Scalable Federated Unlearning via Isolated and Coded Sharding

Yijing Lin[1,2], **Zhipeng Gao**[1*], **Hongyang Du**[4], **Dusit Niyato**[4], **Gui Gui**[5], **Shuguang Cui**[3,2], **Jinke Ren**[2,3*]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications
[2] The Future Network of Intelligence Institute, The Chinese University of Hong Kong (Shenzhen)
[3] School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen)
[4] College of Computing and Data Science, Nanyang Technological University
[5] School of Automation, Central South University
jinkeren@cuhk.edu.cn, gaozhipeng@bupt.edu.cn

Abstract

Federated unlearning has emerged as a promising paradigm to erase the client-level data effect without affecting the performance of collaborative learning models. However, the federated unlearning process often introduces extensive storage overhead and consumes substantial computational resources, thus hindering its implementation in practice. To address this issue, this paper proposes a scalable federated unlearning framework based on isolated sharding and coded computing. We first divide distributed clients into multiple isolated shards across stages to reduce the number of clients being affected. Then, to reduce the storage overhead of the central server, we develop a coded computing mechanism by compressing the model parameters across different shards. In addition, we provide the theoretical analysis of time efficiency and storage effectiveness for the isolated and coded sharding. Finally, extensive experiments on two typical learning tasks, i.e., classification and generation, demonstrate that our proposed framework can achieve better performance than three state-of-the-art frameworks in terms of accuracy, retraining time, storage overhead, and F1 scores for resisting membership inference attacks.

1 Introduction

With the increasing awareness and stringent regulations of data protection, there is a growing demand for new machine learning technologies that can reap the full benefit of rich data while preserving data privacy. Federated learning (FL) [McMahan *et al.*, 2017] has become a promising paradigm to collaboratively train a learning model across multiple clients without sharing their local data. As shown in Figure 1(a), each client independently trains a local model using its local dataset and uploads the model parameters to a central server. The server collects the model parameters from clients and broadcasts the aggregated model to all clients. This process is iterated until model convergence.

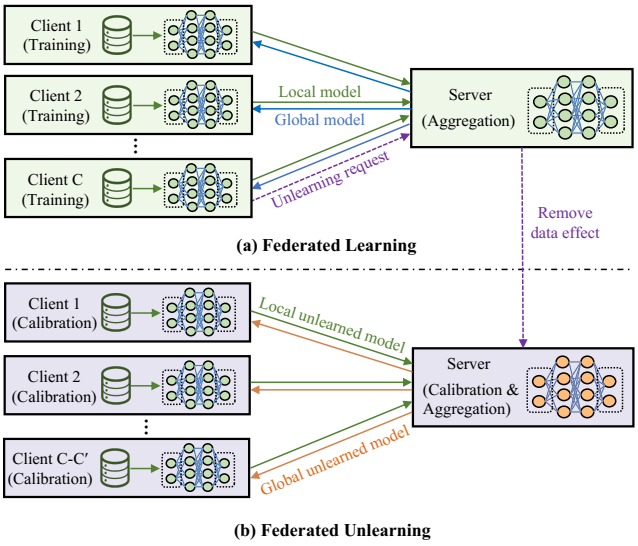

Figure 1: **Federated Learning vs. Federated Unlearning.** (a) In federated learning, the clients and the server collaboratively train a global model by exchanging model parameters. (b) In federated unlearning, upon receiving an unlearning request from a specific client C', the well-trained global model will be unlearned by using the local models of other clients for calibration, thus removing the corresponding data effect.

Despite the great potential of FL, it encounters challenges in data privacy since the shared model parameters still contain some data information. On the other hand, the European Union's General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA) in the United States have released critical requirements for the "right to be forgotten", which allows clients to erase the data effects from the model parameters trained on their local datasets [Chen and Yang, 2023; Liu *et al.*, 2022; Su and Li, 2023], thus motivates a new computing paradigm called *federated unlearning* [Liu *et al.*, 2021; Liu *et al.*, 2022; Su and Li, 2023]. As shown in Figure 1(b), each client performs local model calibration and sends the calibrated model parameters to the central server, which removes the data effect from the well-trained FL model via calibration and aggregation. Since the unlearned model does not need to be trained from scratch, the

*Corresponding authors: Jinke Ren and Zhipeng Gao.

computational overhead can be significantly reduced.

