FEDCARR: CONTRIBUTION-AWARE RETENTION AND ROLLBACK OPTIMIZATION FOR FEDERATED UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated unlearning (FU) aims to remove the influence of clients requesting data deletion from a collaboratively trained model, but existing approaches primarily focus on the operational execution of unlearning while overlooking the economic decisions surrounding batched deletion requests. In practical federated learning systems, a server may have a limited compensation budget and must jointly determine which clients to temporarily retain and which historical model state to use for subsequent unlearning. We propose FedCARR, a two-stage mechanism for contribution-aware batched client-level FU. In the first stage, the server assigns contribution-weighted compensation caps and determines individually rational retention decisions under a hard budget constraint. These decisions and associated payments are then fixed as binding commitments. In the second stage, given the resulting set of clients to be forgotten, the server selects an executable rollback state from loadable or reconstructible historical checkpoints by jointly considering retained model utility, compensation expenditure, and delayed-deletion privacy costs. We theoretically establish individual rationality, retention-decision stability, the existence and optimality of feasible rollback decisions, and the existence of an optimal solution to the formulated problem. Extensive experiments on CIFAR-10, CIFAR-100, and Fashion-MNIST evaluate the tradeoffs among model utility, compensation cost, deletion delay, and rollback requirements, demonstrating the effectiveness of the proposed mechanism for budget-constrained FU.

1 INTRODUCTION

Federated learning (FL) (McMahan et al., 2017; Kairouz et al., 2021) enables multiple clients to collaboratively train a shared model without directly transmitting their raw data to a central server. This decentralized learning paradigm has been widely adopted in privacy-sensitive applications (Wu et al., 2022), where data are inherently distributed across heterogeneous clients. However, keeping raw data local does not eliminate the privacy risks associated with FL. The information contained in clients' model updates can still be encoded into the global model, potentially allowing an adversary to infer whether a particular record participated in the training process through membership inference attacks (Shokri et al., 2017). Consequently, when a client withdraws its consent or requests the deletion of its data, merely removing the raw data from the client is insufficient: the influence of the corresponding data must also be removed from the trained model.

This requirement has motivated the development of Federated Unlearning (FU), which aims to eliminate the contribution of specified clients from a collaboratively trained model (Cao & Yang, 2015; Ginart et al., 2019; Bourtoule et al., 2021). Compared with conventional centralized machine unlearning, FU is considerably more challenging because the server does not possess clients' raw data, client data distributions are generally non-IID, and client influence propagates across multiple aggregation rounds (Kairouz et al., 2021; Li et al., 2020)

A straightforward approach to FU is to completely retrain the model using only the remaining clients after receiving deletion requests. Although this approach can provide a natural reference for effective unlearning, its computational and communication costs can be comparable to those of the original FL training process, making it unattractive for resource-constrained edge environments. To reduce

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

this overhead, a number of efficient FU approaches have been developed. For example, FedEraser (Liu et al., 2021) reconstructs historical updates to avoid complete retraining, FedRecovery Zhang et al. (2023) perturbs historical updates to recover a model with reduced computational cost, FFMU Che et al. (2023) approximates the effect of deleting a client, SIFU Fraboni et al. (2024) identifies sensitivity-aware rollback states, and FedAU Gu et al. (2024) incorporates anticipated future deletion requests into the training process.FATS Tao et al. (2024) modifies the federated training procedure to provide communication-efficient and provable exact unlearning at both the client and sample levels. More recently, FedOSD (Pan et al., 2025) replaces direct gradient ascent with an unlearning cross-entropy objective and computes conflict-mitigated update directions to preserve utility on the remaining clients, highlighting the need to assess utility, forgetting effectiveness, efficiency, storage overhead, and client-side verifiability (Nguyen et al., 2024; Romandini et al., 2025; Gao et al., 2024).

These approaches have made substantial progress toward the efficient execution of unlearning. Nevertheless, they predominantly focus on the operational question of how to remove the influence of a given set of clients. An important question remains largely unexplored: when multiple clients request data deletion and the server has a limited budget for compensation, which requests should be accommodated, and how should the corresponding unlearning state be selected?

This question is particularly important in practical FL systems because deletion requests are inherently heterogeneous. Clients may make unequal contributions to the global model (Ghorbani & Zou, 2019; Liu et al., 2022), exhibit different levels of privacy sensitivity, and have different tolerances for delayed deletion. Consequently, instead of immediately removing all requesting clients, a server may benefit from compensating selected requesters and temporarily retaining their contributions. However, such retention decisions are intrinsically coupled with the subsequent unlearning process. On the one hand, compensating a requester may preserve valuable model utility, whereas excessive compensation may consume the available budget and reduce the server's overall benefit. On the other hand, retaining some requesters changes the set of clients that must eventually be forgotten, thereby changing the rollback state required for unlearning. A later rollback state generally preserves more useful training progress, but it also prolongs the period during which the forgotten clients' influence remains in the model. Therefore, retention, compensation, and rollback-state selection should be considered jointly rather than optimized independently.

Existing FL incentive studies cover participation, contribution evaluation, fairness, and strategic behavior (Nair et al., 2025), while studies examine multi-stage learning and unlearning interactions and rewards for clients supporting deletion operations (Ding et al., 2025; Lin et al., 2024). However, these studies do not jointly address two key decisions: which deletion requesters should be retained under a hard compensation budget and which executable historical state should subsequently be selected for unlearning. This gap becomes even more significant in practical systems where the server cannot arbitrarily rollback to any point on a continuous training trajectory. Historical checkpoints may only be sparsely stored, while intermediate states may need to be reconstructed, and some theoretically optimal rollback points may therefore be unavailable. Thus, an executable FU mechanism should optimize over the set of historical states that the server can actually load or reconstruct, rather than over an idealized continuous training timeline.

To address these challenges, we propose FedCARR, a feasibility-first two-stage mechanism for batched client-level federated unlearning. The key design principle is to separate irreversible retention commitments from subsequent rollback optimization. In the first stage, the server assigns each deletion requester a contribution-weighted compensation cap and derives a critical cap-coefficient threshold that characterizes whether retaining the requester is financially feasible. A requester is retained only if its allocated compensation cap can cover its minimum acceptable payment. Once determined, the retention decisions and corresponding payments are irrevocably fixed. This binding commitment prevents the subsequent rollback optimization from changing the economic outcome that has already been promised to the requesters. In the second stage, given the resulting set of clients to be forgotten, the server searches over the set of loadable or reconstructible historical states and selects the rollback state that jointly accounts for retained model value, compensation expenditure, and the privacy cost induced by delayed deletion. As a result, the proposed mechanism connects the economic decision of whom to retain with the operational decision of where to roll back.

The main contributions of this work are summarized as follows.

- We formulate a compensation-cap-constrained batched FU problem that jointly considers client contribution, compensation, retention, delayed deletion, and historical-state recoverability.
- We propose a two-stage mechanism, FedCARR, which determines contribution-aware client retention and compensation, followed by executable rollback-round selection. We theoretically establish individual rationality and decision stability, the existence and optimality of feasible rollback decisions, and the existence of an optimal solution to the formulated problem.
- We perform extensive evaluations on three benchmark datasets, demonstrating favorable tradeoffs among utility, compensation cost, deletion delay, and rollback requirements.

