FedCARE: Federated Unlearning with Conflict-Aware Projection and Relearning-Resistant Recovery

Yue Li[1], Mingmin Chu[1], Xilei Yang[1], Da Xiao[1,2*], Ziqi Xu[3], Wei Shao[3,4,5,6], Qipeng Song[1*], Hui Li[1]
[1]School of Cyber Engineering, Xidian University
[2]Huanghe Science and Technology University
[3]RMIT University
[4]Commonwealth Scientific and Industrial Research Organisation (CSIRO)
[5]UNSW Sydney
[6]University of California, Davis

Abstract

Federated learning (FL) enables collaborative model training without centralizing raw data, but privacy regulations such as the right to be forgotten require FL systems to remove the influence of previously used training data upon request. Retraining a federated model from scratch is prohibitively expensive, motivating federated unlearning (FU). However, existing FU methods suffer from high unlearning overhead, utility degradation caused by entangled knowledge, and unintended relearning during post-unlearning recovery. In this paper, we propose **FedCARE**, a unified and low overhead FU framework that enables conflict-aware unlearning and relearning-resistant recovery. FedCARE leverages gradient ascent for efficient forgetting when target data are locally available and employs data free model inversion to construct class level proxies of shared knowledge. Based on these insights, FedCARE integrates a pseudo-sample generator, conflict-aware projected gradient ascent for utility preserving unlearning, and a recovery strategy that suppresses rollback toward the pre-unlearning model. FedCARE supports client, instance, and class level unlearning with modest overhead. Extensive experiments on multiple datasets and model architectures under both IID and non-IID settings show that FedCARE achieves effective forgetting, improved utility retention, and reduced relearning risk compared to state of the art FU baselines. The source code can be found at https://github.com/thechosenchu/FedCARE.

1 Introduction

Background. Federated Learning (FL) enables collaborative model training without centralizing raw data by aggregating locally computed updates from distributed clients [McMahan *et al.*, 2017]. In practice, an FL system must not only continuously improve the model as new data arrive, but also be able to eliminate the influence of previously used training data upon request in order to comply with privacy regulations. For instance, the European Union's General Data Protection Regulation (GDPR) [Voigt and Von dem Bussche, 2017] establishes the right to erasure, often referred to as the "right to be forgotten" (RTBF). A straightforward way to comply with RTBF is to remove the target data and retrain the federated model from scratch; however, this approach is often prohibitively expensive in terms of computational resources. Inspired by machine unlearning in centralized settings [Bourtoule *et al.*, 2021], Federated Unlearning (FU) [Liu *et al.*, 2021] has emerged as a promising alternative, aiming to remove a target client's (or a data subset's) contribution from a federated global model without retraining from scratch.

Challenges. Despite encouraging progress (refer to Sec. 2 for more details), practical FU still faces the following challenges: **(1) Low-cost and unified unlearning.** A deployable FU framework should complete as quickly as possible upon an unlearning request, while incurring only a small incremental resource cost and minimally affecting non-target clients. Moreover, real unlearning requests arise at different granularities (client-, sample-, and class-level), which calls for a unified design that can cover diverse unlearning paradigms. **(2) Utility degradation due to knowledge entanglement.**Unlearning updates are often computed from a limited view of the data (e.g., only the requesting client), while the global model encodes entangled client-specific and global shared knowledge. As a result, removing a specific contribution can inadvertently damage shared knowledge, leading to noticeable utility drops on retained data, especially under non-IID client distributions.**(3) Relearning during post-unlearning recovery.** To recover model utility, FL commonly continues training on remaining clients after unlearning. Yet this recovery phase can drift the model back toward its pre-unlearning state, implicitly restoring removed knowledge and weakening the unlearning effect.

Method. To address the above challenges, we propose **FedCARE**, a novel FU framework that achieves **C**onflict-**A**ware unlearning and **RE**learning-resistant recovery to retain model utility. The design of FedCARE is motivated by two key observations: (1) when the data to be forgotten are available at

*Corresponding authors.

the target client, gradient ascent provides a simple and low-cost unlearning primitive, provided that its collateral damage to the remaining knowledge can be effectively constrained; and (2) data-free synthesis via model inversion can recover informative class-level signals from a trained model, which can serve as a proxy for shared knowledge and help constrain utility degradation during unlearning without requiring access to real reference data.

Concretely, FedCARE follows a three-stage procedure. First, the server trains a lightweight class-conditional pseudo-sample generator once in a data-free manner and makes it available for future unlearning requests. Second, taking client-level unlearning as an example, the target client uses the generator to synthesize pseudo-samples, computes a reference gradient and performs conflict-aware projected gradient ascent to achieve utility-preserving unlearning. Third, during recovery, the remaining clients freeze their local backbone (feature extractor) and update only the classifier head, while the server filters aggregated updates along a relearning direction to suppress rollback toward the pre-unlearning model. Finally, FedCARE also supports instance- and class-level unlearning by adjusting the target dataset and the corresponding reference gradient, enabling unified unlearning with low overhead.

Contributions. Our contributions are as follows:

- We propose **FedCARE**, a low-overhead FU framework that supports client-, instance-, and class-level unlearning while reducing resource overhead and minimizing impact on non-target clients.
- We introduce a conflict-aware projected forgetting mechanism that constrains forgetting updates with a utility-related protection reference to mitigate utility degradation.
- We design a relearning-resistant recovery strategy that suppresses rollback during post-unlearning recovery via client-specific backbone freezing and server-side update filtering.
- Extensive experiments on multiple datasets and architectures under both IID and non-IID settings show that FedCARE achieves effective forgetting, improved utility retention, and reduced relearning risk with modest additional overhead compared to state-of-the-art FU baselines.