Recent studies on federated unlearning have identified two distinct strategies, including provable and unprovable guarantees. Provable guarantees [Bourtoule *et al.*, 2021] ensure the complete removal of data effects, while unprovable guarantees indicate that the model has almost forgotten the data effects of specific clients. To reduce retraining time, a new framework called FedEraser is proposed in [Liu *et al.*, 2021], which removes the data effects of specific clients by storing intermediate model parameters on the central server. Although FedEraser achieves provable guarantees and satisfactory unlearning performance, it is short of scalability due to the extensive storage overhead. On the other hand, several orthogonal works [Liu *et al.*, 2022; Wu *et al.*, 2022; Wang *et al.*, 2022] employ various unprovable guarantee-based techniques to enhance the unlearning effectiveness. However, these works mainly optimize loss functions and prune network layers to bound the removal level of data effect, which cannot provide stringent provable guarantees.

To overcome the aforementioned challenges, this paper proposes a scalable federated unlearning framework to efficiently remove the data effects of specific clients while maintaining the model accuracy. Specifically, the entire learning and unlearning process is divided into multiple stages, in which we construct several isolated shards and perform efficient retraining to reduce the storage overhead of the central server. Based on this, we carry out *coded computing* on different shards to further improve the storage efficiency and reduce the training time. To demonstrate the effectiveness of the proposed framework, we conduct extensive experiments on two typical learning tasks, i.e., classification and generation, and compare our proposed framework with three state-of-the-art frameworks, i.e., FedEraser [Liu *et al.*, 2021], RapidRetrain [Liu *et al.*, 2022], and FedRetrain [Wang *et al.*, 2022; Su and Li, 2023]. Our main contributions can be summarized as follows:

- We for the first time introduce a stage-based isolated sharding mechanism to reduce the number of affected clients for federated unlearning. Theoretical analysis proves that the expected time cost for processing unlearning requests can be significantly reduced in both sequential and concurrent cases.
- We design a coded computing-based sharding mechanism to improve the system scalability. Specifically, it can improve the storage efficiency by $(1 - 2\mu)C$ and the throughput by $S/O(C^2 \log^2 C \log\log C)$, where C represents the number of clients, μ is the proportion of erroneous results, and S is the number of shards.
- Experiment results show that the proposed framework can reduce at least 65% retraining time and 98% storage overhead while achieving comparable unlearning effectiveness to FedEraser, RapidRetrain, and FedRetrain.

2 Related Work

To mitigate the data effects of specific clients, machine unlearning is proposed by partitioning training data into isolated slices and performing a joint sharded, isolated, sliced, and aggregated (SISA) method [Bourtoule *et al.*, 2021; Xu *et al.*, 2023]. Specifically, upon receiving an unlearning request, these slices completely remove the data effects and retrain the learning model to save computational resources and achieve unlearning with provable guarantees. Besides SISA, some previous works also adopt unprovable guarantee-based methods, such as gradient ascent [Chen and Yang, 2023], adding unlearning layers via a selective teacher-student formulation [Jang *et al.*, 2022], and transforming the unlearning process into a single-class classification task [Yan *et al.*, 2022]. Although these methods have achieved good unlearning performance, they need direct access to client data, thus cannot be applied in FL due to data privacy.

Recent studies have paid attention to removing the data effects of specific clients from well-trained FL models. Specifically, the authors in [Liu *et al.*, 2021] initially propose the concept of federated unlearning and develop a new framework called FedEraser. By using the storage resources of the central server, it can retain intermediate model parameters to achieve unlearning with provable guarantees. To reduce computational overhead while maintaining model accuracy, many unprovable guarantee-based frameworks have also been developed, such as RapidRetrain [Liu *et al.*, 2022], TF-IDF [Wang *et al.*, 2022; Salton and Buckley, 1988], FedRecovery [Zhang *et al.*, 2023], KNOT [Su and Li, 2023], and BFU [Wang *et al.*, 2023]. Specifically, RapidRetrain modifies the loss function to accelerate the retraining process and remove the data effects of all clients. In classification tasks, TF-IDF is utilized to unlearn the contributions of specific classes by pruning the most relevant class-discriminative channels [Wang *et al.*, 2022]. FedRecovery considers differential privacy in the federated unlearning process, where intermediate model parameters from clients are utilized to retrain the global model. KNOT proposes a clustered aggregation-based asynchronous unlearning mechanism to reduce the retraining cost. BFU develops a parameter self-sharing approach to maintain model accuracy while erasing the data effects of clients. While these works have achieved impressive unlearning performance via experiments, they mainly consider unprovable guarantees such that the unlearning requirements may not be fulfilled [Bourtoule *et al.*, 2021].