2 System Model and Problem Formulation

2.1 Federated Learning System

A central server coordinates federated learning (FL) among a set of clients $\mathcal{N} = \{1, \dots, N\}$, where client i holds a private local dataset $\mathcal{D}_i$. In communication round t, a subset of clients $\mathcal{N}_t \subseteq \mathcal{N}$ participates in local training and transmits its updated model to the server. The server aggregates the received local models according to a predefined aggregation rule. For example, under FedAvg, the global model is updated as

$$\boldsymbol{\theta}^{t+1} = \sum_{i \in \mathcal{N}_t} p_i^t \boldsymbol{\theta}_i^{t+1}, \quad \sum_{i \in \mathcal{N}_t} p_i^t = 1. \tag{1}$$

where $p_i^t \geq 0$ denotes the aggregation weight of client i. The aggregated model is then broadcast to the clients for the next communication round. The training process continues until the global model reaches the desired accuracy or the maximum number of communication rounds $T_{\max}$ is reached.

For client-level unlearning, the server maintains a set of available historical model states indexed by $\mathcal{T}_{\text{hist}} \subseteq \{1, \dots, T_{\max}\}$, where each historical state is either directly stored or can be reconstructed from the available training records. These historical states provide candidate model states for rollback when an unlearning request is processed.

Let $\mathcal{Q} \subseteq \mathcal{N}$ denote the set of clients that submit unlearning requests, and let $\overline{\mathcal{Q}} = \mathcal{N} \setminus \mathcal{Q}$ denote the non-requesting clients. Upon receiving unlearning requests, the server must determine whether to retain each requesting client in the current trained model or to remove the client through rollback-based unlearning. Retention allows the server to preserve the client's contribution to the current model, but requires compensation for the continued use of the client's information. For clients selected for removal, the server must select a common historical model state as the rollback point.

In particular, if the selected rollback point occurs after a client submits its unlearning request, the client's information may have continued to contribute to model training during the intervening rounds. The server therefore needs to account for the associated request-to-rollback delay when determining compensation. This leads to a coupled decision problem involving client retention, common rollback, and compensation allocation.

2.2 Unlearning Request

Each requester $i \in \mathcal{Q}$ is characterized by $\tau_i = (q_i, c_i, t_i^{\text{ideal}}, t_i^r)$, where $q_i \in (0, 1]$ is its normalized historical contribution, $c_i \geq 0$ its privacy cost per round, and t_i^{ideal} its contribution-based reference round. The client specifies t_i^r as its preferred historical starting round for unlearning, which also serves as the temporal reference for request-related compensation.

To quantify the historical contribution of client i, we measure the cumulative magnitude of its local model updates as

$$\Psi_i(t) = \sum_{\tau \leq t} \|\Delta_i^\tau\|_2, \tag{2}$$

where Δ_i^τ denotes the local model update of client i at round τ. Given a contribution-preservation ratio $\rho_i \in (0, 1)$, the reference round is defined as

$$t_i^{\text{ideal}} = \max\left\{t \in \mathcal{T}_{\text{hist}} : \Psi_i(t) \leq \rho_i \Psi_i(T_{\max})\right\}. \tag{3}$$

Thus, t_i^{ideal} represents the latest historical point at which the cumulative contribution of client i remains within a prescribed fraction of its final contribution. We assume that q_i and t_i^{ideal} can be audited by the server, whereas c_i is privately known by client i and truthfully reported.

2.3 Client Utility Function

For a requesting client $i \in \mathcal{Q}$, let r_i denote the compensation provided by the server. The compensation requirement is determined by the duration for which the client's information remains used after the unlearning request. Specifically, if the server processes the request using a model state at round T, the request-related compensation requirement is defined as

$$L_i(T) = c_i \max\{T - t_i^r, 0\}. \tag{4}$$

Thus, no compensation is required when $T \leq t_i^r$; otherwise, it increases linearly with the delay $T - t_i^r$. Client i's compensation is capped at $r_i^{\max} = q_i B$, where $B \geq 0$ scales individual caps in proportion to historical contribution rather than imposing a global payment budget.

Therefore, the utility of client i can be defined as

$$U_i(T, r_i) = r_i - L_i(T). \tag{5}$$

In particular, retained clients have utility $U_i^{\text{retain}} = r_i - L_i(T_{\max})$, evaluated at the current model round $T_{\max}$, whereas leaving clients have utility $U_i^{\text{leave}} = r_i - L_i(T)$, evaluated at the common rollback round T.

2.4 Server Utility Function

The server aims to preserve the value of the trained model while processing unlearning requests with appropriate compensation.

Let $\mathcal{S}_{\text{retain}} \subseteq \mathcal{Q}$ denote the set of requesting clients retained in the current model, and let $\mathcal{I}_{\text{leave}} = \mathcal{Q} \setminus \mathcal{S}_{\text{retain}}$ denote the clients selected for unlearning. To characterize the value of a historical global model at round T, We model the historical model value as $G(T) = A(1 - \exp(-kT)) + b$, where $A > 0$ and $k > 0$ are fitting parameters, and b is the baseline model value.

The contribution preserved by the retained clients is measured as

$$W(\mathcal{S}_{\text{retain}}) = \frac{\sum_{i \in \mathcal{S}_{\text{retain}}} q_i}{\sum_{i \in \mathcal{Q}} q_i}. \tag{6}$$

This measure accounts for heterogeneous client contributions and quantifies the relative contribution preserved by retaining a subset of requesting clients. The corresponding contribution-preserved model value is represented by$G(T)W(\mathcal{S}_{\text{retain}})$.

Let $V_{\overline{\mathcal{Q}}}$ denote the baseline value contributed by non-requesting clients, which is independent of the current retention and rollback decisions. Given a model state T, a retained set $\mathcal{S}_{\text{retain}}$, and compensation vector $\boldsymbol{r}$, the server utility is defined as

$$U_s\left(T, \mathcal{S}_{\text{retain}}, \boldsymbol{r}\right) = V_{\overline{\mathcal{Q}}} + \lambda_v |\mathcal{Q}| G(T) W(\mathcal{S}_{\text{retain}}) - \sum_{i \in \mathcal{Q}} r_i, \tag{7}$$

where $\lambda_v > 0$ converts the model-value proxy into the server's utility units. The first term represents the baseline value contributed by non-requesting clients. The second term captures the value associated with the historical model state and the contribution preserved by retaining requesting clients. The final term represents the total compensation paid to the requesting clients.

For a fixed retention or unlearning outcome, the server has no incentive to offer compensation exceeding the client's minimum requirement. Therefore, the compensation variables are subject to the individual rationality constraints and contribution-dependent compensation caps defined above.

2.5 Problem Formulation

Based on the above system model and utility functions, the server jointly determines the retention status of requesting clients, the rollback point for the clients selected for unlearning, and the corresponding compensation.