2 Related Work

Federated Unlearning (FU) integrates federated learning with machine unlearning, aiming to enable privacy-preserving data removal in distributed environments. Traditional centralized unlearning methods are unsuitable for such settings due to their reliance on global access to training data. In FU, client data remains local and is never shared with other clients or the central server; instead, only locally computed model updates are transmitted for aggregation into a global model.

Directly deleting target data is infeasible due to the memorization of sensitive information in trained models [Nasr *et al.*, 2019; Melis *et al.*, 2019; Song *et al.*, 2020; Song *et al.*, 2026]. Existing federated unlearning approaches can be broadly categorized into storage-based and storage-free methods, both of which face critical limitations in practice. Storage-based methods [Liu *et al.*, 2021; Lin *et al.*, 2024] accelerate retraining by retaining historical model updates, but incur substantial storage overhead that scales with training duration and client participation, limiting their deployment in long-running or resource-constrained systems. Storage-free methods avoid historical state retention but often sacrifice efficiency or robustness: MoDe [Zhao *et al.*, 2024] relies on knowledge distillation from randomly degraded models, introducing optimization stochasticity and high communication cost; NoT [Khalil *et al.*, 2025] employs heuristic weight negation that severely disrupts learned representations and requires extensive fine-tuning to recover utility; and FedOSD [Pan *et al.*, 2025] mitigates interference via orthogonal projection but depends on continuous client participation for complex gradient computations, exacerbating computational overhead. Collectively, these limitations reveal the lack of a unified federated unlearning framework that can achieve effective forgetting with low overhead, preserve model utility, and remain robust to unintended relearning during recovery.

3 Methodology

In this section, we introduce **FedCARE**, which consists of three main stages: training a data-free pseudo-sample generator, performing conflict-aware projection unlearning, and executing relearning-resistant model recovery. The overall architecture is illustrated in Figure 1.

3.1 Data-Free Pseudo-Sample Generator

FedCARE requires a lightweight, data-free utility reference to support constrained unlearning. Since the server cannot access any real retained data, we construct a proxy reference set $\mathcal{D}_{\text{ref}}$ by synthesizing class-conditional pseudo-samples from the trained global model θ. Importantly, this generator is trained once at the server and can be reused for future unlearning requests.

We implement $\mathcal{G}$ as a lightweight decoder-style network that maps a latent code and a class label to the input space. Given the noise $z \sim \mathcal{N}(0, I)$ and a label y, we first obtain an initial low-resolution feature map $h_0 = \text{Proj}(z, y)$ through a learnable projection layer. The features are then progressively refined and upsampled via a cascade of K decoder blocks:

$$\tilde{x} = \mathcal{G}(z, y) = \phi\big((B_K \circ B_{K-1} \circ \cdots \circ B_1)(h_0)\big), \quad (1)$$

where $\phi(\cdot)$ denotes an output activation function (e.g., $\tanh$ or sigmoid) to ensure that the generated samples lie within the valid input range. Each decoder block $B_i(\cdot)$ increases the spatial resolution while refining semantic structure, thereby producing class-consistent pseudo-samples.

Prior data-free synthesis methods, such as DeepInversion [Yin *et al.*, 2020], typically rely on Batch Normalization (BN) to regularize the generation process by matching stored feature statistics. However, in non-IID federated settings, the BN statistics aggregated at the server often suffer from severe distribution shifts and fail to accurately reflect the true global data distribution [Hsieh *et al.*, 2020]. Relying on such corrupted statistics can lead to the generation of semantic artifacts rather than valid, generic features. To address this issue, we replace BN with Group Normalization (GN) [Wu and He, 2018] in our generator blocks:

$$B_i(h) = \sigma\Big(\text{GN}\big(\text{Conv}_i(\text{Up}_i(h))\big)\Big), \quad (2)$$