3 Methodology

In this section, we first describe the federated unlearning framework and then introduce two mechanisms based on isolated sharding and coded computing.

3.1 Scalable Federated Unlearning Framework

We consider an FL system consisting of S central servers and C distributed clients, denoted by a set $\mathcal{C} = \{C_1, \ldots, C_C\}$. Each client collects a fraction of data and constitutes its local dataset. A shared learning model $\mathbf{w}$ needs to be collaboratively trained across all clients and servers. In the training process, the intermediate model parameters are stored on the servers for subsequent unlearning purposes. To remove the data effects of specific clients, we define a subset of clients, denoted by $\mathcal{C}' \subseteq \mathcal{C}$, in which each client needs to be unlearned based on the unlearning requests. Meanwhile, we

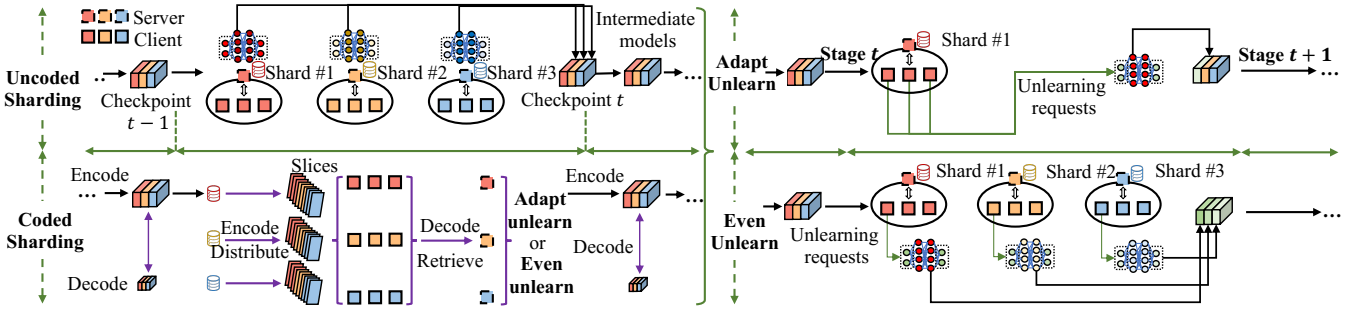

Figure 2: **Scalable Federated Unlearning Framework**. For the unlearning requests initiated at different stages, only affected shards perform calibrations to remove the data effects of specific clients. To reduce storage overhead and improve scalability, intermediate model parameters are encoded/decoded as slices for efficient communication between clients and servers.

define $\mathcal{D}' \subseteq \mathcal{D}$ as the overall dataset associated with the unlearned clients. Let $\mathbf{w}'$ denote the unlearned global model. Then, according to [Kurmanji *et al.*, 2023], the unlearned global model should satisfy

$$\begin{cases} I\left(\mathbf{w}(\mathcal{D}'); \mathbf{w}'(\mathcal{D}')\right) = 0, \\ I\left(\mathbf{w}(\mathcal{D}-\mathcal{D}'); \mathbf{w}'(\mathcal{D}-\mathcal{D}')\right) = 1, \end{cases} \tag{1}$$

where $I(\cdot)$ represents the mutual information.

To achieve (1), most existing solutions utilize the intermediate model parameters stored on the servers to unlearn the well-trained global model, which inevitably increases the storage overhead of the servers. To solve this problem, we present a new federated unlearning framework, which considers both uncoded and coded sharding to meet adaptive and even unlearning requests. As shown in Figure 2, uncoded sharding utilizes a stage-based isolated sharding mechanism to erase the data effect in the global model, which will be illustrated in Section 3.2. Coded sharding compresses intermediate model parameters into slices, which is able to maintain the storage overhead as the number of clients increases and will be detailed in Section 3.3.

