FedShard: Federated Unlearning with Efficiency Fairness and Performance Fairness

Siyuan Wen[1], Meng Zhang[2], Yang Yang[3], Ningning Ding[*1]
[1]Hong Kong University of Science and Technology (Guangzhou)
[2]Zhejiang University
[3]Hong Kong University of Science and Technology
swen211@connect.hkust-gz.edu.cn, mengzhang@intl.zju.edu.cn, yangyangsh@ust.hk, ningningding@hkust-gz.edu.cn

Abstract

To protect clients' right to be forgotten in federated learning, federated unlearning aims to remove the data contribution of leaving clients from the global learned model. While current studies mainly focused on enhancing unlearning efficiency and effectiveness, the crucial aspects of efficiency fairness and performance fairness among decentralized clients during unlearning have remained largely unexplored. In this study, we introduce FedShard, the first federated unlearning algorithm designed to concurrently guarantee both efficiency fairness and performance fairness. FedShard adaptively addresses the challenges introduced by dilemmas among convergence, unlearning efficiency, and unlearning fairness. Furthermore, we propose two novel metrics to quantitatively assess the fairness of unlearning algorithms, which we prove to satisfy well-known properties in other existing fairness measurements. Our theoretical analysis and numerical evaluation validate FedShard's fairness in terms of both unlearning performance and efficiency. We demonstrate that FedShard mitigates unfairness risks such as cascaded leaving and poisoning attacks and realizes more balanced unlearning costs among clients. Experimental results indicate that FedShard accelerates the data unlearning process 1.3-6.2 times faster than retraining from scratch and 4.9 times faster than the state-of-the-art exact unlearning methods.

Extended version —
https://www.arxiv.org/abs/2508.09866

1 Introduction

Federated learning (FL) is a prominent paradigm for collaborative machine learning on distributed data, wherein decentralized clients train a shared global model under the coordination of a central server (McMahan et al. 2017). To protect the "right to be forgotten" (RTBF) for clients who leave the FL system, federated unlearning (FU) has emerged as an essential method for removing the contributions of leaving data from the global model (Voigt and Von dem Bussche 2017; Harding et al. 2019; Truex et al. 2019; Mothukuri et al. 2021). However, prior work in FU mainly focuses on the efficiency and effectiveness of the unlearning process (Nguyen et al. 2022; Zhang et al. 2023a). We argue that the fairness of federated unlearning—with respect to both computational costs and performance impacts—is a critical yet overlooked aspect, especially for system sustainability and security.

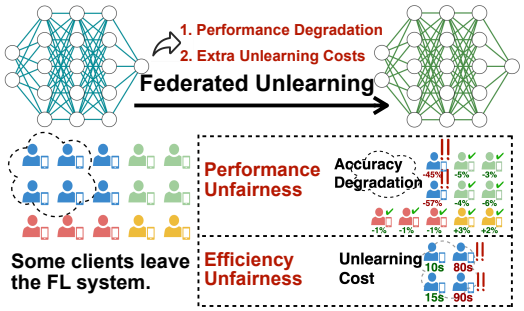
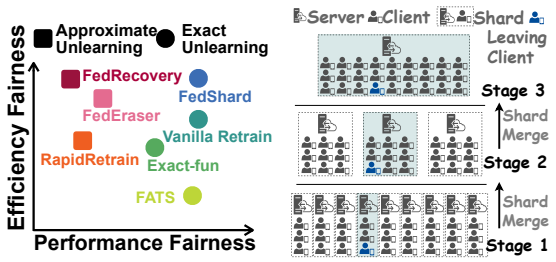


Figure 1: An illustration of performance unfairness and efficiency unfairness in federated unlearning. Red exclamation marks indicate that clients encounter unfairness.

As illustrated in Figure 1, literature usually overlooked two types of issues, performance fairness (**P.F.**) and efficiency fairness (**E.F.**), which are crucial and practically important. P.F. aims to achieve a fair performance degradation caused by unlearning among clients, guaranteeing both remaining clients' rights to good model performance, and leaving clients' RTBF. Without P.F., unlearning leaving clients may cause an excessive accuracy decrease on specific data, thereby encouraging other remaining clients (especially those with similar data) to leave. Such cascaded leaving can severely compromise the sustainability of an FL system. Moreover, unfair unlearning can be further exploited for data poisoning attacks (Tolpegin et al. 2020). By deliberate unlearning requests, malicious clients can launch data poisoning attacks on specific clients with similar data and compromise the security of the FL system. E.F. demands a fair unlearning cost for every client when it leaves. Unlearning methods without E.F. may harm the system sustainability by discouraging clients from participating in the system, as they are concerned about unfairly large leaving costs.

Current FU methods are unsatisfactory in terms of performance fairness and efficiency fairness. Specifically, related work can be classified into two categories: calibration-based approximate unlearning and retraining-based exact unlearn-

*Corresponding author.



(a) (b)

Figure 2: Left: A conceptual illustration showing that only FedShard simultaneously achieves both performance fairness and efficiency fairness. Right: An example of the FedShard architecture with a merging rate of 3. Clients are organized in a tree-like structure. When clients request unlearning (indicated in blue), only the shards on the direct path to the root (dark background) require retraining.

ing. As illustrated in Figure 2(a), calibration unlearning (such as FedRecover (Zhang et al. 2023b), FedEraser (Liu et al. 2021) and RapidRetrain (Liu et al. 2022)) often introduces extra performance degradation on remaining clients or insufficient unlearning on leaving clients, which compromises the P.F. Exact retraining (such as FATS (Tao et al. 2024)) can easily introduce significant variance in unlearning efficiency, thereby violating E.F..

