boldsymbol{u}_2^l, ..., \boldsymbol{u}_k^l\}$. This space encompasses all directions associated with the highest singular values in the representation [37].

Following these two steps, the server obtains the projection matrix P^l for layer l. Specifically, $P^l = S^l(S^l)^T$, and this process is repeated for each layer, leading to $P = [P^1, P^2, \ldots]$.

3.2.3 Update of the global model on the server

After receiving gradient information from clients and calculating the projection matrix P, the server proceeds to update the global model w^*. Initially, the server projects the gradient g_I from the target client onto the gradient space, resulting in the projection $\tilde{g_I}$, where $\tilde{g_I} = Pg_I$. Subsequently, the server updates the global model w^* in the orthogonal subspace, as represented by the following equation:

$$w^{ul} = w^* - g_P. \tag{4}$$

Where $g_P = (g_I - \tilde{g_I})$, representing the projection of g_I onto the subspace. This updated model w^{ul} removes the contribution of the target client C_I while maintaining a performance level similar to that of the original global model.

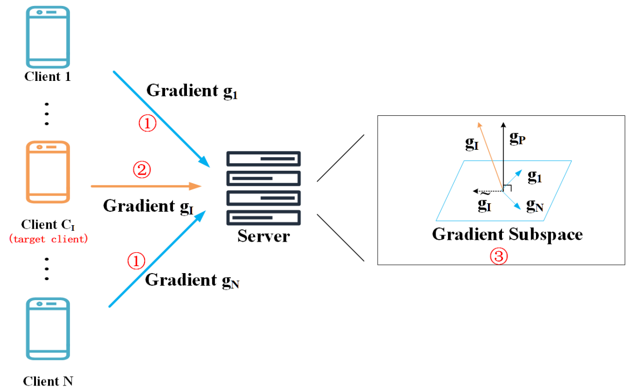


Figure 2: The pipeline of SFU. The entire process occurs subsequent to the training of the FL model. The orange client denotes the target client from which the contribution is to be removed; the blue clients represent others. The boxes on the right of the image symbolize global model updates performed on the server.

In conclusion, we offer several remarks regarding the proposed SFU algorithm. It is noteworthy that subspace learning has found widespread application in diverse domains such as continual learning [37], meta learning [19], one-shot learning [42], adversarial training [26], Graph forgetting [9], and expedited training of deep learning models [25]. However, SFU is the first work to use the orthogonal space of input gradient space for federated unlearning.

3.3 On the effectiveness of SFU

In this section, we investigate the efficacy of SFU by quantifying the difference between our unlearned solution $\mathbf{g}_{SFU}$ and the solution $\mathbf{w}_r$ obtained through retraining. We anticipate that $\|\mathbf{w}_p - \mathbf{w}_u\|_2$ remains bounded to mitigate the generation of arbitrary models. Prior to presenting our results, we establish the following customary assumptions regarding the variance of stochastic gradients, as depicted in Assumption 1. It is important to note that Assumption 1 represents standard assumptions in the theoretical analysis of Federated Learning (FL) [49; 28].

Assumption 1. *The expected squared norm of stochastic gradients of $L(w)$ is uniformly bounded by G^2, i.e., $\mathbb{E}\left\|\nabla L(w)\right\|^2 \leq G^2$.*

Theorem 1. *Let us suppose Assumptions1 hold. Let us define $\mathbf{w}_{ul}$ as the solution obtained by SFU, $\mathbf{w}_r$ is the solution obtained by re-training from scratch. Then, the closeness of $\mathbf{w}_{ul}$ to the weight parameters $\mathbf{w}_r$ can be bounded by*

$$\mathbb{E}[\|\mathbf{w}_r - \mathbf{w}_{ul}\|_2] \leq (2\eta T + 1)G^2, \tag{5}$$

This indicates that the difference between the last computed model by SFU and the retrained model is limited.