3.2 Stage-based Isolated Sharding Mechanism

Initialization. Since each client may join or leave the system at any time, we divide the entire learning and unlearning process into multiple stages, where the learning and unlearning operations are performed within each stage. Specifically, at each stage, the clients are distributed over multiple shards, denoted by a set $\mathcal{S}$. In each shard, there is a particular server for model aggregation. Therefore, the number of shards is equal to the number of servers, i.e., S. The clients in the shard s are defined by a set $\mathcal{C}_s$. In addition, the dataset and the server associated with the shard s are denoted by $\mathcal{D}_s$ and J_s, respectively.

At each stage, the clients in the same shard perform the FedAvg algorithm [McMahan *et al.*, 2017] to train a global model. Specifically, each client first trains its local model by L epochs and then transmits the model parameters to the corresponding server for aggregation. Thereafter, the aggregated model parameters are broadcast to the clients for local model update. These steps are iterated for G rounds to obtain a well-trained FL model, which serves as the foundational model for the subsequent unlearning process. Moreover, the server stores the intermediate model parameters of clients in the same shard, which will also be used in the subsequent unlearning process.

Preparation. At each stage, we assume that there are K unlearning requests across the impacted shards, denoted by a set $\mathcal{S}'$. In each impacted shard $s_i \in \mathcal{S}'$, the clients and unlearning clients are denoted by $\mathcal{C}_{s_i}$ and $\mathcal{C}'_{s_i}$, respectively. Let J_{s_i} denote the server associated with the impacted shard s_i, which collects intermediate model parameters $\mathbf{w}^g_{\mathcal{C}_{s_i}}, \forall g \in \mathcal{G}$ from the clients in the same shard for storage, where $\mathcal{G} = \{1, \cdots, G\}$ is the set of the global learning round. Next, the server J_{s_i} removes the intermediate model parameters associated with the unlearning clients $\mathcal{C}'_{s_i}$ from the whole parameter set, as given by $\mathbf{w}^g_{s_i} = \mathbf{w}^g_{\mathcal{C}_{s_i}} - \mathbf{w}^g_{\mathcal{C}'_{s_i}}, \forall g \in \mathcal{G}$. Then, the server aggregates the model parameters of the retained clients to obtain the initial global unlearned model, as given by

$$\mathbf{w}^{g'=0}_{s_i} = \frac{1}{M}\sum_{m=1}^{M} \mathbf{w}^g_{s_i,m}, \tag{2}$$

where $\mathbf{w}^g_{s_i,m}$ is the local model of the retained client m in the shard s_i, M is the number of retained clients, g' denotes the unlearning round, and $\mathcal{G}' = \{1, \cdots, G\}$ represents the set of all unlearning rounds. Finally, $\mathbf{w}^{g'=0}_{s_i}$ is sent to the retained clients in the same shard, i.e., $\mathcal{C}_{s_i} - \mathcal{C}'_{s_i}$ for subsequent retraining.

Retraining. In the unlearning round g', instead of retraining from scratch, each retained client in shard s_i receives the global unlearned model $\mathbf{w}^{g'}_{s_i}$ and utilizes its local dataset to update $\mathbf{w}^{g'}_{s_i}$ by $\frac{L}{r}$ local epochs, where r is a ratio for reducing the number of local retraining rounds [Liu *et al.*, 2021]. Then, the updated model parameters are uploaded to the server, which calibrates and updates the global unlearned model by

$$\mathbf{w}^{g'+1}_{s_i} = \mathbf{w}^{g'}_{s_i} + \frac{1}{M}\sum_{m=1}^{M} \frac{\|\mathbf{w}^g_{\mathcal{C}_{s_i,m}}\|}{\|\mathbf{w}^{g'}_{\mathcal{C}'_{s_i,m}}\|} \mathbf{w}^{g'}_{\mathcal{C}'_{s_i,m}}. \tag{3}$$

Here, $g = g'$ indicates that $\mathbf{w}^{g'}_{\mathcal{C}'_{s_i,m}}$ is used to calibrate the local model in the same global learning round. Finally, $\mathbf{w}^{g'+1}_{s_i}$

is distributed to the retained clients in the shard. This process will be iterated for G rounds.