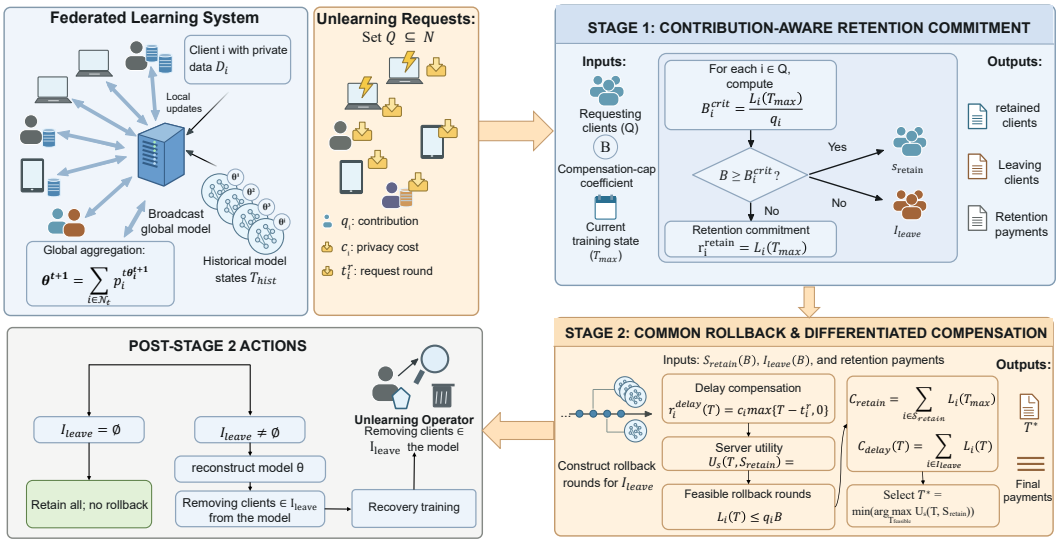

Figure 1: Two-stage workflow of FedCARR.

Let $x_i \in \{0, 1\}$ indicate whether requester i is retained, with $x_i = 1$ for retention and $x_i = 0$ for unlearning. Define $\mathcal{S}_{\text{retain}} = \{i \in \mathcal{Q} : x_i = 1\}$ and $\mathcal{I}_{\text{leave}} = \{i \in \mathcal{Q} : x_i = 0\}$. The server jointly optimizes $(\boldsymbol{x}, T, \boldsymbol{r})$ through the following problem:

$$\begin{aligned} \underset{\boldsymbol{x}, T, \boldsymbol{r}}{\text{maximize}} \quad & U_s\left(T, \mathcal{S}_{\text{retain}}, \boldsymbol{r}\right) \\ \text{subject to} \quad & x_i \in \{0, 1\}, \forall i \in \mathcal{Q}, \\ & T \in \mathcal{T}_{\text{feasible}}, \\ & L_i(T_{\max}) \leq r_i \leq q_i B, \forall i \in \mathcal{S}_{\text{retain}}, \\ & L_i(T) \leq r_i \leq q_i B, \forall i \in \mathcal{I}_{\text{leave}}. \end{aligned} \tag{P1}$$

The constraints specify binary retention decisions, a feasible rollback round, and payments that cover each client's compensation requirement without exceeding its cap, evaluated at $T_{\max}$ for retained clients and T for leaving clients.

Problem (P1) captures the tradeoff between preserving historical model value and compensating requesting clients. Retaining a client avoids removing its contribution from the current model but may require compensation for its continued use after the request. Conversely, removing a client through rollback-based unlearning requires a common historical model state that satisfies the compensation constraints of all leaving clients.

3 Two-Stage Incentive and Rollback Optimization

Problem (P1) couples client retention, rollback selection, andcompensation allocation: retention compensation depends on$T_{\max}$, whereas leaving compensation depends on the common rollback round T. To coordinate retention commitments with rollback feasibility, FedCARR adopts the two-stage mechanism illustrated in figure 1. Stage 1 determines the retention and leaving sets based on individual compensation feasibility and fixes the retention payments. With these commitments fixed, Stage 2 maximizes server utility over feasible historical states to select a common rollback round and assigns leaving clients their minimum individually rational compensation.

3.1 Stage 1: Contribution-Aware Retention Commitment

Stage 1 determines the retention status of each requesting client based on its individual rationality and contribution-dependent compensation cap B. For a requesting client $i \in \mathcal{Q}$, retention requires a compensation that satisfies $L_i(T_{\max}) \leq r_i \leq q_i B$. Since $q_i > 0$, a feasible retention payment exists

if and only if$B \geq B_i^{\text{crit}}$, where

$$B_i^{\text{crit}} = \frac{L_i(T_{\max})}{q_i} = \frac{c_i \max\{T_{\max} - t_i^r, 0\}}{q_i}. \tag{8}$$

The threshold B_i^{crit} is the minimum cap coefficient that makes retention individually rational, normalizing client i's request-related privacy cost by its historical contribution.

Given B, the server determines the retained and leaving sets as:

$$\begin{aligned} \mathcal{S}_{\text{retain}}(B) &= \{i \in \mathcal{Q} : B_i^{\text{crit}} \leq B\}, \\ \mathcal{I}_{\text{leave}}(B) &= \mathcal{Q} \setminus \mathcal{S}_{\text{retain}}(B). \end{aligned} \tag{9}$$

For each $i \in \mathcal{S}_{retain}(B)$, the server commits to the minimum individually rational payment $r_i^{retain} = L_i(T_{max})$, which satisfies the compensation cap and yields zero utility to the retained client. We assume that clients accept retention when indifferent. Since the retention decision is fixed in Stage 1, offering a payment above the minimum would increase the server's expenditure without changing the retention outcome.

Algorithm 1 in Appendix B summarizes the retention classification and payment procedure. It returns the retained set, the leaving set, and the corresponding retention payments. These outputs are passed to Stage 2 and remain fixed throughout the subsequent rollback optimization.

3.2 Stage 2: Common Rollback and Differentiated Compensation

When $\mathcal{I}_{\text{leave}}(B) \neq \varnothing$, Stage 2 determines a common rollback round T for all leaving clients. The retention and leaving sets, together with the retention payments determined in Stage 1, remain fixed throughout this stage.

The server constructs the candidate rollback set $\mathcal{T}_{\text{cand}} = \{t_i^{\text{ideal}} : i \in \mathcal{I}_{\text{leave}}\} \subseteq \mathcal{T}_{\text{hist}}$. Each candidate corresponds to a loadable or reconstructible historical model state. The reference rounds define the candidate set rather than client-specific constraints on the common rollback round.

For each candidate $T \in \mathcal{T}_{\text{cand}}$, leaving client $i \in \mathcal{I}_{\text{leave}}$ receives the minimum individually rational payment $r_i^{\text{delay}}(T) = L_i(T)$. This payment depends on the client's request round t_i^r and privacy-cost coefficient c_i, so payments may differ across clients under the same rollback round. It compensates for request-to-rollback delay; the client is still unlearned.

A candidate rollback round is feasible only if every leaving client can be compensated within its individual cap. The feasible candidate set is therefore defined as

$$\mathcal{T}_{\text{feasible}} = \{T \in \mathcal{T}_{\text{cand}} : L_i(T) \leq q_i B,\ \forall i \in \mathcal{I}_{\text{leave}}\}. \tag{10}$$

Since $L_i(T)$ is non-decreasing in T, if a candidate is feasible, then every earlier candidate in $\mathcal{T}_{\text{cand}}$ is also feasible. This monotonicity allows the feasible candidates to be identified efficiently when they are examined.

Let C_{retain} denote the sum of $L_i(T_{\max})$ over retained clients, fixed in Stage 1, and let $C_{\text{delay}}(T)$ denote the sum of $L_i(T)$ over leaving clients. The resulting Stage 2 server utility is:

$$U_s\left(T, \mathcal{S}_{\text{retain}}\right) = V_{\overline{\mathcal{Q}}} + \lambda_v |\mathcal{Q}| G(T) W\left(\mathcal{S}_{\text{retain}}\right) - C_{\text{retain}} - C_{\text{delay}}(T). \tag{11}$$

With the Stage 1 decisions fixed, a later rollback round generally preserves more training progress and therefore increases the model-value proxy $G(T)$. However, it may also increase the compensation required by the leaving clients. Stage 2 balances these effects by selecting

$$T^\star = \min\left(\underset{T \in \mathcal{T}_{\text{feasible}}}{\arg\max}\ U_s\left(T, \mathcal{S}_{\text{retain}}\right)\right), \tag{12}$$

provided that $\mathcal{T}_{\text{feasible}} \neq \varnothing$. The outer minimum selects the earliest rollback round among all utility-maximizing candidates, thereby avoiding unnecessary retention of client information when multiple candidates achieve the same utility.