To address these shortcomings, we propose FedShard, a novel sharded federated unlearning framework that prioritizes both fairness and efficiency to achieve effective federated unlearning. As illustrated in Figure 2(b), the FedShard framework operates in two repeating phases: isolated federated training within distinct client shards, and a progressive merging of these shards until all clients are integrated into a single shard. This "isolated yet integrate" design inherently minimizes the impact of a client's leaving and facilitates highly efficient unlearning by leveraging cached intermediate model parameters.

However, designing such a sharded framework presents non-trivial challenges among convergence, unlearning efficiency and unlearning fairness. First, determining appropriate stage-wise training rounds is critical. Excessive training within a shard can cause its model to overfit to local data and diverge from the global optimum, whereas insufficient training results in poor convergence. Second, it presents another challenge to determine which clients should be merged into which new shards in the next stage. While a strategy that minimizes training time disparities across clients would promote efficiency fairness, it may compromise the convergence speed and final performance of the global model. To navigate these trade-offs, we develop two adaptive algorithms that dynamically determine the appropriate training rounds and shard merging configurations, ensuring both fairness and overall efficiency.

Our primary contributions are summarized as follows:

- *Novel sharded framework with adaptive algorithms.* We introduce shards into federated learning and unlearning, providing a novel framework that is both isolated and integrated. We further develop adaptive algorithms to tackle the challenges of stage-wise training rounds and shard merging strategy for sharded training.
- *Fair and efficient federated unlearning.* We propose the first efficient federated unlearning framework that satisfies both performance fairness and efficiency fairness. We also prove that FedShard unlearns at least $P/2$ times faster than retraining from scratch, where P is the number of stages. Furthermore, FedShard does not restrict to a single unlearning request processing but allows multiple simultaneous unlearning requests, which provides additional efficiency improvement.
- *Fairness metrics for federated unlearning.* We identify the performance fairness and efficiency fairness requirements for federated unlearning and introduce novel metrics to measure them for different FU methods. We further validate them by theoretical analysis and numerical evaluation, showing that they are practical and effective.
- *Evaluation contribution.* Our evaluation demonstrates the fairness weakness of existing state-of-the-art works and confirms FedShard's ability to address fairness issues, including cascaded leaving, data poisoning attacks, and unfair-distributed unlearning costs. Our experiment results also show that FedShard unlearns 1.3-6.2 times faster than retraining and up to 4.9 times faster than the state-of-the-art exact unlearning methods.

2 Related Work

Related research of federated unlearning can be categorized into two primary approaches: calibration-based approximate unlearning and exact retraining.[1]

Parameter calibration methods scale down leaving clients' gradients with the Hessian matrix (Liu et al. 2022), cached intermediate information (Liu et al. 2021), or gradient ascent (Tarun et al. 2024). They achieve efficient federated unlearning, while such methods can easily cause extra performance degradation (Zhang et al. 2023b) on remaining clients and insufficient unlearning on leaving clients, violating P.F. Our proposed method will avoid such issues and achieve performance fairness while maintaining efficiency.

Exact retraining methods achieve quite effective unlearning, *e.g.*, FATS (Tao et al. 2024) and Exact-fun (Xiong et al. 2023). They utilize cached information to alleviate the heavy overhead caused by vanilla retraining. Unfortunately, these methods usually lead to disparities in unlearning costs, compromising efficiency fairness. For example, Tao et al. allow clients to join at different rounds for fast federated unlearning. When the later-joined client is leaving, earlier rounds without its participation can be bypassed from unlearning, thereby achieving efficient unlearning. However, earlier-joined clients will burden extra overheads, leading to unfair unlearning costs. Our FedShard ensures that every client has a closer unlearning cost related only to its own data contribution, thereby ensuring a fair unlearning cost while maintaining efficient retraining.

[1]Further literature related to FL fairness and shard methods can be found in Appendix A due to space limit.

3 FedShard Design

In this section, we show the architecture and mechanics of FedShard in detail. We first describe the sharded federated learning and unlearning process in §3.1. We then present the two core adaptive algorithms that enable the fairness and efficiency: a cluster-based shard merging strategy (§3.2) and a variance-aware training round allocation scheme (§3.3).

3.1 Sharded Federated Learning and Unlearning

FedShard efficiently achieves exact unlearning by organizing clients into a hierarchical and multi-stage structure. As illustrated in Figure 3, this structure can be conceptualized as a tree, where clients are the leaves, shards are the nodes, and the training stages represent the levels of the tree. For the convenience of understanding, we formalize the items in FedShard with the following notations. We provide the complete symbol list in Appendix B for quick viewing.

Let $\mathcal{C} = \{c_1, \ldots, c_K\}$ be the set of all clients. An entire FedShard process consists of $P \triangleq \lceil K/R \rceil$ stages, where K is the total number of clients and R is a user-defined merging rate to decide how many shards are merged into one super-shard at the next stage. At each stage $p \in \{1, \ldots, P\}$, there are N_p shards, denoted by $\mathcal{S}^p = \{S_1^p, \ldots, S_{N_p}^p\}$. Each shard S_s^p independently performs federated learning and unlearning in parallel, and we denote its model parameters as θ_s^p. Specially, $\theta_s^{p,0}$ refers to the initial parameters for shard S_s^p. The aggregation weight for S_s^p is denoted as w_s^p. We define the contribution factor α_c, a key concept to measure a client's impact on the global model, as the angle between the local update vector $\Delta\theta_c$ and the global update vector $\Delta\theta$:[2]

$$\alpha_c = \arccos \frac{\langle \Delta\theta, \Delta\theta_c \rangle}{\|\Delta\theta\|_2 \|\Delta\theta_c\|_2}. \quad (1)$$

A smaller angle α_c implies a larger contribution to the global model. We use α_s^p to denote the average of α_cs in S_s^p.