According to (1), the provable guarantees of the unlearning process is achieved if the global unlearned model satisfies

$$\begin{cases} I\left(\mathbf{w}_{s_i}^{g}(\mathcal{D}'_{s_i}); \mathbf{w}_{s_i}^{g'}(\mathcal{D}'_{s_i})\right) = 0, \\ I\left(\mathbf{w}_{s_i}^{g}(\mathcal{D}_{s_i} - \mathcal{D}'_{s_i}); \mathbf{w}_{s_i}^{g'}(\mathcal{D}_{s_i} - \mathcal{D}'_{s_i})\right) = 1. \end{cases} \tag{4}$$

To achieve this, it is important to maintain the isolation of the shard in the entire unlearning process, i.e., avoiding cross-shard interactions at each stage. We shall note that there are scenarios where cross-shard interactions are necessary at different stages. In such cases, unprovable guarantee-based methods can be employed. However, it is not the main focus of this paper and will not be discussed herein.

3.3 Coded Computing-based Sharding Mechanism

As stated in [Bourtoule *et al.*, 2021], an effective federated unlearning framework should be lightweight and scalable, which is able to balance the retraining time, storage overhead, and computational cost for unlearning. To achieve this goal, we develop a new coded computing-based sharding mechanism, which is composed of two parts, including coded computation and coded reconstruction.

Coded Computation. In the isolated sharding mechanism, the server stores the uncoded intermediate model parameters $\mathbf{w}_{\mathcal{C}_s}^{g}, \forall g \in \mathcal{G}$ of the clients in the same shard, which are used to remove the data effects from the global model. Conversely, in the coded computing-based sharding mechanism, the server collects the coded intermediate model parameters, denoted by $\widetilde{\mathbf{w}}_i, \forall i \in \{1, \cdots, C\}$, which are generated from $\mathbf{w}_{\mathcal{C}_s}^{g}$ and distributed in clients. Without loss of generality, we utilize the Lagrange interpolation polynomial method [Roth, 2006] to compute $\widetilde{\mathbf{w}}_i$. Specifically, we first select S different real numbers $\{\omega_1, \ldots, \omega_S\}$, where ω_s is associated with the shard s. Then, the Lagrange polynomial with variable α is given by

$$u(\alpha) = \sum_{s=1}^{S} \mathbf{w}_{\mathcal{C}_s}^{g} \prod_{j \neq s} \frac{\alpha - \omega_j}{\omega_s - \omega_j}. \tag{5}$$

To distribute the intermediate model parameters over all clients, we further select C different real numbers $\{\alpha_1, \ldots, \alpha_C\}$, where α_i is associated with client i and C is the total number of clients. Based on this, the coded intermediate model parameters stored in client i can be generated at point α_i, as

$$\widetilde{\mathbf{w}}_i = u(\alpha_i) = \sum_{s=1}^{S} \mathbf{w}_{\mathcal{C}_s}^{g} \prod_{j \neq s} \frac{\alpha_i - \omega_j}{\omega_s - \omega_j}. \tag{6}$$

In the coded computing-based sharding mechanism, each client not only stores a mix of coded intermediate model parameters across different shards but also generates a series of keys. These keys are shared by the clients and the server in each shard, which will be used in the following coded reconstruction. In particular, each client distributes the coded intermediate model parameters to other clients within or outside the shard.

We shall note that coded computation is utilized in the initialization phase described in Section 3.2. Instead of directly storing the intermediate model parameters, the server only collects coded intermediate model parameters from clients for unlearning. Since the size of the coded model parameters is much smaller than that of the uncoded ones, the storage overhead of the server can be significantly reduced.