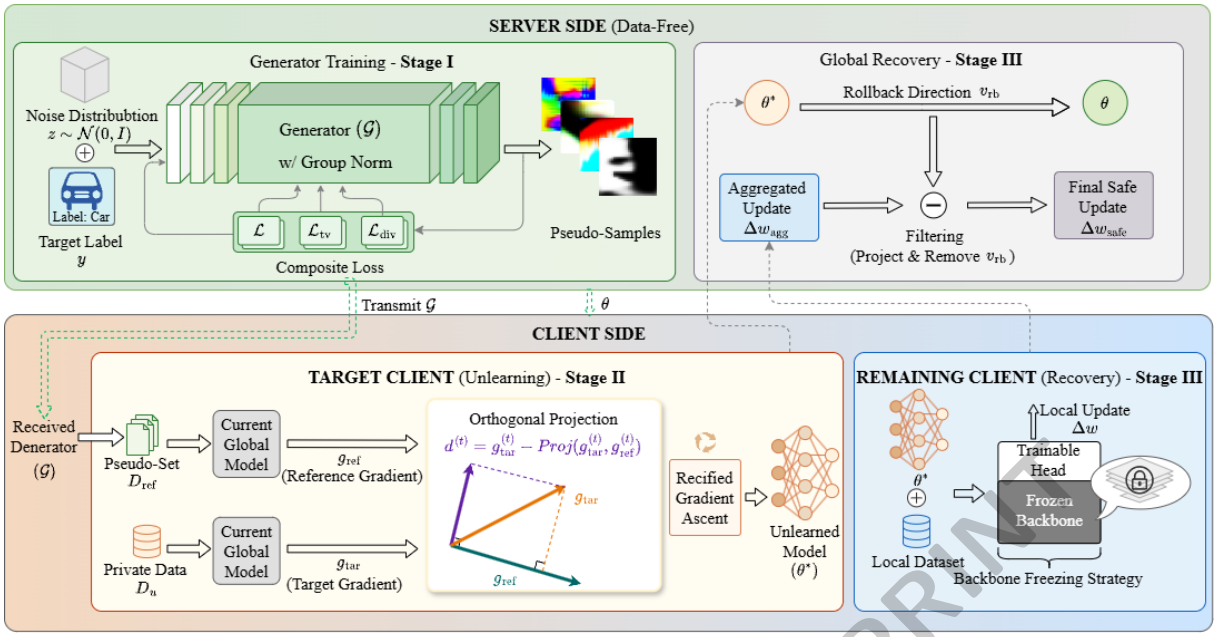


Figure 1: Overall framework illustrated with client-level unlearning. After federated training, the server trains a lightweight data-free generator once. Upon an unlearning request, the generator is sent to the target client, which performs conflict-aware projection unlearning using its private forget set and pseudo-samples. The remaining clients then conduct constrained recovery with relearning-resistant aggregation.

where $\mathrm{Up}(\cdot)$ denotes nearest-neighbor upsampling and $\sigma(\cdot)$ is the activation function. Since GN normalizes features independently of batch statistics, it decouples the generation process from unreliable global distributions, ensuring that the synthesized samples are guided solely by the semantic constraints imposed by the model inversion objective.

To ensure that the constructed feature subspace effectively approximates the generic features of globally shared knowledge, we formulate a unified optimization objective for data-free synthesis as follow:

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \lambda_{\text{tv}}\mathcal{L}_{\text{tv}} + \lambda_{\text{div}}\mathcal{L}_{\text{div}}, \tag{3}$$

where each term plays a distinct role in shaping the synthesized pseudo-samples. We next describe the design and motivation of these components in detail.

To anchor the pseudo-sample $\tilde{x}$ within the correct decision boundary, we adopt the standard cross-entropy loss with respect to the target class y. To mitigate high-frequency artifacts inherent to data-free synthesis [Odena *et al.*, 2016], we further incorporate an anisotropic total variation regularizer,

$$\mathcal{L}_{\text{tv}} = \sum_{i,j} \big(|x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}|\big), \tag{4}$$

which imposes a smoothness prior on the generated samples.

Moreover, to prevent mode collapse and encourage diverse generations, we introduce a diversity regularization term $\mathcal{L}_{\text{div}}$, defined as the inverse of the feature distance ratio:

$$\mathcal{L}_{\text{div}} = \left[\frac{\|\mathcal{G}(z_1, y) - \mathcal{G}(z_2, y)\|_1}{\|z_1 - z_2\|_1} + \epsilon\right]^{-1}. \tag{5}$$

This term enforces a sensitivity constraint on the generator, ensuring that distinct latent codes produce sufficiently different feature realizations and thereby promoting adequate coverage of the latent subspace.

To mitigate gradient domination caused by scale disparity between loss terms (i.e., $\mathcal{L}_{\text{tv}} \gg \mathcal{L}$), we adopt a hybrid loss balancing strategy with adaptive initialization of the smoothness weight λ_{tv}. At the start of training ($t = 0$), λ_{tv} is set as $\lambda_{\text{tv}} = \gamma \cdot (\mathcal{L}^{(0)}/\mathcal{L}_{\text{tv}}^{(0)})$, where $\gamma \in [10^{-3}, 10^{-1}]$ is an empirical attenuation factor inversely related to the dataset's textural complexity. This normalization aligns the gradient magnitudes of the smoothness and semantic objectives, preventing early stage optimization bias. In contrast, we assign a fixed constant weight to the diversity term, setting λ_{div} to a small value (e.g., 0.5) to maintain a stable repulsive force for feature space exploration.

3.2 Conflict-Aware Projection Unlearning

For clarity, we take client-level unlearning as an illustrative example. At the onset of unlearning, the server shares the trained pseudo-sample generator $\mathcal{G}$ with the target client u. Let θ denote the current global model parameters and $\mathcal{D}_u$ the private dataset to be forgotten.

A simple and low-cost primitive for client unlearning is gradient ascent on $\mathcal{D}_u$, which maximizes the empirical loss to erase the client-specific contribution. However, directly maximizing $\mathcal{L}_u(\theta) \triangleq \mathcal{L}(\theta; \mathcal{D}_u)$ can severely degrade model utility, as client-specific and globally shared representations are entangled in θ. To address this issue, we formulate unlearning as a constrained optimization problem:

$$\max_{\Delta\theta} \mathcal{L}_u(\theta + \Delta\theta) \quad \text{s.t.} \quad \mathcal{L}_{\text{ref}}(\theta + \Delta\theta) \le \mathcal{L}_{\text{ref}}(\theta), \tag{6}$$

where $\mathcal{L}_{\text{ref}}(\theta)$ denotes a reference loss that serves as a proxy for retained utility. In practice, $\mathcal{L}_{\text{ref}}$ cannot be evaluated on

real retained data due to data isolation and privacy constraints. FedCARE overcomes this limitation by constructing a proxy reference dataset $\mathcal{D}_{\text{ref}}$ using the generator $\mathcal{G}$ (for class-level unlearning, $\mathcal{D}_{\text{ref}}$ excludes the forgotten class), and defining the reference loss as $\mathcal{L}_{\text{ref}}(\theta) \triangleq \mathcal{L}(\theta; \mathcal{D}_{\text{ref}})$.