The FedShard training workflow proceeds stage-by-stage. Each stage p involves the following four phases: (1) *shard generation*, (2) *shard initialization*, (3) *parallel federated training*, and (4) *state caching*. For the first two phases, the server merges shards from stage $p-1$ to generate new shards using algorithm $\mathcal{A}_1$ (introduced later in §3.2). Each new shard is initialized and the number of training rounds for all shards is determined by algorithm $\mathcal{A}_2$ (introduced later in §3.3). For the last two phases, the server performs federated training within new shards and caches the model updates. We provide the detailed workflow, pseudocode and space complexity analysis in Appendix D.1 due to the space limits. This process repeats, with the number of shards decreasing at each stage, until a single global shard containing all clients is formed in the final stage P.

This isolated and hierarchical training process is the key to efficient unlearning. When a client requests to be forgotten, retraining is only required for the shards that lie on its direct path to the root of the tree. All other shards can reuse their cached model states, significantly reducing the computational cost compared to retraining from scratch. For example, as shown in Figure 3, if client c_{25} requests unlearning, only shards S_4^1, S_2^2, and S_1^3 need to be retrained. The unlearning procedure is detailed in Algorithm 1, and its efficiency will be formally analyzed in §4.

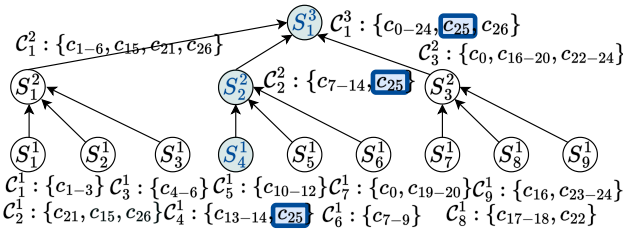


Figure 3: Illustration of the FedShard learning and unlearning process of the example in Figure 2. Each node represents a shard, and each layer presents a training stage. Arrows indicate the merging of shards between stages. When client c_{25} requests unlearning, only the shards on its hierarchical path (dark background) require retraining.

Algorithm 1: FedShard Federated Unlearning Algorithm

Input: Leaving client c_l, cached information during federated training FLCache
Output: Unlearned model θ_u^*

AShards = $\emptyset$ // *Affected shards*
P $\leftarrow$ FLCache
for p = 1, 2, . . . , P **do**
  S_1^p, S_2^p, . . . , $S_{N_p}^p$ $\leftarrow$ FLCache
  for s = 1, 2, . . . , N_p, AShards.append(S_s^p) **if** c_l **in** S_s^p
end for
for p = 1, 2, . . . , P **do**
  // Phase 1&2: Load shard S_s^p, its training round T_s^p, and its initial model $\theta_s^{p,0}$ from FLCache
  S_s^p, T_s^p, $\theta_s^{p,0}$ $\leftarrow$ FLCache, s = 1, 2, . . . , N_p
  // Phase 3: Parallel federated training
  for each S_s^p **in** stage p, s= 1, 2, . . . , N_p **do**
    if S_s^p **in** AShards **then**
      θ_s^p=**FederatedTraining**($\theta_s^{p,0}$, [clients$\in S_s^p$], T_s^p)
    else
      $\theta_s^p \leftarrow$ FLCache
    end if
  end for
  // Phase 4: Cache the retrained shards
  FLCache $\leftarrow$ S_s^p, θ_s^p, α_s^p, w_s^p **if** S_s^p **in** AShards
end for
Return: $\theta_u^* = \theta_1^P$ *//Final model from the last stage*

3.2 Shard Merging Algorithm $\mathcal{A}_1$

The strategy for merging shards between stages is critical yet challenging. A naive approach presents a dilemma: merging shards with divergent model update directions (*e.g.*, S_9^1, S_6^1, S_1^1 in Figure 4) can hinder convergence[3], while

[2] Our framework is also applicable to other metrics for measuring the contribution factor, such as the loss function value, shapley value, etc. For the convenience of understanding, we use parameter similarity, the most intuitive one, in the main paper and provide discussion on other contribution factors in Appendix C.

[3] As the experiment we later show in §6.

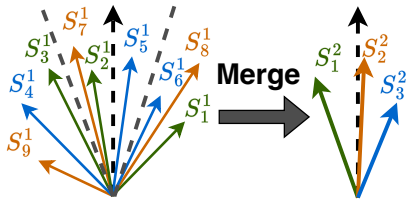


Figure 4: Example of the shard merging strategy. To form a new shard for the next stage, FedShard combines shards with diverse update directions (represented by arrows) to promote stable convergence.

merging shards with highly similar update directions (*e.g.*, S_9^1, S_4^1, S_3^1 in Figure 4) can create an unrepresentative super-shard that biases the global model. This bias would lead to inequitable unlearning costs, thereby compromising efficiency fairness.

Our merging algorithm $\mathcal{A}_1$ navigates this trade-off by promoting directional diversity within newly formed shards. The core idea is to first cluster the existing shards based on their model update directions (relative to the global average) and then to form new shards by selecting constituents from different clusters. For instance, in Figure 4, the new shard S_2^2 is intentionally formed by merging S_7^1 (with an average update direction), S_8^1 (with a large positive angle), and S_9^1 (with a large negative angle). This balanced approach ensures each new shard to be representative of the broader client population, leading to more stable convergence and a fairer distribution of learning contributions. Further details of the algorithm are provided in Appendix D.2.