Algorithm 2 in Appendix B returns $T^\star$ and the delay payments, or declares infeasibility if no feasible candidate exists. If $\mathcal{I}_{\text{leave}} = \varnothing$, the server retains the current model without unlearning.

After selecting $T^\star$, the server reconstructs $\boldsymbol{\theta}^{T^\star}$ and applies the unlearning operator to $\mathcal{I}_{\text{leave}}$. Recovery training uses $\mathcal{N} \setminus \mathcal{I}_{\text{leave}}$, including retained requesters, while excluding leaving clients from aggregation.

3.3 Theoretical Properties

We establish three key properties of the proposed two-stage mechanism. First, the retention and leaving decisions satisfy individual rationality, and the Stage 1 classification remains stable after the rollback decision. Second, the rollback mechanism admits a feasible utility-maximizing rollback round. Finally, the overall optimization problem $(P1)$ admits an optimal solution. The proofs are provided in Appendix A.

Lemma 3.1 (Individual rationality and stability of retention decisions)**.** *For any compensation-cap coefficient B and any feasible Stage 2 rollback round T, the proposed payment mechanism satisfies the following properties:*

1. *Every client in $\mathcal{S}_{\mathrm{retain}}(B)$ is individually rational under retention.*
2. *Every client in $\mathcal{I}_{\mathrm{leave}}(B)$ is individually rational under leaving.*
3. *The Stage 1 leaving decision is stable throughout Stage 2: after receiving the Stage 2 delay compensation, every client in $\mathcal{I}_{\mathrm{leave}}(B)$ strictly prefers leaving to accepting any retention payment satisfying its compensation cap.*

Lemma 3.2 (Existence and optimality of the rollback decision)**.** *Fix the compensation-cap coefficient, $B \geq 0$, and the Stage 1 sets, S_{retain} and $\mathcal{I}_{\mathrm{leave}} \neq \varnothing$. Then the proposed rollback mechanism satisfies the following properties:*

1. *The feasible rollback set $\mathcal{T}_{\mathrm{feasible}}$ is nonempty and finite.*
2. *A utility-maximizing feasible rollback round exists and can be selected by the mechanism.*

Lemma 3.3 (Existence of an optimal solution to $(P1)$)**.** *For any compensation-cap coefficient $B \geq 0$, problem $(P1)$ admits an optimal solution $(\boldsymbol{x}^*, T^*, \boldsymbol{r}^*)$.*

4 Experiments

4.1 Experimental Setup

Dataset We evaluate our approach on CIFAR-10, CIFAR-100, and Fashion-MNIST under class-wise Dirichlet non-IID partitions. For CIFAR-10 and CIFAR-100, we use ResNet-18 adapted to 32×32 images with random cropping and horizontal flipping for data augmentation. For Fashion-MNIST, we employ a two-layer CNN. All clients participate in each FedAvg round, using local SGD with batch size 128, learning rate 0.01, and momentum 0.9. Global checkpoints are saved every 10 rounds. To initialize unlearning at T^*, we load the latest checkpoint at or before T^* and, if necessary, resume training on all clients' original datasets with the same SGD and FedAvg configurations until T^*.

Compared Methods We compare FedCARR against three baselines: SIFU (Fraboni et al., 2024), FedEraser (Liu et al., 2021), and Full Retrain. Starting from the initialization model at round T^*, FedCARR first performs gradient ascent on the forgotten clients' data, followed by FedAvg-based recovery training over the non-forgotten clients with relearning suppression. Forgotten clients are excluded from the recovery aggregation to prevent their updates from being reintroduced. In contrast, SIFU selects sensitivity-based branch states, FedEraser replays calibrated historical updates, and Full Retrain trains a new model from scratch using only the non-forgotten clients' data.

Evaluation Metrics We evaluate the mechanism in terms of the number of proportion of retained clients, retained contribution weight W, rollback round T^* required for unlearning, total compensation cost and server utility U_s. We measure test accuracy (TA) on the official test set, along with forgotten-client accuracy (FA) and remaining-client accuracy (RA) on the corresponding client's original training samples. Membership leakage is evaluated using a maximum-softmax-confidence attack, reported as effective MIA-AUC.

4.2 Experimental Results

Compensation-cap sensitivity. As shown in figure 2, the leaving-client fraction is nonincreasing in B and eventually reaches zero in all evaluated trajectories. Sever utility, However, is non-monotone.

For CIFAR-100 with $N = 20$, utility peaks near $B = 15$ and later becomes negative, while for Fashion-MNIST with $N = 20$, it declines after $B = 30$. Among the 450 audited points, 55 exhibit no retention, 270 partial retention, and 125 full retention. All 30 trajectories satisfy retained-set monotonicity and reach full retention without encountering an empty feasible candidate set when unlearning is required. These results show that the compensation-cap coefficient directly affects retention, but that a larger cap does not necessarily improve server utility.

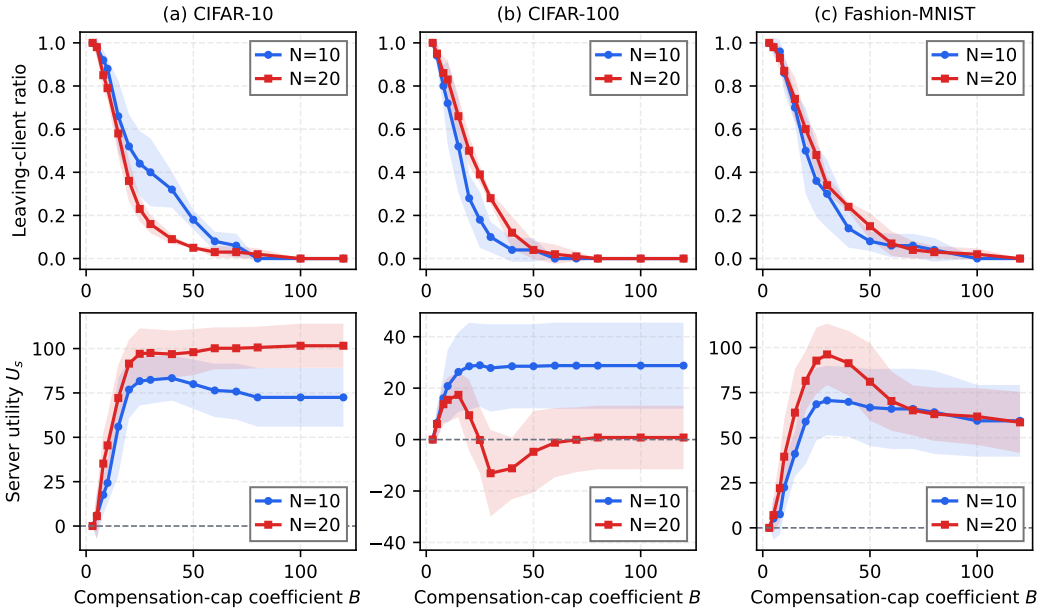

Figure 2: Leaving-client ratio (top) and server utility (bottom) with different values of B.