Proof. Let us define $\mathbf{w}(t)$, $\mathbf{w}_r(t)$ as the weight parameters obtained by training from scratch with target client and without target client for t epochs with the same initialization $\mathbf{w}(0) = \mathbf{w}_r(0)$, and $\mathbf{w}_{ul}(t)$ denote the weight parameters obtained by applying SFU on $\mathbf{w}(t)$. To help the reader better understand the proof strategy used in this section, we derive the upper bound of $\|\mathbf{w}_r(T) - \mathbf{w}_{ul}\|_2$ by first expanding the formula into two terms

$$\begin{aligned} &\|\mathbf{w}_r(T) - \mathbf{w}_{ul}\|_2 \\ \leq &\|\mathbf{w}_r(T) - \mathbf{w}(T)\|_2 + \|\mathbf{w}(T) - \mathbf{w}_{ul}\|_2. \end{aligned} \tag{6}$$

Then, we derive the upper bound of the first term and the second term respectively to obtain the upper bound of $\|\mathbf{w}_r(T) - \mathbf{w}_{ul}\|_2$.

According to the gradinet update rule in gradient descent, we can bound the change of model parameters in the first term $\mathbf{w}_r(t)$ and $\mathbf{w}(t)$ by

$$\begin{aligned}
&\|\mathbf{w}_r(t+1) - \mathbf{w}(t+1)\|_2 \\
&= \|\mathbf{w}_r(t) - \eta\nabla L^r(\mathbf{w}_r(t)) - \mathbf{w}(t) + \eta\nabla L(\mathbf{w}(t))\|_2 \\
&\leq \|\mathbf{w}_r(t) - \mathbf{w}(t)\|_2 + \eta\|\nabla L^u(\mathbf{w}_r(t))\|_2 + \eta\|\nabla L(\mathbf{w}(t))\|_2 \\
&\leq \|\mathbf{w}_r(t) - \mathbf{w}(t)\|_2 + 2\eta G^2,
\end{aligned} \tag{7}$$

where $L^r(L) = \sum_{i \in N/I} \frac{|\mathcal{D}^i|}{|\mathcal{D}|} L_i(w)$. Then after T iterations, we can bound the different between two parameters as

$$\|\mathbf{w}_r(T) - \mathbf{w}(T)\|_2 \leq 2\eta T G^2. \tag{8}$$

P is the projection matrix, so $\|P\|_2 \leq 1$. We can bound the change of model parameters in the second term $\mathbf{w}_r(t)$ and $\mathbf{w}_{ul}$ by

$$\begin{aligned}
&\|\mathbf{w}(T) - \mathbf{w}_{ul}\|_2 \\
&\quad= \|\mathbf{w}(T) - (\mathbf{w}(T) - P\nabla L(w(T)))\|_2 \\
&\quad= \|P\nabla L(w(T))\|_2 \\
&\quad\leq G^2,
\end{aligned} \tag{9}$$

Adding the bound of the first term and the bound of the second term we can obtain the final bound $(2\eta T + 1)G^2$. □

In the following theorem, we show that the new direction $-g_P$ is still a descent direction (*i.e.* $\langle -g_P, g_I \rangle \leq 0$).

Theorem 2. *Let g_I be the ascent gradient of target client C_I and P is the projection matrix to the gradient space of other clients. Let $\tilde{g} = Pg_I$, $g_P = (g_I - \tilde{g_I})$. Then, $-g_P$ is also a descent direction for $L(w)$.*

Proof. For a vector u to be a descent direction it should satisfy $\langle u, g \rangle \leq 0$. To begin with, we have

$$\begin{aligned}
\langle -g_P,\ g_I \rangle &= \langle -g_P, g_P + \tilde{g_I} \rangle \\
&= -\|\ g_P\|^2 - \langle -g_P, \tilde{g_I} \rangle\ .
\end{aligned} \tag{10}$$

Since g_P is orthogonal to the space in which $\tilde{g_I}$ lies, hence $\langle g_P, \tilde{g_I} \rangle = 0$. Substituting this into Eq. 10, we have $\langle -g_P,\ g_I \rangle = -\|\ g_P\|^2 \leq 0$. Therefore, $-g_P$ is a descent direction for client C_I. □

The above two theories ensure that SFU indeed updates the model in the direction of removing the target client contribution. They also provide an explanation for the limited difference between the model resulting from SFU and the model obtained through complete retraining. This guarantees a certain level of performance for the model.