Directly solving Eq. (6) is intractable for deep neural networks. We therefore enforce the utility constraint locally using a first-order approximation. Let d denote the ascent direction and consider an update of the form $\Delta\theta = \eta d$ with a small step size η. Applying a first-order Taylor expansion to the reference loss yields:

$$\mathcal{L}_{\text{ref}}(\theta + \eta d) \approx \mathcal{L}_{\text{ref}}(\theta) + \eta \left\langle \nabla_\theta \mathcal{L}_{\text{ref}}(\theta),\, d \right\rangle. \tag{7}$$

Thus, a sufficient condition to prevent an increase in the reference loss at first order is:

$$\langle g_{\text{ref}},\, d \rangle \le 0, \quad \text{where} \quad g_{\text{ref}} \triangleq \nabla_\theta \mathcal{L}_{\text{ref}}(\theta). \tag{8}$$

Meanwhile, the unconstrained ascent direction for forgetting is given by:

$$g_{\text{tar}} \triangleq \nabla_\theta \mathcal{L}_u(\theta). \tag{9}$$

Given the unconstrained ascent direction g_{tar} and the first-order utility constraint in Eq. (8), the problem reduces to finding a feasible update direction that remains as close as possible to g_{tar} while preserving the reference loss.

In practice, computing g_{ref} using the entire proxy dataset $\mathcal{D}_{\text{ref}}$ at every iteration is computationally unnecessary. Moreover, the local constraint in Eq. (8) is defined with respect to the current parameters $\theta^{(t)}$, and the associated gradient field evolves as the model is updated. To account for this dynamics while maintaining efficiency, we estimate the reference gradient online using a mini-batch $\mathcal{B}_{\text{ref}}^{(t)} \subset \mathcal{D}_{\text{ref}}$ at each step t:

$$g_{\text{ref}}^{(t)} \triangleq \nabla_\theta \mathcal{L}\Big(\theta^{(t)}; \mathcal{B}_{\text{ref}}^{(t)}\Big), \tag{10}$$

which serves as a lightweight and step-adaptive proxy for retained utility. Similarly, the forgetting gradient is computed on a mini-batch $\mathcal{B}_u^{(t)} \subset \mathcal{D}_u$:

$$g_{\text{tar}}^{(t)} \triangleq \nabla_\theta \mathcal{L}\Big(\theta^{(t)}; \mathcal{B}_u^{(t)}\Big). \tag{11}$$

At each step t, we obtain the closest feasible ascent direction to $g_{\text{tar}}^{(t)}$ by solving the following constrained optimization problem:

$$d^{(t)} = \arg\min_d \left\| d - g_{\text{tar}}^{(t)} \right\|_2^2 \quad \text{s.t.} \quad \left\langle g_{\text{ref}}^{(t)},\, d \right\rangle \le 0, \tag{12}$$

which projects $g_{\text{tar}}^{(t)}$ onto the half-space defined by the local utility constraint. This problem admits a closed-form solution given by:

$$d^{(t)} = g_{\text{tar}}^{(t)} - \frac{\max\Big(0, \Big\langle g_{\text{tar}}^{(t)},\, g_{\text{ref}}^{(t)} \Big\rangle\Big)}{\left\| g_{\text{ref}}^{(t)} \right\|_2^2 + \epsilon}\, g_{\text{ref}}^{(t)}, \tag{13}$$

where ϵ is a small constant introduced for numerical stability. Intuitively, when $\langle g_{\text{tar}}^{(t)}, g_{\text{ref}}^{(t)} \rangle \le 0$, the unconstrained ascent direction already satisfies the utility constraint and remains unchanged. Otherwise, the minimal conflicting component along $g_{\text{ref}}^{(t)}$ is removed so that the corrected direction lies exactly on the constraint boundary.

Finally, the target client updates the model parameters via projected gradient ascent for T steps:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \eta_{\text{ul}}\, d^{(t)}, \qquad t = 0, \ldots, T-1, \tag{14}$$

where η_{ul} denotes the unlearning learning rate.

3.3 Relearning-resistant Model Recovery

Although the projection mechanism in Sec. 3.2 constrains utility degradation during the erasure process, the performance on retained tasks may still degrade in practice. Consequently, FU methods commonly adopt a post unlearning recovery stage, such as standard FedAvg fine tuning, by continuing federated training over the remaining clients' local datasets, denoted by $\mathcal{D}_{\text{rem}}$. However, such recovery procedures can inadvertently reintroduce partially removed knowledge, thereby weakening the effectiveness of unlearning [Chen *et al.*, 2021; Pan *et al.*, 2025].

Following [Ilharco *et al.*, 2023], we adopt task vectors to represent weight space directions that encode task specific changes. Specifically, the knowledge removed during unlearning can be characterized as:

$$\Delta\theta_{\text{ul}} \triangleq \theta^* - \theta, \tag{15}$$

where θ denotes the global model parameters immediately before unlearning and θ^* denotes the unlearned model.

Similarly, the knowledge acquired during the recovery stage is given by:

$$\Delta\theta_{\text{rec}} \triangleq \theta^{*,r} - \theta^*, \tag{16}$$

where $\theta^{*,r}$ denotes the recovered model. Relearning, also referred to as rollback, becomes prominent when recovery updates move the model back toward θ, that is, when $\Delta\theta_{\text{rec}}$ contains a positive component along the rollback direction $\theta - \theta^* = -\Delta\theta_{\text{ul}}$:

$$\langle \Delta\theta_{\text{rec}},\, -\Delta\theta_{\text{ul}} \rangle > 0. \tag{17}$$

Accordingly, a recovery strategy should explicitly suppress update components that align with the rollback direction.

FedCARE mitigates this effect through a dual client and server design. At the client side, we decompose the original and unlearned models as $\theta = (\phi, w)$ and $\theta^* = (\phi^*, w^*)$, where ϕ and w denote the backbone feature extractor and classifier head, respectively. During recovery, each remaining client freezes the backbone at ϕ^* and updates only the classifier head for a small number of rounds. This restricts recovery to adjusting decision boundaries within the post unlearning representation space, reducing the risk of reconstructing previously removed features.