Retention benefit. We compare retention-enabled and no-retention settings at $B = 15$ for $N \in \{10, 20, 50\}$, with approximately 40% of clients submitting unlearning requests. Each paired comparison uses the same request set and rollback round, isolating the effect of retention while holding the rollback decision fixed. The top row of figure 3 shows positive mean retained contribution weight in all nine configurations, whereas the no-retention baseline retains none by definition. The bottom row shows that the mean contribution-value gain exceeds the incremental compensation cost in every configuration, yielding positive mean net utility gains. These results support the effectiveness of Stage 1 in improving mean server utility under the specified utility model, although the confidence interval for the net utility gain includes zero for CIFAR-100 at $N = 10$. An ablation study evaluating Stage 2 rollback selection is provided in Appendix C.

End-to-end unlearning. We evaluate end-to-end performance over the same five training seeds using the retention and rollback decisions from our mechanism. For a common request set $\mathcal{Q}$, FedCARR retains eligible clients and unlearns $\mathcal{F} = \mathcal{I}_{\text{leave}} \subseteq \mathcal{Q}$, while SIFU and FedEraser process all requests. Full Retrain ($\mathcal{F}$) is the matched reference for this residual task. As shown in table 1 and the convergence plots in Appendix D (figure 4), FedCARR reduces the mean number of clients requiring unlearning from 10 to 4.2, 20 to 15, and 10 to 4.2 on CIFAR-10, CIFAR-100, and Fashion-MNIST, respectively. Mean TA exceeds SIFU's on CIFAR-10 and Fashion-MNIST but is 0.84 percentage points lower on CIFAR-100; it exceeds FedEraser's on all three datasets.

On the residual forgetting task, FedCARR achieves mean FA within 1.5 percentage points of Full Retrain ($\mathcal{F}$) on all datasets, with similar or higher mean TA. Relative to this reference, effective MIA-AUC is closer to 0.5 on CIFAR-100, numerically similar on Fashion-MNIST, and higher on CIFAR-10. Together with the retention and rollback analyses, these results indicate modeled utility gains and empirical forgetting performance close to matched retraining on the residual task, as measured by FA.

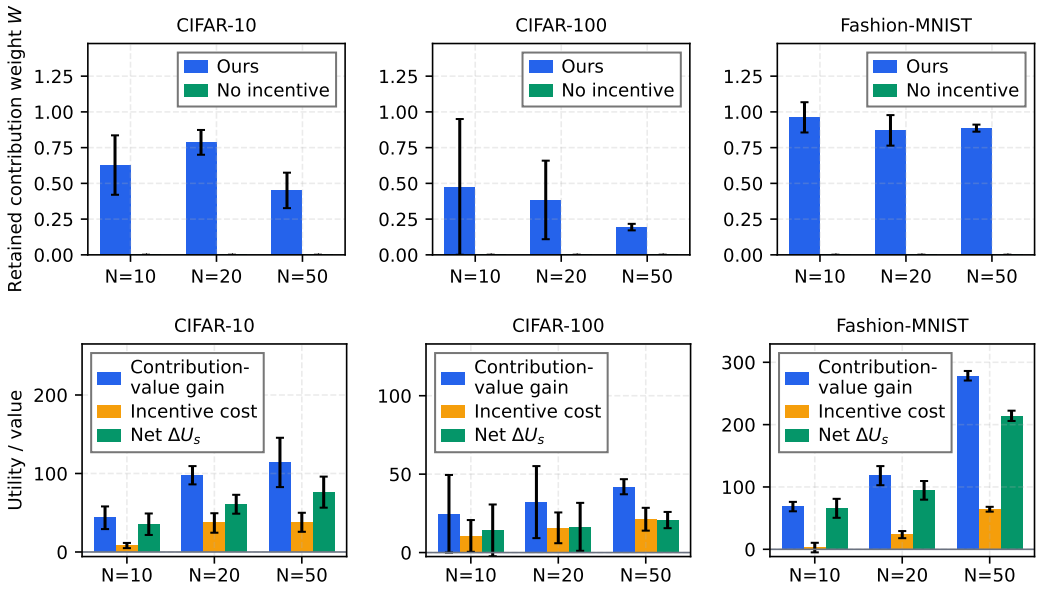

Figure 3: Retention benefits across three datasets. Top: mean retained contribution weight (retention vs. no retention). Bottom: modeled value gain, additional compensation cost, and net server-utility gain over no retention at the same rollback round.

Table 1: End-to-end results over five training seeds (mean $\pm$ sample standard deviation).

Dataset	Method	Forget set	n_{forget}	FA	TA	RA	Effective MIA-AUC
CIFAR-10	FedCARR	$\mathcal{F}$	4.2 ± 0.8	83.65 ± 1.83	86.09 ± 0.75	93.07 ± 0.83	0.587 ± 0.014
	SIFU	$\mathcal{Q}$	10.0 ± 0.0	86.68 ± 1.67	85.23 ± 1.19	92.37 ± 0.98	0.560 ± 0.011
	FedEraser	$\mathcal{Q}$	10.0 ± 0.0	63.25 ± 8.28	65.11 ± 6.83	69.68 ± 5.48	0.563 ± 0.008
	Full Retrain ($\mathcal{F}$)	$\mathcal{F}$	4.2 ± 0.8	85.13 ± 2.29	85.89 ± 0.70	91.57 ± 1.03	0.569 ± 0.024
CIFAR-100	FedCARR	$\mathcal{F}$	15.0 ± 1.0	50.83 ± 0.50	55.09 ± 0.61	90.80 ± 1.16	0.535 ± 0.005
	SIFU	$\mathcal{Q}$	20.0 ± 0.0	60.93 ± 0.37	55.93 ± 0.41	74.24 ± 0.39	0.556 ± 0.002
	FedEraser	$\mathcal{Q}$	20.0 ± 0.0	16.18 ± 5.70	16.61 ± 5.82	18.20 ± 6.48	0.562 ± 0.022
	Full Retrain ($\mathcal{F}$)	$\mathcal{F}$	15.0 ± 1.0	51.67 ± 0.72	53.09 ± 0.46	70.89 ± 1.01	0.562 ± 0.005
Fashion-MNIST	FedCARR	$\mathcal{F}$	4.2 ± 0.8	90.46 ± 2.43	87.61 ± 1.08	88.35 ± 1.72	0.552 ± 0.032
	SIFU	$\mathcal{Q}$	10.0 ± 0.0	79.89 ± 4.07	82.11 ± 1.01	86.55 ± 2.53	0.537 ± 0.036
	FedEraser	$\mathcal{Q}$	10.0 ± 0.0	69.34 ± 9.87	76.81 ± 2.69	85.26 ± 2.57	0.560 ± 0.045
	Full Retrain ($\mathcal{F}$)	$\mathcal{F}$	4.2 ± 0.8	91.27 ± 1.94	87.46 ± 1.05	87.70 ± 1.65	0.560 ± 0.032

TA uses the common official test set. FA and MIA use each method's forgotten clients; RA uses its remaining clients. Effective MIA-AUC is $\max\{\text{AUC}, 1 - \text{AUC}\}$; values closer to 0.5 indicate weaker discrimination by the evaluated attack.