4 Experiments

In this section, we empirically assess the effectiveness of SFU across various model architectures on three datasets. Our experimental findings demonstrate that our unlearning strategies can adeptly eliminate the target client's impact on the global model while incurring minimal performance degradation. Furthermore, these strategies facilitate rapid accuracy recovery within a few training rounds. We commence by outlining the experimental setup and subsequently unveil the evaluation outcomes. For comprehensive experimental settings, please consult the Appendix.

4.1 Experimental Setup

Datasets. We evaluate the performance of SFU using three widely recognized datasets: MNIST [47], CIFAR10, and CIFAR100 [21]. The datasets exhibit increasing levels of training difficulty from MNIST to

			FedAvg		Retraining		UPGA		ULS		EWC-SAG		SFU	
Dataset	distribution	network	test acc	atk acc	test acc	atk acc	test acc	atk acc	test acc	atk acc	test acc	atk acc	test acc	atk acc
MNIST	IID	MLP	96.57	99.14	96.45	0.611	86.48	0.10	83.42	0.0	90.19	0.24	**92.8**	0.15
		CNN	99.15	99.63	99.21	0.20	51.59	0.0	34.95	0.0	88.9	1.56	**98.62**	0.06
	Dir(0.6)	MLP	96.03	99.04	95.95	0.60	81.91	0.14	68.41	0.0	86.69	1.41	**89.12**	0.25
		CNN	98.96	99.58	99.01	0.26	14.11	0.0	46.43	0.0	85.25	0.0	**98.76**	0.01
	Dir(0.3)	MLP	95.54	98.95	95.48	0.66	83.21	0.24	72.96	0.0	86.34	1.88	**88.63**	0.26
		CNN	98.92	99.78	98.98	0.28	52.41	0.0	38.76	0.0	86.83	0.0	**98.39**	0.03
CIFAR10	IID	MLP	50.86	52.38	50.6	3.77	11.05	0.01	13.75	0.0	20.45	0.08	**44.76**	0.96
		CNN	66.40	37.14	67.56	4.42	60.94	9.4	10.0	0.0	61.48	1.02	**61.68**	0.46
	Dir(0.6)	MLP	47.58	53.58	48.42	4.81	16.54	0.02	11.97	0.0	33.4	0.12	**42.95**	3.67
		CNN	63.74	39.88	65.83	3.21	64.2	8.34	53.27	0.85	56.27	0.94	**61.52**	6.43
	Dir(0.3)	MLP	46.41	54.4	45.51	6.01	13.69	0.01	10.36	0.0	30.73	0.26	**45.21**	1.53
		CNN	61.52	51.84	63.51	4.54	38.76	0.08	10.06	0.0	50.07	1.43	**54.67**	3.72
CIFAR100	IID	MLP	21.93	57.79	23.42	1.73	9.05	0.01	2.03	0.0	20.01	0.26	**22.06**	1.46
		CNN	33.20	4.66	33.68	0.48	34.35	0.13	1.75	0.0	31.50	0.20	**32.45**	1.27
	Dir(0.6)	MLP	21.12	50.89	21.8	0.78	9.72	0.0	1.59	0.0	19.12	0.50	**20.12**	0.71
		CNN	31.26	7.77	32.06	0.58	29.38	0.02	28.52	0.03	30.89	0.09	**31.55**	0.12
	Dir(0.3)	MLP	20.17	57.93	20.67	1.30	6.94	0.01	1.06	0.0	16.52	0.61	**19.2**	1.34
		CNN	31.61	5.81	32.02	0.65	31.96	0.18	1.0	0.0	30.86	0.20	**31.12**	2.10

Table 1: Results of different unlearning methods. We record the attack success rate as "atk acc," and "test acc" represents the accuracy metric on the clean test data.

CIFAR100. Our evaluation encompasses two distinct data distribution scenarios: Independent and Identically Distributed (IID), as well as Non-IID (Non Independent and Identically Distributed). For the Non-IID setting, we adopt the Dirichlet distribution (β): The label distribution on each device follows the Dirichlet distribution, where β serves as the concentration parameter ($\beta > 0$).