At the server side, with the backbone fixed, recovery updates are confined to the classifier head subspace. Suppressing rollback in the full parameter space therefore reduces to preventing head updates from moving toward the original head w. Nevertheless, due to client data heterogeneity and optimization noise, the aggregated head update may still contain a non negligible rollback component. To address this, we

further filter the aggregated head update along a fixed rollback direction in the head subspace.

Following the server side filtering strategy described above, we explicitly characterize the rollback direction in the classifier head subspace. We define the normalized rollback direction as:

$$\mathbf{v}_{\mathrm{rb}} \triangleq \mathrm{Normalize}(w - w^*)\,, \tag{18}$$

where w denotes the original classifier head and w^* denotes the head after unlearning. Let Δw_{agg} be the standard aggregated head update from the remaining clients in a recovery round. Analogous to the conflict aware projection in Sec. 3.2, FedCARE computes the closest corrected update that does not move toward w along $\mathbf{v}_{\mathrm{rb}}$, which admits the following closed form projection:

$$\Delta w_{\mathrm{safe}} = \Delta w_{\mathrm{agg}} - \frac{\max(0, \langle \Delta w_{\mathrm{agg}}, \mathbf{v}_{\mathrm{rb}} \rangle)}{\|\mathbf{v}_{\mathrm{rb}}\|_2^2 + \epsilon} \cdot \mathbf{v}_{\mathrm{rb}}. \tag{19}$$

This filtering guarantees that $\langle \Delta w_{\mathrm{safe}}, \mathbf{v}_{\mathrm{rb}} \rangle \leq 0$, which eliminates first order rollback along the defined direction while still allowing effective utility recovery within the constrained parameter space.

3.4 Instance- and Class-level Unlearning Support

FedCARE supports different unlearning granularities with minimal modifications to the pipeline described in Secs. 3.2-3.3. When only a subset of samples on the target client needs to be removed, we define the forget set as $\mathcal{D}_f \subset \mathcal{D}_u$ and compute the forgetting update using $\mathcal{D}_f$, that is, by replacing $\mathcal{D}_u$ with $\mathcal{D}_f$ in the forget loss and gradient in Sec. 3.2. The utility reference construction via the pseudo-sample generator $\mathcal{G}$, the conflict-aware projection, and the subsequent recovery procedure remain unchanged.

For class-level unlearning of a class c, we apply two coordinated modifications. During unlearning, the forgetting update is computed using only local samples labeled c on the target client (e.g., the client with the largest amount of class c data). When constructing the utility reference, $\mathcal{G}$ synthesizes pseudo-samples only from the remaining label set excluding c, ensuring that the projection constraint does not preserve the removed class. Meanwhile, all remaining clients remove class c samples from their local datasets and follow the same recovery protocol described in Sec. 3.3.

4 Experiments

4.1 Experimental Setup

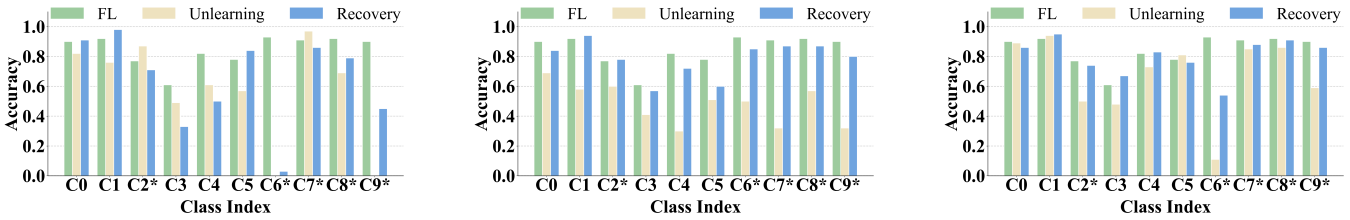
Datasets, Models, and Experimental Settings. We evaluate FedCARE on four standard datasets: MNIST [LeCun *et al.*, 1998], SVHN [Netzer *et al.*, 2011], and CIFAR-10/100 [Alex, 2009]. For MNIST, we adopt a simple CNN consisting of two convolutional layers followed by two fully connected layers. For SVHN, CIFAR-10, and CIFAR-100, we use ResNet-18 [He *et al.*, 2016] as the backbone model. In federated learning, we employ the standard FedAvg algorithm for training with 10 clients under both IID and non-IID settings. For the non-IID case, data are partitioned according to a Dirichlet distribution $\mathrm{Dir}(\alpha)$ with concentration parameter $\alpha = 0.1$ [Yurochkin *et al.*, 2019].

Baselines. We compare FedCARE with six baselines, including Retraining and five state-of-the-art FU methods: FedEraser [Liu *et al.*, 2021], FedSE [Lin *et al.*, 2024], MoDe [Zhao *et al.*, 2024], FedOSD [Pan *et al.*, 2025], and NoT [Khalil *et al.*, 2025]. Note that FedEraser and FedSE are limited to client-level unlearning.

Evaluation Metrics. We evaluate the effectiveness and efficiency of FedCARE using the following metrics: (1) Accuracy Metrics: We use R-Acc to denote the classification accuracy on classes excluding the unlearning target, measuring the retention of non-target knowledge. U-Acc represents the accuracy on classes containing the forgotten data for assessing forgetting effectiveness, while Test denotes the overall accuracy of the global model on the complete test set. For U-Acc, R-Acc and Test, the ideal objective is to closely match the performance of the Retraining baseline. (2) Membership Inference Attack (MIA) [Yeom *et al.*, 2018]: This metric quantifies privacy leakage by measuring the extent to which the forgotten dataset $\mathcal{D}_u$ can still be inferred as part of the training data. (3) Attack Success Rate (ASR): We employ ASR to assess whether implanted backdoor triggers have been fully removed from the model by injecting triggers into the target client's data and flipping the corresponding labels following [Halimi *et al.*, 2022; Gu *et al.*, 2017]. (4) Communication and Computation Costs: To evaluate system efficiency, we report the total time overhead incurred during the unlearning and recovery phases as the communication cost, along with the FLOPs [Kaplan *et al.*, 2020] consumed by the remaining clients as the computation cost.