5 Conclusion

In this work, We proposed FedCARR, a feasibility-first two-stage mechanism for contribution-aware batched federated unlearning that jointly considers client retention, contribution-weighted compensation, deletion delay, and executable rollback-state selection under individual compensation caps. Fixing retention decisions before rollback optimization ensures consistency between compensation commitments and subsequent unlearning. We established individual rationality, decision stability, existence and optimality of feasible rollback decisions, and existence of an optimal solution. Experiments on CIFAR-10, CIFAR-100, and Fashion-MNIST demonstrate its effectiveness in balancing model utility, compensation cost, deletion delay, and rollback requirements. Future work will address dynamic deletion requests, more heterogeneous client preferences, and stronger unlearning guarantees in practical deployments.

AI USE STATEMENT

We used large language models to assist with writing and polishing the manuscript, including improving clarity, organization, and LaTeX presentation. We also used these tools for literature retrieval and discovery, including identifying potentially relevant related work. For research ideation and execution, we used them to discuss and refine mathematical statements, organize and draft proofs, and assist in interpreting experimental results. The authors take full responsibility for the final manuscript, including the correctness of its theoretical claims, the accuracy of its citations, and the validity of its experimental results and interpretations.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the source code, experimental configurations, and instructions for reproducing the results reported in this paper in an anonymous repository: `https://anonymous.4open.science/r/FedCARR-6417`. The repository includes details on the required dependencies, data preparation, training, and evaluation procedures.

REFERENCES

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, 2021. doi: 10.1109/SP40001.2021.00019.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pp. 463–480, 2015. doi: 10.1109/SP.2015.35.

Tianshi Che, Yang Zhou, Zijie Zhang, Lingjuan Lyu, Ji Liu, Da Yan, Dejing Dou, and Jun Huan. Fast federated machine unlearning with nonlinear functional theory. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Ningning Ding, Zhenyu Sun, Ermin Wei, and Randall Berry. Incentivized federated learning and unlearning. *IEEE Transactions on Mobile Computing*, 24(9):8794–8810, 2025. doi: 10.1109/TMC.2025.3557857.

Yann Fraboni, Martin Van Waerebeke, Kevin Scaman, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. SIFU: Sequential informed federated unlearning for efficient and provable client unlearning in federated optimization. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 3457–3465. PMLR, 02–04 May 2024. URL `https://proceedings.mlr.press/v238/fraboni24a.html`.

Xiangshan Gao, Xingjun Ma, Jingyi Wang, Youcheng Sun, Bo Li, Shouling Ji, Peng Cheng, and Jiming Chen. Verifi: Towards verifiable federated unlearning. *IEEE Transactions on Dependable and Secure Computing*, 21(6):5720–5736, 2024. doi: 10.1109/TDSC.2024.3382321.

Amirata Ghorbani and James Y. Zou. Data shapley: Equitable valuation of data for machine learning. *ArXiv*, abs/1904.02868, 2019. URL `https://api.semanticscholar.org/CorpusID:102350503`.

Antonio A. Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making ai forget you: data deletion in machine learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Hanlin Gu, Gongxi Zhu, Jie Zhang, Xinyuan Zhao, Yuxing Han, Lixin Fan, and Qiang Yang. Unlearning during learning: an efficient federated machine unlearning method. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, 2024. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/446. URL `https://doi.org/10.24963/ijcai.2024/446`.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael

G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1–2):1–210, June 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL https://doi.org/10.1561/2200000083.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 429–450, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf.

Yijing Lin, Zhipeng Gao, Hongyang Du, Dusit Niyato, Jiawen Kang, and Xiaoyuan Liu. Incentive and dynamic client selection for federated unlearning. In *Proceedings of the ACM Web Conference 2024*, WWW '24, pp. 2936–2944, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701719. doi: 10.1145/3589334.3645462. URL https://doi.org/10.1145/3589334.3645462.

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, 2021. doi: 10.1109/IWQOS52092.2021.9521274.

Zelei Liu, Yuanyuan Chen, Han Yu, Yang Liu, and Lizhen Cui. Gtg-shapley: Efficient and accurate participant contribution evaluation in federated learning. *ACM Trans. Intell. Syst. Technol.*, 13(4), May 2022. ISSN 2157-6904. doi: 10.1145/3501811. URL https://doi.org/10.1145/3501811.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 1273–1282, 2017.

Akarsh K Nair, Sinem Coleri, Jayakrushna Sahoo, Linga Reddy Cenkeramaddi, and Ebin Deni Raj. Incentivized federated learning: A survey. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 9(5):3190–3209, 2025. doi: 10.1109/TETCI.2025.3547609.

Thai-Hung Nguyen, Hong-Phuc Vu, Dung Thuy Nguyen, Tuan Minh Nguyen, Khoa D Doan, and Kok-Seng Wong. Empirical study of federated unlearning: Efficiency and effectiveness. In Berrin Yanıkoğlu and Wray Buntine (eds.), *Proceedings of the 15th Asian Conference on Machine Learning*, volume 222 of *Proceedings of Machine Learning Research*, pp. 959–974. PMLR, 11–14 Nov 2024. URL https://proceedings.mlr.press/v222/nguyen24a.html.

Zibin Pan, Zhichao Wang, Chi Li, Kaiyan Zheng, Boqi Wang, Xiaoying Tang, and Junhua Zhao. Federated unlearning with gradient descent and conflict mitigation. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i19.34181. URL https://doi.org/10.1609/aaai.v39i19.34181.

Nicolò Romandini, Alessio Mora, Carlo Mazzocca, Rebecca Montanari, and Paolo Bellavista. Federated unlearning: A survey on methods, design guidelines, and evaluation metrics. *IEEE Transactions on Neural Networks and Learning Systems*, 36(7):11697–11717, 2025. doi: 10.1109/TNNLS.2024.3478334.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2017. doi: 10.1109/SP.2017.41.

Youming Tao, Cheng-Long Wang, Miao Pan, Dongxiao Yu, Xiuzhen Cheng, and Di Wang. Communication efficient and provable federated unlearning. *Proc. VLDB Endow.*, 17(5):1119–1131, January 2024. ISSN 2150-8097. doi: 10.14778/3641204.3641220. URL https://doi.org/10.14778/3641204.3641220.

Qiong Wu, Xu Chen, Zhi Zhou, and Junshan Zhang. Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring. *IEEE Transactions on Mobile Computing*, 21 (08):2818–2832, 2022. doi: 10.1109/TMC.2020.3045266.

Lefeng Zhang, Tianqing Zhu, Haibin Zhang, Ping Xiong, and Wanlei Zhou. Fedrecovery: Differentially private machine unlearning for federated learning frameworks. *IEEE Transactions on Information Forensics and Security*, 18:4732–4746, 2023. doi: 10.1109/TIFS.2023.3297905.

A Proofs of Theoretical Properties

A.1 Proof of Lemma 3.1

Proof. First, consider any retained client $i \in \mathcal{S}_{\text{retain}}(B)$. By the definition of the retention set,

$$L_i(T_{\max}) \le q_i B.$$

The mechanism assigns this client the retention payment

$$r_i^{\text{retain}} = L_i(T_{\max}),$$

which satisfies the compensation cap. Hence, the client's utility under retention is

$$u_i^{\text{retain}} = r_i^{\text{retain}} - L_i(T_{\max}) = 0.$$

Therefore, retention is individually rational because clients accept retention when indifferent.

Next, consider any leaving client $i \in \mathcal{I}_{\text{leave}}(B)$ and any feasible Stage 2 rollback round T. Feasibility implies

$$L_i(T) \le q_i B.$$

The mechanism assigns the client the delay compensation $r_i^{\text{delay}}(T) = L_i(T)$.