Baselines. Our goal is to achieve the unlearning process by adapting the FL model. To this end, we select three prominent federated unlearning algorithms that *directly modify the final FL model* as reference benchmarks: Unlearning via Projected Gradient Ascent (UPGA) [15], UL-Subtraction (ULS) [45], and EWC-SAG[1] [15]. Furthermore, we include a full *retraining from scratch* baseline for comparative evaluation. These methods are chosen because they (i) require no multi-epoch retraining of all clients, (ii) do not assume access to auxiliary server-side data, and (iii) represent three different constraint strategies—ℓ_2-ball, gradient subtraction, and elastic weight regularisation—thus providing a diverse yet fair yardstick for SFU.

Unlearning Assessment Approaches. We employ backdoor attacks during the target client's updates to the global model, enabling us to intuitively examine the unlearning impact through the success rate of a model backdoor attack on the unlearned global model. An effective unlearning method should diminish the success rate of such an attack post unlearning. It is worth noting that due to model prediction errors, even retraining can yield an attack success rate greater than 0. In our experiments, we follow to the approach of Halimi et al. [15] and execute the backdoor attack using a "pixel pattern" trigger of size 2×2, altering the label to "9" on data with labels other than "9". Furthermore, we evaluate the model's performance after unlearning using accuracy metrics on untainted test data.

Implementation details. We employed two network architectures, MLP and CNN, to handle the MNIST and CIFAR10 datasets. These models were implemented using PyTorch [34]. Our experimental setup consists of 10 clients, with one designated as the target client. All clients fully participate in each training round. Unlearning experiments were carried out on the FL model after 100 training rounds. We backdoor 80% of the data on the target client.The hyperparameter settings for each approach were as follows: In the case of SFU, the learning rate was set to 0.01, the number of epochs was set to 1, and a mini-batch size of 128 was utilized for gradient ascent on the target client. We explored the SVD parameter ϵ within the range $[0.90 - 0.99]$ and selected the optimal value. SFU constructs the gradient space by aggregating gradient information from all other clients.For UPGA and EWC-SAG, the learning rate, the number of epochs, and the mini-batch size were maintained consistent with those of SFU. Additionally, we performed a parameter search to determine the specific values of their unique parameters.

4.2 Main Results

Efficient Unlearning with Minimal Performance Loss. The unlearning effects of SFU and other baseline methods on various IID datasets and model architectures are presented in Table 1. It is evident

[1] We follow the notation of Halimi et al. [15] and denote *elastic weight consolidation with single ascent gradient* as EWC-SAG.

that SFU, similar to retraining, effectively mitigates the backdoor attack success rate across all datasets. Other baselines also yield comparable results, indicating their ability to efficiently remove the influence of the target client.Furthermore, Table 1 also provides accuracy results for each baseline on the clean test dataset after unlearning. The outcomes reveal that several alternative methods lead to significant performance degradation while accomplishing forgetting. For instance, UPGA results in an 47% reduction in accuracy for the CNN model on the IID MNIST dataset, whereas SFU demonstrates a mere 1% reduction. This contrast underscores SFU's capability to maintain model performance while eliminating the contribution of the target client.

Robustness Across Different Data Settings. The heterogeneity of data across different clients and the complexity of training tasks can both impact the federated learning process. Table 1 illustrates the influence of these factors on the unlearning process. It is evident that as the data heterogeneity increases, various unlearning methods lead to more significant performance degradation. For instance, during the transition from IID data to Dir(0.3) data, SFU's performance degradation on the CNN model trained on the CIFAR10 dataset increases from 4.72% to 6.85%. However, this performance change remains smaller than the performance losses observed in other algorithms when facing changes in data heterogeneity. Furthermore, as the task complexity shifts from MNIST to CIFAR100, some unlearning methods may become ineffective. For instance, ULS generates a random CNN model on the CIFAR10 dataset, while SFU manages to maintain the model's performance after unlearning. This highlights SFU's superior robustness across different data settings.

Recovery of Model Accuracy After Unlearning After undergoing different Federated Unlearning methods, it is a common practice to fine-tune the unlearned model on the remaining clients to restore its accuracy. We conducted fine-tuning on the CNN models generated by these unlearning methods on the remaining clients using the IID CIFAR10 dataset. In contrast, the ULS method employs clean server data for knowledge distillation-based fine-tuning. Fig. 3 presents the results on the CIFAR10 dataset under the IID setting. Impressively, SFU achieves higher accuracy with just one or a few rounds of retraining, while other methods require more training rounds. This emphasizes that SFU can offer a superior initial model for accuracy recovery.