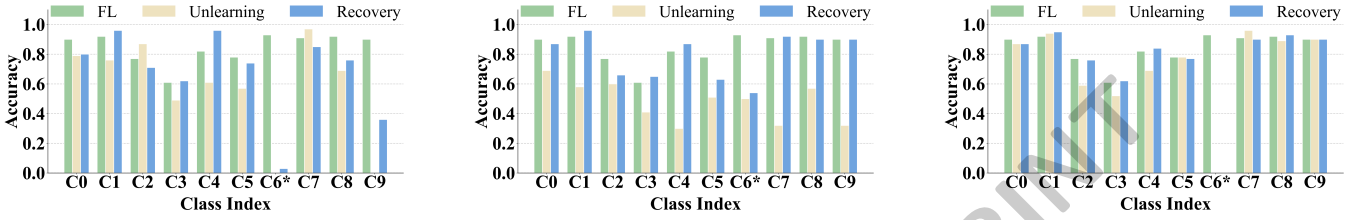
4.2 Main Results

Unlearning Effectiveness and Utility Maintenance We evaluate utility maintenance across three granularities: client, instance, and class levels. For client-level unlearning, Table 1 shows that FedCARE achieves a U-Acc of 78.52% on CIFAR-10, closely matching the Retraining baseline (78.63%), indicating effective disentanglement of client-specific and shared knowledge. In contrast, partition-based and distillation-based methods, such as FedEraser and MoDe, suffer concurrent drops in both U-Acc and R-Acc, reflecting limited generalization. For instance-level unlearning (Table 3), FedCARE maintains an R-Acc of 82.21% on CIFAR-10, comparable to Retraining (82.52%), demonstrating minimal collateral impact on global decision boundaries. For class-level unlearning, Table 2 shows that FedCARE uniquely achieves complete erasure, driving U-Acc to 0% under both IID and non-IID settings while preserving R-Acc close to Retraining. By contrast, FedOSD retains residual U-Acc under both IID and non-IID settings (3.54% and 2.76%), and NoT preserves substantial accuracy (54.10%), indicating ineffective removal of the target concept. Figures 2 and 3 further confirm that FedCARE incurs minimal utility degradation on retained classes during both unlearning and recovery. Since FedOSD approximates class-level unlearning via client-level unlearning, it exhibits similar performance trends across Figures 2a and 3a under identical data partitions.



(a) FedOSD (b) NoT (c) FedCARE

Figure 2: Client-level unlearning performance across different classes on CIFAR-10 (non-IID). The (∗) denotes classes contained within the unlearning client.



(a) FedOSD (b) NoT (c) FedCARE

Figure 3: Class-level unlearning performance across different classes on CIFAR-10 (non-IID). The (∗) denotes the class to be forgotten.

	Method	Accuracy			Privacy	Cost	
		R-Acc	U-Acc	Test	MIA (↓)	Time (↓)	FLOPs (↓)
MNIST	Retrain	95.59%	94.43%	95.11%	50.21%	2441.20s	7.67×10^{12}
	FedEraser	94.85%	96.11%	94.42%	56.16%	631.64s	3.28×10^{12}
	FedSE	85.44%	72.10%	83.63%	63.80%	500.24s	1.30×10^{12}
	MoDe	93.62%	90.55%	91.50%	53.72%	796.75s	7.57×10^{11}
	FedOSD	95.05%	92.17%	93.81%	53.47%	180.20s	4.12×10^{11}
	NoT	95.61%	94.70%	95.24%	54.24%	210.78s	6.11×10^{11}
	FedCARE	95.11%	93.80%	94.91%	50.18%	54.41s	1.54×10^{11}
SVHN	Retrain	90.98%	84.27%	86.37%	50.13%	7565.44s	6.52×10^{15}
	FedEraser	90.14%	88.61%	86.16%	58.42%	1420.35s	1.85×10^{15}
	FedSE	79.43%	68.25%	78.50%	63.83%	1150.28s	2.68×10^{15}
	MoDe	85.20%	79.16%	84.89%	61.58%	2890.61s	1.15×10^{15}
	FedOSD	91.88%	81.14%	85.21%	53.60%	385.49s	6.90×10^{14}
	NoT	90.92%	83.33%	85.32%	54.10%	3410.10s	2.38×10^{15}
	FedCARE	90.46%	84.51%	86.36%	50.22%	90.66s	2.07×10^{14}
CIFAR-10	Retrain	81.62%	78.63%	80.37%	50.28%	7892.47s	5.66×10^{15}
	FedEraser	80.97%	75.15%	77.58%	57.30%	1466.31s	2.25×10^{15}
	FedSE	66.45%	58.20%	65.10%	63.45%	1372.27s	3.29×10^{15}
	MoDe	77.80%	62.50%	72.35%	60.54%	3365.68s	1.49×10^{15}
	FedOSD	83.11%	73.26%	79.64%	54.10%	487.29s	8.65×10^{14}
	NoT	81.58%	77.53%	79.33%	57.42%	4078.19s	2.98×10^{15}
	FedCARE	81.30%	78.52%	80.01%	50.38%	125.39s	1.49×10^{14}
CIFAR-100	Retrain	61.98%	56.01%	60.03%	50.19%	15474.10s	1.25×10^{16}
	FedSE	40.60%	32.17%	38.45%	66.10%	4159.66s	3.13×10^{15}
	MoDe	47.88%	51.32%	46.20%	64.08%	5741.02s	1.66×10^{15}
	FedOSD	62.40%	58.90%	61.85%	56.21%	873.95s	1.05×10^{15}
	NoT	61.77%	55.46%	59.68%	50.15%	2955.30s	2.67×10^{15}
	FedCARE	61.11%	55.32%	59.64%	50.22%	250.77s	4.07×10^{14}

Table 1: Client-level unlearning performance under non-IID settings. We report results under non-IID scenarios to reflect more challenging and realistic unlearning conditions, and omit IID results where data redundancy tends to yield overly optimistic performance.