Therefore, the client's utility under leaving is

$$u_i^{\text{leave}}(T) = r_i^{\text{delay}}(T) - L_i(T) = 0.$$

Thus, leaving is individually rational for every feasible Stage 2 rollback round.

Finally, since $i \in \mathcal{I}_{\text{leave}}(B)$, the definition of the Stage 1 leaving set gives

$$L_i(T_{\max}) > q_i B.$$

For any retention payment satisfying the compensation cap, $r_i^{\text{retain}} \le q_i B$, the corresponding retention utility is bounded by

$$u_i^{\text{retain}} = r_i^{\text{retain}} - L_i(T_{\max}) \le q_i B - L_i(T_{\max}) < 0.$$

In contrast, the leaving utility satisfies

$$u_i^{\text{leave}}(T) = 0.$$

Consequently,

$$u_i^{\text{leave}}(T) > u_i^{\text{retain}}, \qquad \forall r_i^{\text{retain}} \le q_i B.$$

Thus, after receiving the Stage 2 delay compensation, every leaving client strictly prefers leaving to accepting any retention payment within its compensation cap. Therefore, the retention and leaving decisions determined in Stage 1 remain stable throughout Stage 2. □

A.2 Proof of Lemma 3.2

Proof. First, consider the feasible rollback set $\mathcal{T}_{\text{feasible}}$. Since $\mathcal{I}_{\text{leave}}$ is nonempty and finite, and each leaving client $i \in \mathcal{I}_{\text{leave}}$ has an associated ideal round t_i^{ideal}, the candidate set

$$\mathcal{T}_{\text{cand}} = \left\{ t_i^{\text{ideal}} : i \in \mathcal{I}_{\text{leave}} \right\} \subseteq \mathcal{T}_{\text{hist}}$$

is finite. The feasible rollback set is defined by

$$\mathcal{T}_{\text{feasible}} = \left\{ T \in \mathcal{T}_{\text{cand}} : L_i(T) \le q_i B \text{ for all } i \in \mathcal{I}_{\text{leave}} \right\}.$$

Consequently,

$$1 \le |\mathcal{T}_{\text{feasible}}| < \infty.$$

Thus, $\mathcal{T}_{\text{feasible}}$ is nonempty and finite.

Next, consider the selection of an optimal rollback round. The mechanism selects

$$T^* = \min \arg\max_{T \in \mathcal{T}_{\text{feasible}}} U_s(T, S_{\text{retain}}).$$

For each candidate $T \in \mathcal{T}_{\text{feasible}}$, the server utility is

$$U_s(T, S_{\text{retain}}) = V_{\overline{\mathcal{Q}}} + \lambda_v |\mathcal{Q}| W(S_{\text{retain}}) G(T) - \sum_{i \in S_{\text{retain}}} L_i(T_{\max}) - \sum_{i \in \mathcal{I}_{\text{leave}}} L_i(T).$$

With the Stage 1 sets and model inputs fixed, the baseline value $V_{\overline{\mathcal{Q}}}$, the coefficient $\lambda_v |\mathcal{Q}| W(S_{\text{retain}})$, and the retention payments are finite and computable. For each candidate T, the model value $G(T)$ and the privacy costs $L_i(T)$ are also finite and computable under the specified models. Since the client sets are finite, both payment sums are finite and computable. Therefore, $U_s(T, S_{\text{retain}})$ can be evaluated for every candidate in $\mathcal{T}_{\text{feasible}}$.

Thus, with exact utility evaluation, the mechanism can select the earliest utility-maximizing feasible rollback round. □

A.3 Proof of Lemma 3.3

Proof. First, we establish the existence of a feasible solution. Under the proposed mechanism, each client chooses retention or leaving according to its compensation cap, yielding a decision vector $\boldsymbol{x}$. For every retained client, the retention rule ensures

$$L_i(T_{\max}) \le q_i B, \qquad \forall i \in \mathcal{S}_{\text{retain}}.$$

By Lemma 3.2, the feasible rollback set is nonempty. Choose any $T \in \mathcal{T}_{\text{feasible}}$. Then

$$L_i(T) \le q_i B, \qquad \forall i \in \mathcal{I}_{\text{leave}}.$$

Setting

$$r_i = \begin{cases} L_i(T_{\max}), & i \in \mathcal{S}_{\text{retain}}, \\ L_i(T), & i \in \mathcal{I}_{\text{leave}}, \end{cases}$$

satisfies both the loss-compensation requirements and the compensation caps. Therefore, $(\boldsymbol{x}, T, \boldsymbol{r})$ is a feasible solution to $(P1)$.

Next, we eliminate the payment variables. For any fixed retention set $\mathcal{S}_{\text{retain}}$ and feasible rollback round T, the server utility decreases as any payment increases. Thus, the optimal payments are

$$r_i^* = \begin{cases} L_i(T_{\max}), & i \in \mathcal{S}_{\text{retain}}, \\ L_i(T), & i \in \mathcal{I}_{\text{leave}}. \end{cases}$$

Substituting these payments into the objective yields

$$U_s(T, \mathcal{S}_{\text{retain}}) = V_{\overline{\mathcal{Q}}} + \lambda_v |\mathcal{Q}| W(\mathcal{S}_{\text{retain}}) G(T) - \sum_{i \in \mathcal{S}_{\text{retain}}} L_i(T_{\max}) - \sum_{i \in \mathcal{Q} \setminus \mathcal{S}_{\text{retain}}} L_i(T).$$

Therefore, after optimizing the payments, the server utility depends only on the retention set $\mathcal{S}_{\text{retain}}$ and the rollback round T.

Algorithm 1 Stage 1: Critical-Compensation Classification and Retention Payments

Require: Requester set $\mathcal{Q}$, $\{q_i, c_i, t_i^r\}_{i\in\mathcal{Q}}$, B and $T_{\max}$
Ensure: $\mathcal{S}_{\text{retain}}, \mathcal{I}_{\text{leave}}, \boldsymbol{r}^{\text{retain}}$
for all $i \in \mathcal{Q}$ **do**
  $B_i^{\text{crit}} \leftarrow L_i(T_{\max})/q_i$
end for
$\mathcal{S}_{\text{retain}} \leftarrow \{i \in \mathcal{Q} \mid B \geq B_i^{\text{crit}}\}$
$\mathcal{I}_{\text{leave}} \leftarrow \mathcal{Q} \setminus \mathcal{S}_{\text{retain}}$
for all $i \in \mathcal{S}_{\text{retain}}$ **do**
  $r_i^{\text{retain}} \leftarrow L_i(T_{\max})$
end for
return $\mathcal{S}_{\text{retain}}, \mathcal{I}_{\text{leave}}, \boldsymbol{r}^{\text{retain}}$

Algorithm 2 Rollback Search and Compensation

Require: Stage 1 outputs, client parameters, and server parameters
Ensure: $(T^\star, \boldsymbol{r}^{\text{delay}})$ or INFEASIBLE
if $\mathcal{I}_{\text{leave}} = \varnothing$ **then return** $(T_{\max}, \mathbf{0})$
$\mathcal{T}_{\text{feasible}} \leftarrow \{t_i^{\text{ideal}} : i \in \mathcal{I}_{\text{leave}}\} \cap \{T : L_j(T) \leq q_j B,\ \forall j \in \mathcal{I}_{\text{leave}}\}$
if $\mathcal{T}_{\text{feasible}} = \varnothing$ **then return** INFEASIBLE
$(T^\star, u^\star) \leftarrow (\varnothing, -\infty)$
for all $T \in \mathcal{T}_{\text{feasible}}$ in ascending order **do**
  $u \leftarrow U_s(T, \mathcal{S}_{\text{retain}})$
  if $u > u^\star$ **then**
    $(T^\star, u^\star) \leftarrow (T, u)$
  end if
end for
$r_i^{\text{delay}} \leftarrow L_i(T^\star), \forall i \in \mathcal{I}_{\text{leave}}$
return $(T^\star, \boldsymbol{r}^{\text{delay}})$

Finally, since the client set $\mathcal{Q}$ is finite, there are finitely many possible retention sets $\mathcal{S}_{\text{retain}} \subseteq \mathcal{Q}$. For each retention set, the feasible rollback candidate set is also finite. Therefore, there are finitely many feasible pairs $(T, \mathcal{S}_{\text{retain}})$. The feasible solution constructed above ensures that the collection of these pairs is nonempty.