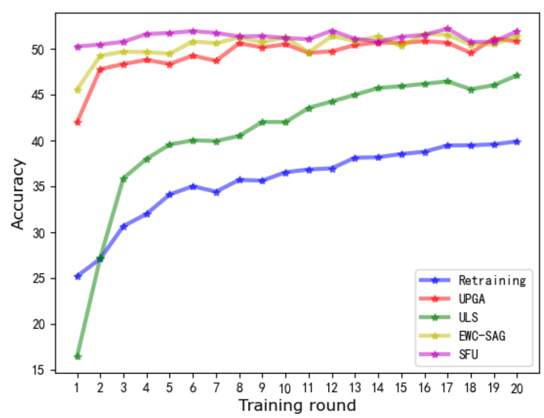
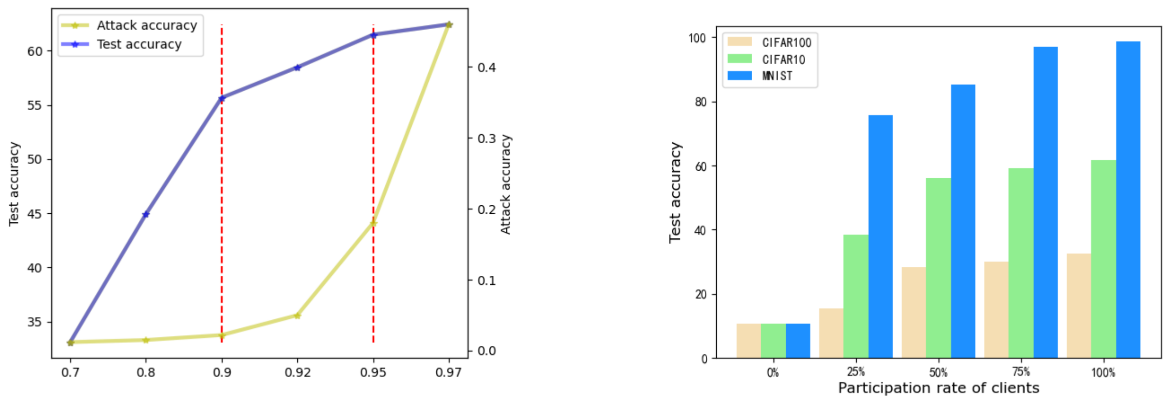


Figure 3: Convergence plots for SFU and other baselines on CIFAR-10 (CNN, IID).

4.3 Exploratory Study

Impact of different SVD coefficient ϵ. As discussed in Section 3.2, our approach aims to mitigate the decline in accuracy by taking gradient steps orthogonal to the gradient space of the remaining clients. This gradient space is formed using bases that approximate the significant task representations of the remaining clients. The extent of this approximation is controlled by the threshold hyperparameter ϵ^l at layer l of the network, defined in Equation 3. In our experiments, we maintain the same parameter ϵ for all layers. The impact of different ϵ values is visually demonstrated in Fig. 4(a). A lower ϵ value (closer to 0) empowers the optimizer to adjust the model's weights mainly along directions where the gradients from other clients exhibit higher representational significance. Consequently, this can lead to substantial changes in the model's performance across other clients. Conversely, a higher ϵ value (closer to 1) preserves these correlations, but it may not fully eliminate the target client's contribution due to the constraints imposed by the abundant gradients in the space. Furthermore, Fig. 4(a) indicates that selecting a value for ϵ between 0.9 and 0.95 in our federated unlearning algorithm strikes a balance between eliminating the target client's contribution and preserving model performance as effectively as possible.

Impact of client's participation rate. In a real-world FL scenario, not all clients can participate simultaneously in the federated training process. SFU relies on clients providing descending gra-



(a) SFU results on CIFAR-10 with varying ϵ values.

(b) Accuracy under IID setting vs. client participation rate.