3.3 Training Round Allocation Algorithm $\mathcal{A}_2$

Determining the number of training rounds per shard at each stage is another critical challenge. Too many rounds can cause a shard's model to overfit on its local data, making it difficult to merge effectively and wasting computation. Conversely, too few rounds can lead to under-training, shifting an excessive convergence burden to the final stage, which in turn increases the unlearning cost for all clients.

Our allocation algorithm $\mathcal{A}_2$ addresses this by dynamically assigning training rounds based on the heterogeneity of clients within each shard. The intuition is that shards with more homogeneous clients (i.e., lower contribution variance) can be trained for more rounds without overfitting, while heterogeneous shards require fewer rounds to prevent their models from diverging. Specifically, after determining a valid range of training rounds $[T_0^*, T_1^*]$, the number of rounds T_s^p for shard s in stage p is set using the following inverse relationship:

$$\frac{T_s^p - T_0^*}{T_1^* - T_0^*} = \left[\frac{\sigma^2(\alpha_s^p) - \sigma^2(\alpha)_{min}}{\sigma^2(\alpha)_{max} - \sigma^2(\alpha)_{min}}\right]^{-1}, \tag{2}$$

where $\sigma^2(\alpha_s^p)$ is the variance of client contributions within shard s, and $\sigma^2(\alpha)_{min/max}$ are the minimum/maximum variances across all shards in the stage. This policy promotes E.F. by avoiding extreme numbers of training rounds. More details of $\mathcal{A}_2$ is available in Appendix D.3.

4 Analysis

In this section, we provide a theoretical foundation for FedShard. We first prove that FedShard achieves exact unlearning, equivalent to retraining from scratch (§4.1). We then analyze its computational efficiency for both single-client and multi-client unlearning requests, demonstrating significant speedups over baseline methods (§4.2 and §4.3).[4]

4.1 Effectiveness of Unlearning

We hold the following proposition that is widely accepted and leveraged in the exact unlearning literature (Bourtoule et al. 2021; Tao et al. 2024; Xiong et al. 2023):

Proposition 1. *For any training process $\theta^* = \mathcal{A}_{fl}(D_0; \theta^0)$ with training dataset D_0 and initial model θ^0 and $\forall D_l \in \mathcal{D}$,*

$$\textit{if } (D_l \not\subseteq D_0) \wedge (\theta^0 \textit{ not affected by } D_l),$$

then we say θ^* is not affected by D_l and using cached updates θ^* for unlearning D_l is equivalent to retraining from scratch on $D_0 \backslash D_l$.

With this proposition, we prove by induction to show that FedShard unlearning, which only trains the one shard containing the leaving client every stage, is as effective as retraining from scratch.

4.2 Efficiency of Single-Client Unlearning

To build intuition, we first consider a simplified case with a constant number of training rounds (T_0) for all shards.

Proposition 2. *FedShard FU is r_1 times faster compared with retraining from scratch, where*

$$r_1 = \frac{R-1}{R}\frac{K}{K-1}P. \tag{3}$$

With $R \in \mathbb{Z}^+$, this proposition shows that FedShard unlearns at least $P/2\times$ (usually $P\times$) faster than retraining the model from scratch through FedAvg, *i.e.*, vanilla retraining.

Next, we consider the general case where training rounds T_s^p can vary across shards and stages, as determined by our algorithm $\mathcal{A}_2$. Let T_{max} and T_{min} be the maximum and minimum rounds assigned to any shard.

Proposition 3 (Efficiency in the General Case)**.** *In the general case with variable training rounds, the speedup factor $\hat{r_1}$ over vanilla retraining is bounded as follows:*

$$r_1\frac{T_0}{T_{max}} \le \hat{r_1} \le r_1\frac{T_0}{T_{min}}. \tag{4}$$

This proposition reveals that the bound of training rounds (T_{max} and T_{min}) directly impacts efficiency. It provides the theoretical justification to design Algorithm $\mathcal{A}_2$, which seeks to maintain a narrow bound for high efficiency and fairness.

4.3 Multiple Unlearning and Time Complexity

Next we show that unlearning multiple clients together is more efficient in FedShard. Considering m clients that leave at the same time, we denote p' as the stage after which all shards contain leaving clients. Then we have:

[4]All proofs are provided in Appendix E due to space limits.

Proposition 4. *Compared with unlearning one by one, performing multiple unlearning requests in FedShard achieves at least r_2^- times and at most r_2^+ times faster, where:*

$$r_2^+ = m, \quad r_2^- = \frac{R}{R-1}\frac{K-1}{K}\frac{m}{(P-p')} \geq 1. \quad (5)$$

This proposition shows that multiple unlearning in FedShard is even more efficient compared to FedShard unlearning one by one. Based on Propositions 2, 3 and 4, we have:

Corollary 1. *FedShard unlearning requires $\mathcal{O}(K)$ time for one client, and $\mathcal{O}(K \lg m)$ time for m clients, where K is the number of clients.*

5 Fairness Metrics

To quantitatively evaluate the fairness of FU algorithms, we introduce novel metrics for P.F. (§5.1) and E.F. (§5.2). We begin by additionally formalizing necessary notations.