Coded Reconstruction. In the preparation phase introduced in Section 3.2, instead of directly utilizing the intermediate model parameters, the server in each impacted shard utilizes the keys (generated in coded computation) to retrieve and reconstruct these model parameters from clients. This process is akin to decoding a Reed-Solomon code [Roth, 2006]. Given the evaluations at C distinct points, the server reconstructs the model parameters by decoding a Reed-Solomon code with dimension S and length C. The detailed decoding process can be described as follows: *Step 1:* Let $\{\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_C\}$ denote the corresponding coded slices of the model parameters among clients, which can be obtained according to (6); *Step 2:* The server uses the keys shared by the clients in the same shard to access these slices; *Step 3:* The server reconstructs the original model parameters by solving the following linear equation derived from the Reed-Solomon decoding process, as

$$\widetilde{\mathbf{w}}_{\mathcal{C}_s}^{g} = \begin{bmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{S-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{S-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_C & \cdots & \alpha_C^{S-1} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \widetilde{\mathbf{w}}_1 \\ \widetilde{\mathbf{w}}_2 \\ \vdots \\ \widetilde{\mathbf{w}}_C \end{bmatrix}, \tag{7}$$

where the matrix formed by α_i is a Vandermonde matrix. It is invertible as long as all elements are distinct. According to (7), the server can accurately reconstruct the intermediate model parameters from the coded slices distributed among all clients. Moreover, since each client only receives a single slice of the coded model parameters, it is difficult for other clients to reconstruct all coded model parameters, thus preserving the data privacy.

After finishing the decoding process, the server in each impacted shard removes the intermediate model parameters associated with the clients requesting unlearning, initiates the retraining phase, and retrains the global model. The overall federated unlearning process is summarized in Algorithm 1.

4 Theoretical Analysis

4.1 Time Efficiency for Isolated Sharding

Sequential Unlearning Requests. In the sequential setting, each unlearning request is addressed individually. Given S shards and K unlearning requests, the probability that a shard s is selected for retraining j times across $i-1$ unlearning requests is given by

$$P_s = \binom{i-1}{j} \left(\frac{1}{S}\right)^{j} \left(1 - \frac{1}{S}\right)^{i-1-j}, \tag{8}$$

where $\frac{1}{S}$ is the probability that affects any given shard. Let $\overline{C}_t$ denote the average time cost associated with all shards.

Algorithm 1 Federated Unlearning with Isolated Sharding and Coded Computing

for each shard $s \in \mathcal{S}$ **do**
  Initialization: // `Run on clients`
  Perform coded computation via (6).
  Distribute $\widetilde{\mathbf{w}}_i$ to the clients within or outside shard.
  if $s \in \mathcal{S}$ **then**
    Preparation: // `Run on the server`
    Perform coded reconstruction via (7).
    Remove the intermediate model parameters of the unlearning clients.
    Obtain the initial unlearned global model via (2).
    Retrain: // `Run on clients and server`
    Retrain the global model for unlearning via (3).
  end if
end for

Then, the expected time cost for processing all unlearning requests can be estimated by

$$T_s = \sum_{i=1}^{K} \sum_{j=0}^{i-1} \left(\frac{1}{S}\right)^{j} \left(1 - \frac{1}{S}\right)^{i-1-j} \overline{C}_t \overset{(a)}{=} K\overline{C}_t, \quad (9)$$

where (a) is obtained by the binomial theorem [Bourtoule *et al.*, 2021].

Concurrent Unlearning Requests. In the concurrent setting, the unlearning requests are aggregated in a batch for joint processing. Let $\{b_1, \ldots, b_S\}$ denote a set of Bernoulli random variables, where b_s represents whether shard s is affected in the unlearning process. Therefore, $P(b_s = 1) = \frac{1}{S}$. Accordingly, the expected time cost for processing all unlearning requests can be estimated by

$$T_c = \mathbb{E}\left(\sum_{s=1}^{S} b_s \overline{C}_t\right) \overset{(b)}{=} S\overline{C}_t \left(1 - \left(1 - \frac{1}{S}\right)^{K}\right), \quad (10)$$

where (b) is derived from the expected value of the Bernoulli random variable.

4.2 Storage Effectiveness for Coded Sharding

In this work, we use two typical metrics to evaluate the effectiveness of the coded sharding mechanism: 1) *Storage efficiency*, denoted by γ, which is defined as the ratio of the size of intermediate model parameters to that of the data stored in the server; 2) *Throughput*, denoted by λ, which characterizes the capacity for processing unlearning requests. The throughput is mainly determined by two factors, i.e., the number of unlearning requests being processed and the associated computational cost. We shall note that the storage efficiency and throughput are the same for the sequential and concurrent settings since only the size of the model parameters is different.