Figure 4: Ablation results of SFU.

dients to establish the gradient space. We investigated the impact of removing the target client's contribution when employing SFU with varying proportions of available clients. The experimental results, illustrated in Fig. 4(b), reveal that SFU effectively eliminates the target client's contribution across nearly all client participation rates. Moreover, as the client participation rate increases, the performance of the unlearning model on the remaining clients improves.Interestingly, at a client participation rate of 0, which is equivalent to directly updating the global model using the target client's descending gradient, the outcome is a random model with a 10% accuracy across ten classification tasks. This underscores the significance of conducting SFU within an orthogonal gradient space, emphasizing its role in achieving meaningful updates even when only a subset of clients is available for participation.

Impact of the amount of data on the target client. The data volume of the target client indicates to some extent the importance of the target client in the overall FL system. As shown in Fig. 5 , as the amount of data in the target client increases, the performance of the model after performing SFU will produce degradation in the remaining clients. However, the SFU can successfully remove its contribution to the FL system under almost all data volume conditions of the target client.

Figure 5: Outcome under the IID setting for different amounts of data on the target client.

5 Conclusion

In this paper, we propose a novel federated unlearning approach that can successfully eliminate the contribution of a specified client to the global model, which also can minimize the model accuracy loss by performing a gradient ascent process within the subspace at any stage of model training. Our approach only relies on the target client to be forgotten from the federation without the server or any other client keeping track of its history of parameter updates. We have used a backdoor attack to effectively evaluate the performance of the proposed method and our experimental results demonstrate the efficiency and effectiveness of SFU.

References

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

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

[3] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[4] 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 (SP)*, pp. 141–159. IEEE, 2021.

[5] Axel Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer, 2017.

[6] 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 (SP)*, pp. 1366–1383. IEEE, 2023.

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

[8] Yudong Chen and Martin J Wainwright. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*, 2015.

[9] Weilin Cong and Mehrdad Mahdavi. Efficiently forgetting what you have learned in graph representation learning via projection. In *International Conference on Artificial Intelligence and Statistics*, pp. 6674–6703. PMLR, 2023.

[10] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3762–3773. PMLR, 2020.

[11] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *RAID*, pp. 301–316, 2020.

[12] 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.

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

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

[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.

[16] Andreas Hoecker and Vakhtang Kartvelishvili. Svd approach to data unfolding. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 372(3):469–481, 1996.

[17] 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*, pp. 2008–2016. PMLR, 2021.

[18] Hyejun Jeong, Shiqing Ma, and Amir Houmansadr. Sok: Challenges and opportunities in federated unlearning. *arXiv preprint arXiv:2403.02437*, 2024.

[19] Weisen Jiang, James Kwok, and Yu Zhang. Subspace learning for effective meta-learning. In *International Conference on Machine Learning*, pp. 10177–10194. PMLR, 2022.

[20] Yasser H Khalil, Leo Brunswic, Soufiane Lamghari, Xu Li, Mahdi Beitollahi, and Xi Chen. Not: Federated unlearning via weight negation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25759–25769, 2025.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Guanghao Li, Yue Hu, Miao Zhang, Ji Liu, Quanjun Yin, Yong Peng, and Dejing Dou. Fedhisyn: A hierarchical synchronous federated learning framework for resource and data heterogeneity. In *Proceedings of the 51st International Conference on Parallel Processing*, pp. 1–11, 2022.

[23] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[24] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021.

[25] Tao Li, Lei Tan, Zhehao Huang, Qinghua Tao, Yipeng Liu, and Xiaolin Huang. Low dimensional trajectory hypothesis is true: Dnns can be trained in tiny subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[26] Tao Li, Yingwen Wu, Sizhe Chen, Kun Fang, and Xiaolin Huang. Subspace adversarial training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13409–13418, 2022.

[27] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[28] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[29] 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)*, pp. 1–10. IEEE, 2021.

[30] Yun Liu, Krishna Gadepalli, Mohammad Norouzi, George E Dahl, Timo Kohlberger, Aleksey Boyko, Subhashini Venugopalan, Aleksei Timofeev, Philip Q Nelson, Greg S Corrado, et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442*, 2017.

[31] 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*, pp. 1273–1282. PMLR, 2017.

[32] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pp. 691–706. IEEE, 2019.

[33] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1–15, 2018.

[34] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[35] 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.

[36] Rowayda A Sadek. Svd based image processing applications: state of the art, contributions and research challenges. *arXiv preprint arXiv:1211.7102*, 2012.

[37] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*, 2021.