Following an unlearning request, the set of clients $\mathcal{C}$ is partitioned into leaving clients $\mathcal{C}_L$ and remaining clients $\mathcal{C}_R$. We use $c_l \in \mathcal{C}_L$ and $c_r \in \mathcal{C}_R$ to specify a leaving client and a remaining client, respectively, *e.g.*, c_{l1}, c_{l2}, and c_{r1}. We denote the local dataset of client c as D_c, and denote the model's accuracy on a test set D_c is given by $\mathcal{Y}(D_c)$. Based on α_c defined in Equation (1), $\mathrm{Dis}(c_1, c_2) \triangleq |\alpha_{c_1} - \alpha_{c_2}|$ can represent the data distance between any two clients α_{c_1} and α_{c_2}, where a smaller value indicates greater similarity in their data's impact on the model.

5.1 Performance Fairness Metric M_p

We define M_p as the P.F. metric for federated unlearning:

$$M_p \triangleq \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} f_\oplus(\Delta\mathcal{Y}(D_c), \min_{c_r\in\mathcal{C}_R}\mathrm{Dis}(c, c_r)). \quad (6)$$

M_p captures whether the clients' local performance degradation caused by unlearning ($\Delta\mathcal{Y}(D_c)$) is unfair considering clients' data uniqueness. We assess the uniqueness by $\min_{c_r\in\mathcal{C}_R}\mathrm{Dis}(c, c_r)$, the minimal data distance between the leaving client c and any client c_r remaining in the system. Specifically, $f_\oplus(x, y) = (\tilde{x} + \tilde{y}) \cdot (\tilde{x}^{-1} + \tilde{y}^{-1})$ is a convex function and reaches its minimum at $\tilde{x} = \tilde{y}$. Operation $\tilde{x}$ normalizes x to (0, 1], *i.e.*, $\tilde{x}{=}\epsilon + (x - x_{min})/(x_{max} - x_{min})$, where ϵ is a small constant to avoid division by zero. A lower M_p score indicates superior performance fairness.

We give two examples to illustrate the rationales behind M_p. First, suppose a leaving client c_{l1} that has unique data in the system (*i.e.*, $\min_{c_r\in\mathcal{C}_R}\mathrm{Dis}(c_r, c_{l1})$ is large). A fair FU algorithm will effectively forget data of client c_{l1} (*i.e.*, $\Delta\mathcal{Y}(D_{c_{l1}})$ is also large) and receive a lower M_p score, indicating being performance-fair. Second, suppose that leaving client c_{l2} and a remaining client c_{r1} have the same data (*i.e.*, $\min_{c_r\in\mathcal{C}_R}\mathrm{Dis}(c_r, c_{l2}) = \mathrm{Dis}(c_{r1}, c_{l2})$ is small). It's unfair for client c_{r1} if the unlearned model has an excessively large performance drop $\Delta\mathcal{Y}(D_{c_{r1}})$ on its dataset, as client c_{r1}'s data is still in the system. Therefore, such an unlearning algorithm receives a high M_p score, indicating poor P.F.. As we shall see in §6, our experiments validate the effectiveness of our proposed metrics in measuring fairness.

5.2 Efficiency Fairness Metric M_e

We define the E.F. metric M_e as the contribution-weighted variance of unlearning costs:

$$M_e \triangleq \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}\frac{(Z_{avg} - Z_c)^2}{|\alpha_c|}. \quad (7)$$

E.F. captures whether clients' unlearning cost (*i.e.*, computing time) Z_c is aligned with their impact (*i.e.*, contribution $|\alpha_c|$) to the global model. Specifically, the numerator represents the variance of federated unlearning costs between one client and the average Z_{avg}. The denominator $|\alpha_c|$ normalizes the cost by the client's impact. A lower M_e score indicates superior efficiency fairness.

We also give an example to illustrate the rationales behind M_e. Suppose that there are two clients c_1 and c_2 in the system who made similar contributions to the global model (i.e., $\alpha_{c_1} \approx \alpha_{c_2}$). An unfair FU algorithm requires vastly different costs to unlearn them ($Z_{c_1} \gg Z_{c_2}$), resulting in a large cost variance and thus a very high M_e score.

5.3 Fairness Properties

It is important to note that the absolute values of M_p and M_e are only meaningful when comparing different FU methods under identical settings. Furthermore, M_p and M_e are designed to be orthogonal, capturing independent dimensions of fairness related to performance and cost, respectively.

Our proposed metrics also satisfy several desirable properties established in economic and computer science literature, including envy-freeness (FP.1), Pareto optimality (FP.2), fairness axiom (FP.3), and reduction (FP.4) (Li and Xue 2013a; Procaccia 2013; Feldman and Kirman 1974; Luc 2008; Lan et al. 2010; Li and Xue 2013b). FU algorithms minimizing M_p satisfy FP.1, FP.2, and FP.3; FU algorithms minimizing M_e satisfy FP.3 and FP.4. These properties ensure the robustness and theoretical grounding of our metrics. We provide formal definitions and proofs that FedShard satisfies these properties in Appendix F.

6 Experimental Results

In this section, we conduct comprehensive experiments on FedShard (FS) regarding the following evaluation questions. In §6.1, we answer **Q1**: How does the sharded framework affect federated training performance? In §6.2, we answer **Q2**: How fast and efficiency-fair is FedShard's FU mechanism? In §6.3, we answer **Q3**: How effective and performance-fair is FedShard's FU mechanism? For each subsection, we first explain the evaluation setting and metrics before presenting the results. We compare FS against state-of-the-art baselines spanning both calibration-based approximate methods—RapidRetrain (RR), FedEraser (FE), and FedRecovery (FR)—and exact methods—FATS (FT) and vanilla retraining via FedAvg (FA). More details like experimental errors, environment and settings are in Appendix G.1.