We take the full storage mechanism [Liu *et al.*, 2021] as the benchmark, where all the intermediate model parameters are stored in the servers. Therefore, its storage efficiency is set as $\gamma_f = 1$. Moreover, since the servers handle all unlearning requests, the throughput is $\lambda_f = 1$. For the proposed uncoded sharding mechanism, all clients are divided into S shards. Therefore, its storage efficiency and throughput are given by $\gamma_s = S$ and $\lambda_s = S$, respectively. On the other hand, the proposed coded sharding mechanism is resistant to μC erroneous results, where μ is the proportion of the erroneous results to the coded slices [Roth, 2006]. Thus, it follows

$$2\mu C \leq C - S, \quad (11)$$

which implies that the upper bound of S is $(1 - 2\mu)C$. Accordingly, we have

$$S \leq \gamma_c \leq (1 - 2\mu)C. \quad (12)$$

Then, given $O(C^2 \log^2 C \log\log C)$ as the additional computational overhead for coded computing [Li *et al.*, 2020], the throughput of the coded sharding mechanism can be expressed as

$$\lambda_c = S/O(C^2 \log^2 C \log\log C). \quad (13)$$

5 Experiments

5.1 Experimental Settings

Federated Learning and Unlearning. We consider a total of 100 clients in our experiments to demonstrate the effectiveness of the proposed framework. In the learning process, only 20 clients are randomly selected in each training round. These clients are divided into 4 shards such that each shard has 5 clients. The numbers of local epochs and training rounds are set as 10 and 30, respectively. In the unlearning process, we consider two types of unlearning requests: 1) Even, where all requests are evenly distributed across shards; 2) Adapt, where all requests are adaptively initiated in one shard [Bourtoule *et al.*, 2021].

Datasets and Models. We use four commonly-adopted datasets including MNIST [LeCun *et al.*, 1998], Fashion-MNIST [Xiao *et al.*, 2017], CIFAR-10 [Krizhevsky *et al.*, 2009], and Tiny Shakespeare [McMahan *et al.*, 2017] for experiments, which are applicable to a diverse range of tasks. On the other hand, we consider two typical learning tasks, i.e., image classification and language generation. To accomplish the two tasks, we use two learning models: 1) Convolutional neural network, which is composed of 2 convolutional, 2 pooling, and 2 fully connected layers. It will be trained on the MINST, Fashion-MINST, and CIFAR-10 datasets, respectively; 2) NanoGPT[1] [Radford *et al.*, 2019], which consists of a 4-layer transformer with 4 attention heads [Vaswani *et al.*, 2017], an embedding layer with dimension = 16, a block layer, and a vocabulary with size = 109. In particular, NanoGPT is trained on the Tiny Shakespeare dataset. For data distribution, we consider two data-partition approaches: 1) Independent and identically distributed (IID), where all data samples are randomly divided into 100 equal parts, and each client is assigned with one part; 2) Non-IID. Specifically, for the image classification task, 80% data samples of each client belong to one primary class, while the remaining data samples belong to other classes [Wang *et al.*, 2020]. For the language generation task, the entire dataset is divided into