Efficiency and Scalability Analysis FedCARE demonstrates strong efficiency in both execution time and computational overhead (Figure 4). As shown in Table 1, FedCARE

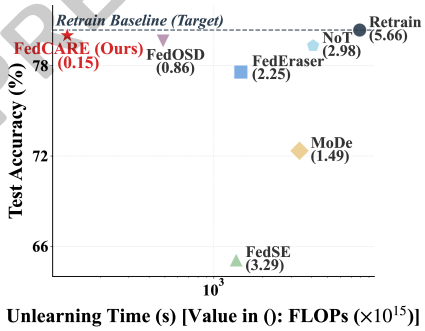


Figure 4: Efficiency comparison between FedCARE and baseline methods on CIFAR-10 under the non-IID setting.

completes CIFAR-10 unlearning in 125.39s, yielding a 63× speedup over Retrain and a 32× speedup over NoT (4078s), which incurs high latency due to iterative fine-tuning for repairing disruptive weight negation. Beyond wall-clock time, the frozen-backbone recovery strategy significantly reduces computation, limiting FLOPs to 1.49×10^{14}. This is substantially lower than the compute-intensive FedOSD (8.65×10^{14}) and an order of magnitude more efficient than storage-heavy methods such as FedSE, highlighting FedCARE's suitability for resource-constrained edge environments.

Privacy and Security Verification We evaluate data privacy using MIA and model security using backdoor defense. As shown in Table 3 (and Table 1 for client-level unlearning), FedCARE achieves an instance-level MIA score of 50.38% on CIFAR-10, which is close to random guessing, indicating that membership information of forgotten samples is effectively removed. In contrast, NoT and FedEraser exhibit higher MIA scores ($> 57\%$), suggesting that identifiable traces remain in the model. For backdoor defense (Ta-

	Method	non-IID				IID			
		U-Acc	R-Acc	Time (↓)	FLOPs (↓)	U-Acc	R-Acc	Time (↓)	FLOPs (↓)
CIFAR-10	Retrain	0	83.96%	8587.48s	7.23×10^{15}	0	84.69%	7291.03s	6.02×10^{15}
	MoDe	0	76.59%	3105.61s	1.36×10^{15}	1.57%	77.20%	3000.54s	1.12×10^{15}
	FedOSD	2.76%	81.77%	483.28s	8.17×10^{14}	3.54%	82.77%	477.31s	8.05×10^{14}
	NoT	54.10%	78.53%	6969.16s	6.02×10^{15}	33.90%	85.83%	6143.88s	5.42×10^{15}
	FedCARE	0	83.93%	300.53s	7.45×10^{14}	0	83.34%	280.87s	7.53×10^{14}

Table 2: Class-level unlearning performance. Class 6 is removed from all clients.

	Method	Accuracy			Privacy	Cost	
		R-Acc	U-Acc	Test	MIA (↓)	Time (↓)	FLOPs (↓)
CIFAR-10	Retrain	82.52%	80.79%	82.08%	50.71%	8696.20s	6.21×10^{15}
	FedOSD	84.29%	79.04%	82.13%	54.38%	517.14s	9.61×10^{14}
	NoT	80.07%	78.36%	79.10%	58.65%	3780.26s	3.31×10^{15}
	FedCARE	82.21%	80.50%	81.56%	51.38%	137.79s	1.65×10^{14}

Table 3: Instance-level unlearning performance under non-IID settings.

	Method	Accuracy	Security	Cost	
		Test	ASR (↓)	Time (↓)	FLOPs (↓)
CIFAR-10	Retrain	80.01%	0	7892.47s	5.66×10^{15}
	FedEraser	78.33%	12.56%	1429.02s	2.17×10^{15}
	FedSE	62.72%	8.71%	1380.71s	3.43×10^{15}
	MoDe	73.45%	25.87%	3437.68s	1.72×10^{15}
	FedOSD	79.31%	0	479.26s	8.65×10^{14}
	NoT	77.66%	38.64%	4022.94s	2.77×10^{15}
	FedCARE	79.42%	0	200.37s	1.89×10^{14}

Table 4: Backdoor defense performance under non-IID settings.

Method	Accuracy			Privacy	Security
	R-Acc	U-Acc	Test	MIA (↓)	ASR (↓)
Retrain	81.62%	78.60%	80.01%	50.20%	0.00%
M1	80.37%	79.44%	79.06%	51.36%	33.50%
M2	77.20%	67.00%	75.44%	50.38%	0.00%
M3	81.66%	77.54%	76.79%	57.12%	45.77%
M4	82.74%	80.70%	79.95%	55.35%	29.60%
M5	81.16%	79.82%	78.04%	51.67%	20.71%
FedCARE	81.30%	78.52%	78.42%	50.38%	0.00%

Table 5: Ablation study on client-level unlearning. Performance comparison of different components under the non-IID setting.

ble 4), FedCARE reduces the ASR to 0% while maintaining high utility (79.42%). Although FedOSD also eliminates the backdoor, it incurs substantially higher computational cost, whereas MoDe and NoT fail to fully remove embedded triggers, with ASR exceeding 25%, which poses serious security risks and limits their applicability in safety-critical scenarios.