Since $U_s(T, \mathcal{S}_{\text{retain}})$ is finite at every feasible pair, it attains its maximum over this nonempty finite collection. Hence, there exists a feasible pair $(T^*, \mathcal{S}^*_{\text{retain}})$ such that

$$U_s(T^*, \mathcal{S}^*_{\text{retain}}) \geq U_s(T, \mathcal{S}_{\text{retain}})$$

for every feasible pair $(T, \mathcal{S}_{\text{retain}})$. □

B Algorithm of the Two-Stage Mechanism

This appendix presents the detailed algorithms for the two-stage mechanism introduced in Sections 3.1 and 3.2.

B.1 Stage 1: Retention Classification and Payments

Algorithm 1 summarizes the retention classification and compensation procedure in Stage 1.

B.2 Stage 2: Rollback Selection and Delay Compensation

Algorithm 2 presents the rollback selection and delay compensation procedure in Stage 2.

Table 2: Second-stage rollback ablation

Dataset	Stage-II policy	T	Total cost	U_s
CIFAR-10	Earliest feasible	10.0 ± 0.0	29.96 ± 4.16	37.44 ± 7.55
	Latest feasible	58.8 ± 7.3	46.46 ± 9.36	60.54 ± 15.71
	Optimized T^*	43.0 ± 1.4	30.22 ± 4.33	73.78 ± 10.75
CIFAR-100	Earliest feasible	10.0 ± 0.0	30.96 ± 9.83	-7.53 ± 6.11
	Latest feasible	98.4 ± 3.3	67.55 ± 11.06	-2.18 ± 15.82
	Optimized T^*	78.4 ± 0.9	30.96 ± 9.83	31.96 ± 7.30
Fashion-MNIST	Earliest feasible	10.0 ± 0.0	19.23 ± 2.35	82.83 ± 14.59
	Latest feasible	28.2 ± 3.9	32.35 ± 5.42	75.12 ± 18.00
	Optimized T^*	16.0 ± 1.0	19.38 ± 2.19	86.56 ± 13.41

C ROLLBACK ABLATION

We evaluates the utility effect of rollback selection while holding the Stage 1 retained and leaving sets and retention payments fixed within each run. All three policies use the same feasible candidate set $\mathcal{T}_{\text{feasible}}$, according to the leaving clients' compensation constraints. The policies select the earliest feasible round, the latest feasible round, or the round maximizing U_s. The earliest policy minimizes rollback-dependent compensation, whereas the latest policy maximizes the historical model-value proxy $G(T)$ over the feasible candidates. The optimized policy balances model value against compensation cost. Table 2 summarizes the results over five training seeds per dataset. Run-level results show that the optimized policy selects a round strictly between the earliest and latest feasible candidates in all 15 runs. Relative to the latest feasible policy, the mean paired utility gains are 13.24 ± 8.79, 34.15 ± 13.62, and 11.44 ± 6.56 on CIFAR-10, CIFAR-100, and Fashion-MNIST, respectively. Relative to the earliest feasible policy, the corresponding gains are 36.34 ± 4.30, 39.49 ± 10.77, and 3.73 ± 3.20. The optimized policy yields strictly higher utility than both alternatives in every run. These results quantify the benefit of balancing the lower model value of earlier rounds against the higher delay compensation of later rounds, supporting the value of Stage 2 rollback optimization under the specified utility model.

D END-TO-END UNLEARNING CONVERGENCE

Figure 4 presents the end-to-end unlearning convergence curves on Fashion-MNIST, CIFAR-10, and CIFAR-100. The columns report forgotten-client accuracy (FA), test accuracy (TA), and remaining-client accuracy (RA).

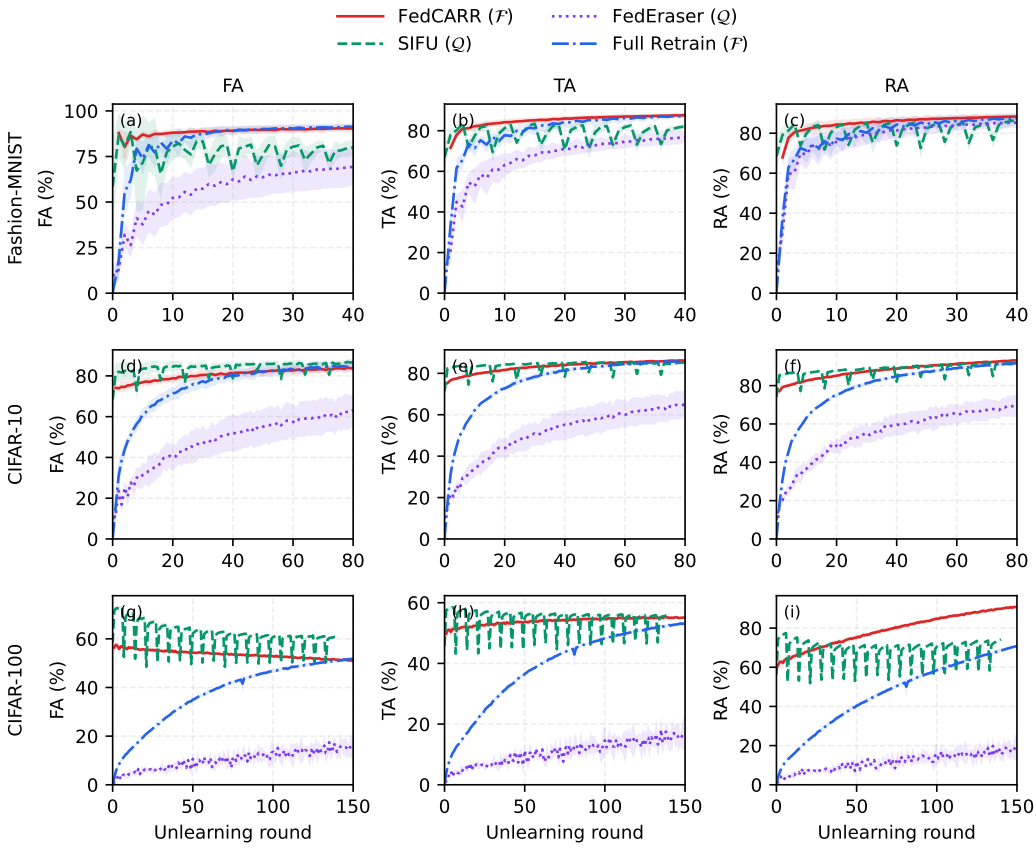

Figure 4: End-to-end unlearning convergence on Fashion-MNIST, CIFAR-10, and CIFAR-100. Columns report forgotten-client accuracy (FA), test accuracy (TA), and remaining-client accuracy (RA).