[38] 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.

[39] Supreeth Shastri, Melissa Wasserman, and Vijay Chidambaram. The seven sins of {Personal-Data} processing systems under {GDPR}. In *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, 2019.

[40] Mengkai Song, Zhibo Wang, Zhifei Zhang, Yang Song, Qian Wang, Ju Ren, and Hairong Qi. Analyzing user-level privacy attack against federated learning. *IEEE Journal on Selected Areas in Communications*, 38(10):2430–2444, 2020.

[41] Ningxin Su and Baochun Li. Asynchronous federated unlearning. In *IEEE INFOCOM 2023-IEEE conference on computer communications*, pp. 1–10. IEEE, 2023.

[42] Shangchao Su, Bin Li, and Xiangyang Xue. One-shot federated learning without server-side training. *Neural Networks*, 2023.

[43] Youming Tao, Cheng-Long Wang, Miao Pan, Dongxiao Yu, Xiuzhen Cheng, and Di Wang. Communication efficient and provable federated unlearning. *arXiv preprint arXiv:2401.11018*, 2024.

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

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

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

[47] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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

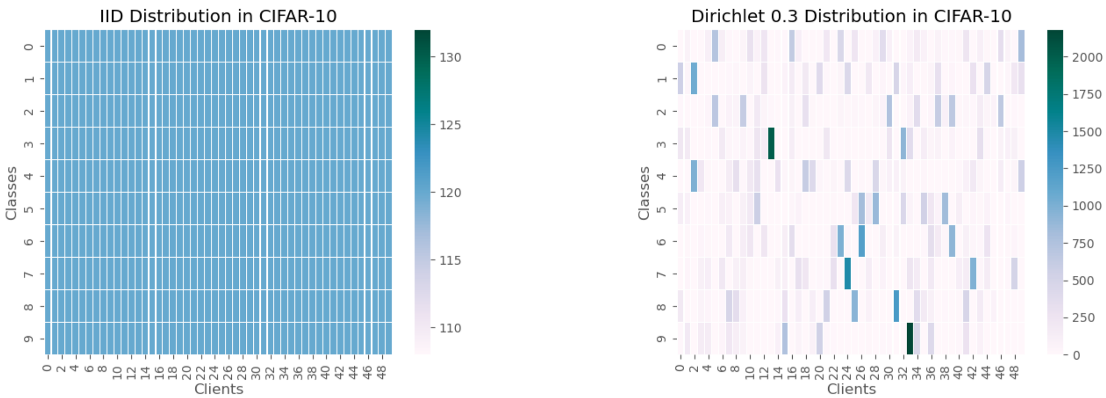
[49] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

We conducted experiments on real-world datasets including MNIST, CIFAR10, and CIFAR100. The experimental settings are described in detail below.

Dataset. We utilized real-world datasets, specifically MNIST, CIFAR10, and CIFAR100. The MNIST dataset [47] consists of 60,000 training samples and 10,000 test samples distributed across 10 classes. Each data sample is a grayscale image of dimensions 28×28. The CIFAR10 dataset comprises 50,000 training samples and 10,000 test samples spread across 10 classes. Each data sample is a color image with dimensions $3 \times 32 \times 32$. Similarly, the CIFAR100 dataset [21] includes 50,000 training samples and 10,000 test samples distributed among 100 classes, with 500 training samples per class. The normalization of pixel values for MNIST and CIFAR10/100 is performed using mean and standard deviation values of [0.5, 0.5, 0.5] for both.

Datasets	Training Data	Test Data	Classes	Size
MNIST	60,000	10,000	10	28×28
CIFAR-10	50,000	10,000	10	$3 \times 32 \times 32$
CIFAR-100	50,000	10,000	100	$3 \times 32 \times 32$

Table 2: Summary of dataset characteristics



(a) IID (b) Dir(0.3)

Figure 6: Heat maps for each client using CIFAR10 dataset under different data partitions. Color bar indicates the number of data samples, and each rectangle represents the count of data samples per class in a partition.