6.1 Ablation Study: FedShard Performance

In this section, we evaluate FedShard's impact on the primary task of federated training. This experiment also serves

as an ablation study, indicating the necessity of our proposed algorithms $\mathcal{A}_1$ (shard merging) and $\mathcal{A}_2$ (round allocation). We take CIFAR-10 with Non-IID degree $\rho = 0.1$, the merging rate $R = 2$ and the client number $K = 32$ as an example, using a 2-layer CNN model as in FedAvg.

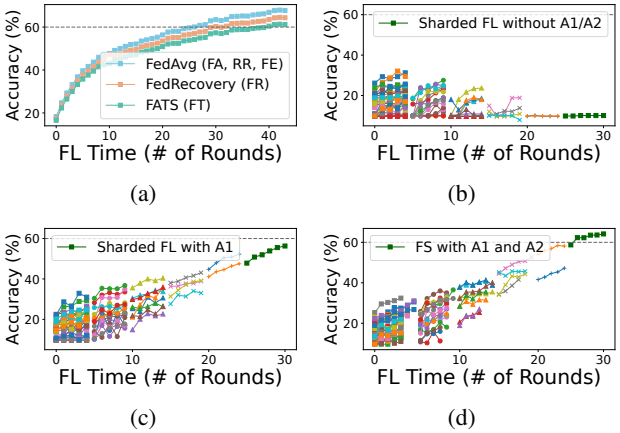


Figure 5: Global model accuracy on CIFAR-10 (Non-IID $\rho = 0.1$). The standard FedAvg (a) gives a baseline model performance on the federated training. The ablation study shows that naive sharding (b) is unstable. Our merging algorithm $\mathcal{A}_1$ (c) stabilizes training, and our round allocation algorithm $\mathcal{A}_2$ (d) further enables FedShard to surpass the accuracy of standard FedAvg.

Figure 5 shows that our adaptive algorithms are crucial for achieving high model accuracy in a sharded environment. Firstly, as shown in Figure 5(a), the standard FedAvg achieves an accuracy of 62.92% on CIFAR-10 using 30 epochs. However, as shown in Figure 5(b), a naive sharded approach without our algorithms fails and suffers from severe model collapse issues. After merging of each stage, the accuracy of the new shards drops sharply, and the model struggles to converge in the new stage. This empirical result proves the necessity of our merging algorithm $\mathcal{A}_1$.

Second, as shown in Figure 5(c), our cluster-based merging algorithm $\mathcal{A}_1$ significantly mitigates this issue and stabilize the sharded training. By creating directionally balanced shards, $\mathcal{A}_1$ ensures stable convergence, though the accuracy (56.3%) is still lower than standard FedAvg (62.92%). With both $\mathcal{A}_1$ and $\mathcal{A}_2$, FedShard achieves an accuracy of 64.14% as shown in Figure 5(d), even outperforming standard FedAvg. This further demonstrates the effects of $\mathcal{A}_2$.

This experiment fundamentally shows FedShard's practicality as a hierarchical federated training process, where $\mathcal{A}_1$ and $\mathcal{A}_2$ play an essential role in achieving even better convergence and model performance compared to traditional FL. Based on this, we further evaluate FedShard unlearning.

6.2 Efficiency and Efficiency Fairness

Next, we evaluate FedShard's unlearning cost (time) and its E.F. (M_e score). Due to space constraints, we show the experimental results on CIFAR-10 with Non-IID degree $\rho = 0.1$ and 256 clients, while additional results for other datasets, Non-IID degrees, and client numbers are in Appendix G.2 and G.3.

Figure 6 illustrates the distribution of unlearning costs (*i.e.*, unlearning time), revealing a clear efficiency and efficiency fairness dilemma among prior methods. Specifically, FA requires high unlearning costs. The state-of-the-art exact unlearning method (FT) exhibits extremely high variance of unlearning costs, signifying poor efficiency fairness. While approximate methods (FE, RR, and FR) are fast, they are challenged for performance unfairness as we later discuss in §6.3. FedShard achieves low costs as calibration-based approximate methods (i.e., efficiency) while maintaining a fair cost distribution for all clients' unlearning requests (i.e., efficiency fairness).

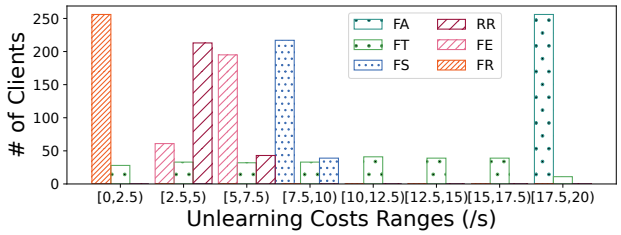


Figure 6: Unlearning cost distributions of different methods on CIFAR-10 with 256 clients, $R = 2$ and $\rho = 0.1$.

Table 1 quantifies these results across multiple datasets though unlearning time t and efficiency fairness metric M_e. For the efficiency, FedShard is consistently one of the fastest methods, achieving up to a 6.2 times speedup over vanilla retraining (FA) and a 4.9 times speedup over the other exact method FT. For the efficiency fairness (M_e): FedShard consistently achieves the best or second-best M_e score. In contrast, FT suffers from extremely high M_e scores (e.g., 1081.73 on CIFAR-10), confirming its inherent efficiency unfairness. Approximate methods also exhibit worse E.F. than FedShard, especially under high Non-IID settings. We provide more experimental results in Appendix G.2, indicating that FedShard maintains its efficiency and E.F. robustly for various datasets, Non-IID degrees. Furthermore, with more clients, FedShard is more advanced in efficiency.