4.3 Ablation Experiments

To rigorously assess the contribution of each component in FedCARE, we conduct ablation studies on the non-IID CIFAR-10 dataset using ResNet-18. Specifically, we consider five variants: (M1) replacing Group Normalization in the generator with Batch Normalization to examine the effect of normalization under heterogeneous data distributions; (M2) performing unlearning via raw gradient ascent on private data without the projection constraint or protective gradient to evaluate the necessity of conflict-aware projection; (M3) applying standard FedAvg for post-unlearning recovery without any anti-relearning mechanism to expose rollback risks; (M4) disabling backbone freezing during recovery while retaining server-side projection to test whether global calibration alone can prevent feature reconstruction; and (M5) removing server-side vector filtering while keeping local backbone freezing to examine whether local constraints alone suffice to suppress directional rollback of the classifier head.

As shown in Table 5, the ablation results validate the necessity of each component in FedCARE. During unlearning, removing the projection constraint (M2) yields the lowest R-Acc (77.20%), confirming that unconstrained gradient ascent severely degrades model utility, while replacing Group Normalization with Batch Normalization in the generator (M1) harms both utility and security, leading to a high ASR of 33.50% and highlighting the importance of constructing a reliable generic knowledge subspace under non-IID settings. During recovery, the FedAvg baseline (M3) exhibits severe relearning with an ASR of 45.77%, and disabling local backbone freezing (M4) causes a substantial ASR rebound to 36.60%, indicating that the backbone is a primary carrier of backdoor features and that server-side projection alone is insufficient to prevent reconstruction. Although retaining backbone freezing but removing global vector filtering (M5) reduces ASR to 20.71%, residual directional drift remains. In contrast, FedCARE combines backbone freezing with vector filtering to achieve complete backdoor removal (0% ASR) while maintaining strong utility (81.30% R-Acc), demonstrating the effectiveness of the proposed dual-constraint recovery mechanism.

5 Conclusion

This paper presents FedCARE, a unified and low-overhead federated unlearning framework that addresses key challenges in FU, including high unlearning cost, utility degradation, and unintended relearning during recovery. By combining conflict-aware projected unlearning with a relearning-resistant recovery mechanism, FedCARE enables effective removal of client-, instance-, and class-level information without retraining from scratch. Extensive experiments across multiple datasets and model architectures under both IID and non-IID settings demonstrate that FedCARE achieves reliable forgetting, strong utility retention, and improved robustness compared to state-of-the-art FU methods. These results indicate that FedCARE provides a practical solution for privacy-compliant federated learning systems.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant no. U24A20240, no. 62441226).

References

[Alex, 2009] Krizhevsky Alex. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*, 2009.

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

[Chen *et al.*, 2021] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 896–911. ACM, 2021.

[Gu *et al.*, 2017] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017.

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

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[Hsieh *et al.*, 2020] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. The non-iid data quagmire of decentralized machine learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4387–4398. PMLR, 2020.

[Ilharco *et al.*, 2023] Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Kaplan *et al.*, 2020] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[Khalil *et al.*, 2025] Yasser H. Khalil, Leo Maxime Brunswic, Soufiane Lamghari, Xu Li, Mahdi Beitollahi, and Xi Chen. Not: Federated unlearning via weight negation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 25759–25769. Computer Vision Foundation / IEEE, 2025.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[Lin *et al.*, 2024] Yijing Lin, Zhipeng Gao, Hongyang Du, Dusit Niyato, Gui Gui, Shuguang Cui, and Jinke Ren. Scalable federated unlearning via isolated and coded sharding. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 4551–4559. ijcai.org, 2024.

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

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 1273–1282, 2017.

[Melis *et al.*, 2019] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy, SP*, pages 691–706, 2019.

[Nasr *et al.*, 2019] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *IEEE Symposium on Security and Privacy, SP*, pages 739–753, 2019.

[Netzer *et al.*, 2011] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, 2011.

[Odena *et al.*, 2016] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[Pan *et al.*, 2025] Zibin Pan, Zhichao Wang, Chi Li, Kaiyan Zheng, Boqi Wang, Xiaoying Tang, and Junhua Zhao. Federated unlearning with gradient descent and conflict mitigation. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 19804–19812. AAAI Press, 2025.

[Song *et al.*, 2020] 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*, 38:2430–2444, 2020.

[Song *et al.*, 2026] Qipeng Song, Nan Yang, Ziqi Xu, Yue Li, Wei Shao, and Feng Xia. Synthetic forgetting without access: A few-shot zero-glance framework for machine unlearning. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 25545–25553. AAAI Press, 2026.

[Voigt and Von dem Bussche, 2017] 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.

[Wu and He, 2018] Yuxin Wu and Kaiming He. Group normalization. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, volume 11217 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2018.

[Yeom *et al.*, 2018] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 268–282. IEEE Computer Society, 2018.

[Yin *et al.*, 2020] Hongxu Yin, Pavlo Molchanov, José M. Álvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via DeepInversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8712–8721, Seattle, WA, USA, June 2020. Computer Vision Foundation / IEEE.

[Yurochkin *et al.*, 2019] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan H. Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7252–7261. PMLR, 2019.

[Zhao *et al.*, 2024] Yian Zhao, Pengfei Wang, Heng Qi, Jianguo Huang, Zongzheng Wei, and Qiang Zhang. Federated unlearning with momentum degradation. *IEEE Internet of Things Journal*, 11(5):8860–8870, 2024.