Dataset Partitions. To ensure a fair comparison with other baselines, we introduce heterogeneity by partitioning the total dataset based on label ratios sampled from the Dirichlet distribution. A parameter controls the level of heterogeneity in the data partition. Fig. 6 displays heat maps illustrating the label distribution across different datasets. Notably, for a heterogeneity weight of 0.3 in the Dirichlet distribution, approximately 10% to 20% of categories dominate each client, as depicted by the blue blocks in Fig. 6. In contrast, the IID dataset is uniformly distributed across clients, as indicated by the blue block in Fig. 6.

Baseline Algorithms. We evaluate our proposed federated unlearning algorithm (SFU) against four baseline methods:

- **Retraining**: This method involves retraining the entire FL system without excluding the target client, resulting in computational and communication overhead.
- **UL-Subtraction** [45]: This approach forgets the target client by subtracting historical parameter updates unique to the target client from the global model. Knowledge distillation is used to alleviate the distortion caused by subtraction.
- **Unlearning via Projected Gradient Ascent (UPGA)** [15]: UPGA leverages gradient ascent information from the target client to revert the learning process and achieve unlearning, while constraining updates to an ℓ_2-norm ball.
- **EWC-SGA** [15]: EWC-SGA employs the Fisher Information matrix to regularize the cross-entropy loss and control parameter updates, with higher importance factors imposing stricter constraints.

Network Architectures. We employ two neural network architectures in our experiments:

- **MLP:** A fully-connected neural network with 2 hidden layers, containing 200 and 100 neurons, respectively.

- **CNN:** A network architecture consisting of 2 convolutional layers with 64 5×5 filters, followed by 2 fully connected layers with 800 and 500 neurons and a ReLU activation function.

Both MLP and CNN models were implemented using PyTorch [34].

Implementation Details. For the unlearning experiment, we conducted trials on the FL model after 100 training rounds. The hyperparameters for each method were set as follows: SFU used a learning rate of 0.01, epoch as 1, and a mini-batch size of 128 for gradient ascent on the target client. SVD parameters followed the setting of Saha et al. [37], exploring ϵ within the range $[0.90 - 0.99]$ and selecting the optimal value. UPGA and EWC-SAG maintained the same learning rate and mini-batch size as SFU. ULS on the server used a public dataset formed by randomly sampling one-tenth of the total data.

For the model performance recovery experiment, FL training commenced with the stochastic model for full retraining. For SFU, UPGA, and EWC-SAG, FL training started on the unlearned local model without the target client's involvement. Knowledge distillation with the server's public data aided model accuracy recovery for UL. Additionally, parameter searches were performed to determine the best hyperparameter values.

Unlearning Metrics Comparing the distinction between the unlearned model and the retrained model serves as a fundamental criterion for measuring the effectiveness of unlearning. Common dissimilarity metrics encompass model test accuracy difference [4], ℓ_2-distance [46], and Kullback-Leibler (KL) divergence [38]. Nevertheless, in the context of Federated Learning (FL), assessing the removal of a specific client's contribution through these model difference-based evaluation methods can be challenging. Alternative metrics involve privacy leakage within the differential privacy framework [38] and membership inference attacks [13; 2]. In this study, we adopt backdoor triggers [14] as an effective means to evaluate unlearning methods, akin to Wu et al. [45]. Specifically, the target client employs a dataset containing a fraction of images with inserted backdoor triggers. Consequently, the global FL model becomes susceptible to these triggers. A successful unlearning process should result in a model that reduces accuracy for images with backdoor triggers while maintaining strong performance on regular (clean) images. It is important to note that we employ backdoor triggers solely for the purpose of evaluating unlearning methods; we do not consider any malicious clients [48; 1; 11].

Our Evaluation Metric. We use backdoor attacks in the target client's updates to the global model, as described earlier, to intuitively assess the unlearning effect based on the attack success rate of the unlearned global model. In Table 1, we record the attack success rate as "atk acc". A lower "atk acc" indicates a cleaner removal of the target client's contribution. In our experiments, we implement the backdoor attack using a "pixel pattern" trigger of size 2×2 and change the label to "9". Due to the predictive errors of the model, even the attack success rate after retraining is greater than 0. We can consider the attack success rate after retraining as the benchmark for successful removal of the target client's contribution. Additionally, we use the accuracy metric on clean test data to gauge the model's performance after unlearning, denoted as "test acc" in Table 1. A high accuracy suggests that unlearning has minimal impact on the model's performance.