6.3 Effectiveness and Performance Fairness

Finally, we evaluate FedShard's effectiveness and performance fairness through membership inference attacks (MIA), a cascaded leaving simulation, and data poisoning attacks (DPA) specific to unfair FU.

Effectivenss evaluation via MIA attack experiment. As shown in Table 2, the FS effectively unlearns the data, reducing the MIA F1-score to a low level same as retraining FA. This is aligned with our theoretical analysis in Proposition 1. Approximate methods (FE, RR, FR) often "over-forget", resulting in even lower F1-scores that hint at unfair performance degradation on remaining clients' data.

P.F. evaluation via cascaded leaving experiment. Cascaded leaving is a common issue in federated unlearning, where clients can leave the system due to unfair performance

		FA	FE	RR	FT	FR	**FS**
MNIST	t	9244.6	3099.2	2814.6	6832.1	**10.13**	**2485.1**
	M_e	52.44	**35.18**	36.87	727.66	53.17	**9.51**
	M_p	**5.93**	9.57	9.90	6.54	14.04	**6.08**
FMNIST	t	9984.4	2915.45	**2214.2**	6163.7	**11.83**	2508.3
	M_e	49.31	32.64	41.83	447.59	**31.39**	**11.62**
	M_p	**5.85**	9.38	9.82	6.46	14.23	**5.95**
CIFAR-10	t	13459.3	4166.8	4644.7	8403.9	**17.34**	**3361.1**
	M_e	75.64	41.32	39.30	1081.73	**35.92**	**10.9**
	M_p	**6.35**	9.78	9.23	6.67	19.06	**6.08**
CIFAR-100	t	29660.2	7576.1	**6398.2**	16822.5	**18.01**	6859.2
	M_e	76.98	42.24	41.58	1103.06	**31.82**	**11.23**
	M_p	**5.68**	11.74	10.94	7.13	22.14	**5.94**

Table 1: Average unlearning costs t (/s), M_e values, and M_p values of different FU methods. This table shows the results with client number $K = 512$ and non-iid level $\rho = 0.1$. We bold the best two scores and underline the worst score. We provide the experimental error in Table 4 in Appendix G.2.

Method	N.U	FA	FE	RR	FT	FR	**FS**
MIA F1-Score	91.37	34.18	30.14	28.65	36.07	25.07	35.04
DPA Precision (%)	NaN	**0%**	20.83%	22.91%	**0%**	28.50%	**0%**
M_p **Score**	NaN	6.23	9.81	9.14	6.42	21.55	**5.95**

Table 2: MIA attacks and DPA attacks of different methods on CIFAR-10. N.U. refers to the non-unlearned model, serving as an MIA baseline.

degradation caused by unlearning. The details and setting of the experiment are in Appendix G.5.

Figure 7 demonstrates the P.F. of our FS through both the P.F. metric (M_p) and the number of cascaded leavers. Specifically, calibration-based approximate unlearning methods including RR and FE have high M_p scores (*i.e.*, unfairer) and more cascaded leaving clients than retraining-based exact unlearning methods FS, FA, and FT. Their poor performance fairness causes cascaded leaving, and the number of cascaded leaving clients increases with non-IID degree decreasing. That is because with less non-IID data, the unfair performance degradation $\Delta\mathcal{Y}(D_r)$ of approximate unlearning methods is more significant. Other retraining methods including FT and FA also achieve low M_p scores and do not cause cascaded leaving as our proposed FS does. However, as discussed in § 6.2, those methods are less efficiency-fair and less efficient compared with our FS.

P.F. evaluation via DPA Attack experiment. If an FU algorithm is not performance-fair (insufficient unlearning or over-unlearning), it can be easily attacked by data poisoning attacks (DPA) (Tolpegin et al. 2020; Nuding and Mayer 2022; Gupta et al. 2023; Wang and Chaudhuri 2018) as we explain in Appendix G.5. As shown in Table 2, the approximate unlearning methods RR and FE fail to defend against DPA and exhibit high M_p scores, consistent with the performance unfairness observed in the cascaded leaving experiment. In contrast, the retraining methods FS, FT, and FA successfully defend against DPA while achieving low M_p scores, indicating that FS performs well in terms of P.F..

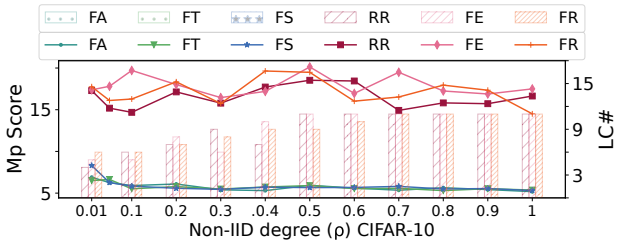


Figure 7: Comparison of M_p and cascaded leaving. The lines (left y-axis) represent the scores of P.F. metric M_p for different FU methods under different ρ and the bars (right y-axis) show the number of cascaded leaving clients (LC#).

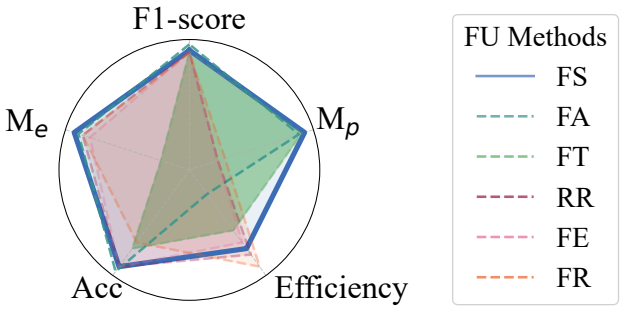


Figure 8: Holistic comparison of FU methods. Only FedShard (FS) excels across all five dimensions: Efficiency, efficiency fairness (M_e), performance fairness (M_p), model accuracy (Acc) and effectiveness (F1-score).

6.4 Summary of Comparison Results

We summarize the results of the above three subsections in Figure 8. Our experiments reveal a fundamental trade-off in prior work: approximate methods sacrifice P.F. for efficiency, while prior exact methods sacrifice E.F. or are prohibitively slow. FedShard is the only framework demonstrated to resolve this trade-off. As summarized in Figure 8, FedShard uniquely achieves high efficiency, efficiency fairness, and performance fairness simultaneously.

7 Conclusion

In this paper, we study the performance fairness and efficiency fairness of federated unlearning. We propose two practical fairness metrics to measure the fairness degree of an unlearning method. More importantly, we introduce FedShard, a sharded federated learning and unlearning framework to achieve fair and efficient federated unlearning. Our theoretical analysis shows that FedShard achieves both performance fairness and efficiency fairness at least $P/2$ times faster unlearning than retraining from scratch, where P is the stage number. Our evaluation results demonstrate that unfair federated unlearning methods cause cascaded leaving, poisoning attacks, and extreme unlearning cost distributions, which can be avoided by our fair FedShard unlearning.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant 62502412. This research is also partially supported by National Key R&D Program of China under Grant 2024YFE0200500 and National Natural Science Foundation of China under Grants 62202427 and 62572433.

References

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C. A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, 141–159. IEEE.

Feldman, A.; and Kirman, A. 1974. Fairness and envy. *The American Economic Review*, 64(6): 995–1005.

Gupta, P.; Yadav, K.; Gupta, B. B.; Alazab, M.; and Gadekallu, T. R. 2023. A novel data poisoning attack in federated learning based on inverted loss function. *Computers & Security*, 130: 103270.

Harding, E. L.; Vanto, J. J.; Clark, R.; Hannah Ji, L.; and Ainsworth, S. C. 2019. Understanding the scope and impact of the california consumer privacy act of 2018. *Journal of Data Protection & Privacy*, 2(3): 234–253.

Lan, T.; Kao, D.; Chiang, M.; and Sabharwal, A. 2010. An Axiomatic Theory of Fairness in Network Resource Allocation. In *2010 Proceedings IEEE INFOCOM*, 1–9.

Li, J.; and Xue, J. 2013a. Egalitarian division under Leontief preferences. *Economic Theory*, 54(3): 597–622.

Li, J.; and Xue, J. 2013b. Egalitarian division under Leontief preferences. *Economic Theory*, 54(3): 597–622.

Liu, G.; Ma, X.; Yang, Y.; Wang, C.; and Liu, J. 2021. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th international symposium on quality of service*, 1–10. IEEE.

Liu, Y.; Xu, L.; Yuan, X.; Wang, C.; and Li, B. 2022. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *2022-IEEE Conference on Computer Communications*, 1749–1758. IEEE.

Luc, D. T. 2008. Pareto optimality. *Pareto optimality, game theory and equilibria*, 481–515.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Mothukuri, V.; Parizi, R. M.; Pouriyeh, S.; Huang, Y.; Dehghantanha, A.; and Srivastava, G. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115: 619–640.

Nguyen, T. T.; Huynh, T. T.; Ren, Z.; Nguyen, P. L.; Liew, A. W.-C.; Yin, H.; and Nguyen, Q. V. H. 2022. A survey of machine unlearning. arXiv:2209.02299.

Nuding, F.; and Mayer, R. 2022. Data poisoning in sequential and parallel federated learning. In *Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics*, 24–34.

Procaccia, A. D. 2013. Cake cutting: Not just child's play. *Communications of the ACM*, 56(7): 78–87.

Tao, Y.; Wang, C.-L.; Pan, M.; Yu, D.; Cheng, X.; and Wang, D. 2024. Communication Efficient and Provable Federated Unlearning. *Proc. VLDB Endow.*, 17(5): 1119–1131.

Tarun, A. K.; Chundawat, V. S.; Mandal, M.; and Kankanhalli, M. 2024. Fast Yet Effective Machine Unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9): 13046–13055.

Tolpegin, V.; Truex, S.; Gursoy, M. E.; and Liu, L. 2020. Data poisoning attacks against federated learning systems. In *Computer security–ESORICs 2020: 25th European symposium on research in computer security, ESORICs 2020, guildford, UK, September 14–18, 2020, proceedings, part i 25*, 480–501. Springer.

Truex, S.; Baracaldo, N.; Anwar, A.; Steinke, T.; Ludwig, H.; Zhang, R.; and Zhou, Y. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 1–11.

Voigt, P.; and Von dem Bussche, A. 2017. The eu general data protection regulation. *A Practical Guide, : Springer International Publishing*, 10(3152676): 10–5555.

Wang, Y.; and Chaudhuri, K. 2018. Data poisoning attacks against online learning.

Xiong, Z.; Li, W.; Li, Y.; and Cai, Z. 2023. Exact-Fun: An Exact and Efficient Federated Unlearning Approach. In *2023 IEEE International Conference on Data Mining (ICDM)*, 1439–1444.

Zhang, H.; Nakamura, T.; Isohara, T.; and Sakurai, K. 2023a. A review on machine unlearning. *SN Computer Science*, 4(4): 337.

Zhang, L.; Zhu, T.; Zhang, H.; Xiong, P.; and Zhou, W. 2023b. FedRecovery: Differentially Private Machine Unlearning for Federated Learning Frameworks. *IEEE Transactions on Information Forensics and Security*, 18: 4732–4746.