[1] https://github.com/karpathy/nanoGPT

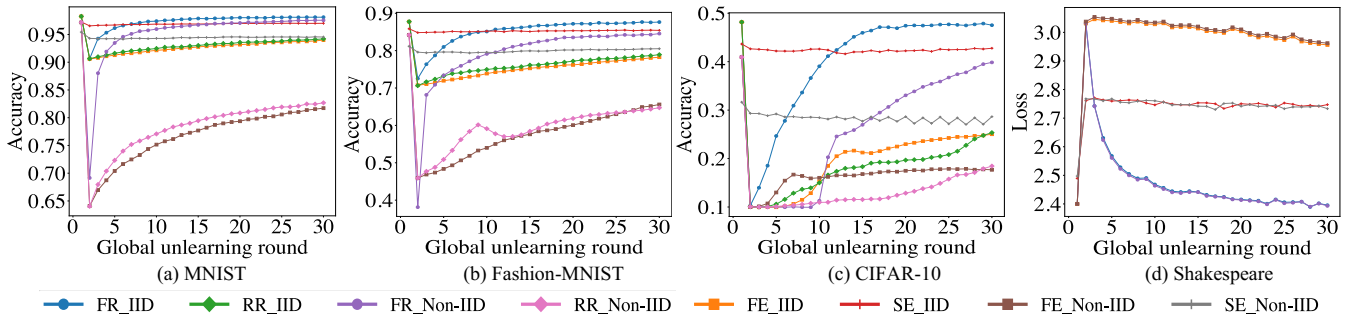

Figure 3: Performance with a single unlearning request.

	MNIST		FMNIST		CIFAR-10		Shakespeare	
F1 Score (↓)	IID	Non-IID	IID	Non-IID	IID	Non-IID	IID	Non-IID
FR	0.5455	0.5354	0.5423	0.5434	0.5074	**0.4742**	**0.5426**	0.5995
FE	**0.5450**	**0.1953**	**0.4659**	**0.2803**	**0.1649**	0.6536	-	-
RR	0.5496	0.2289	0.5203	0.5553	0.5264	0.6684	-	-
SE	0.5486	0.5959	0.5447	0.5560	0.4812	0.5141	0.6240	**0.0392**
Retraining Time (↓)	IID	Non-IID	IID	Non-IID	IID	Non-IID	IID	Non-IID
FR	565.68	563.66	556.57	558.25	573.78	571.92	2406.96	2343.51
FE	293.84	287.40	290.90	291.94	301.57	304.22	1196.03	1213.69
RR	391.71	396.40	376.49	390.88	386.00	392.61	-	-
SE	**96.79**	**96.51**	**96.50**	**98.12**	**105.05**	**103.96**	**148.06**	**136.95**
Gain	↓ 67.06%	↓ 66.41%	↓ 66.82%	↓ 66.39%	↓ 65.16%	↓ 65.82%	↓ 87.62%	↓ 88.71%

Table 1: F1 score and retraining time in IID and non-IID scenarios.

several unbalanced buckets, and each client is assigned with two buckets to ensure non-IID data distribution across different clients.

Baseline Frameworks. We adopt three state-of-the-art baseline frameworks: 1) FedRetrain (FR) [Liu *et al.*, 2021], which retrains the model from scratch to achieve unlearning purposes; 2) FedEraser (FE) [Liu *et al.*, 2021], which calibrates model parameters using the historical updates from retained clients; 3) RapidRetrain (RR) [Liu *et al.*, 2022], which employs a diagonal empirical Fisher information matrix to expedite retraining. We name our proposed framework as SE, which is the abbreviation of **S**harding **E**raser enabled by isolated and coded sharding.

Performance Metrics. Without loss of generality, the unlearning effectiveness can be evaluated by four metrics, including accuracy, retraining time, storage overhead, and F1 score for resisting membership inference attacks (MIAs) [Shokri *et al.*, 2017]. Note that F1 score is utilized to verify whether data is successfully unlearned from the model [Bourtoule *et al.*, 2021; Liu *et al.*, 2021]. These four metrics align with the fundamental principle outlined in [Bourtoule *et al.*, 2021], which focuses on maintaining model accuracy, reducing unlearning time, as well as providing security guarantees.

5.2 Experimental Results

Performance with Single Unlearning Request. We first consider a simple scenario in which each retained client has a single unlearning request. In this scenario, we test the unlearning performance of the four frameworks, and the results are shown in Figure 3. As can be seen from Figures 3(a)-(c), the proposed framework SE can achieve comparable accuracy to the baseline framework FR in both IID and non-IID cases. Moreover, SE always outperforms the other two baseline frameworks, i.e., FE and RR. The results on the Tiny Shakespear dataset are very similar, except that the rapid retraining framework RR cannot converge [Su and Li, 2023]. On the other hand, since FR retrains the model from scratch and entirely removes data effects of specific clients, its F1 score and accuracy should be jointly considered for performance comparison. As shown in Table 1 and Figure 3, the F1 score of FE is lower than those of FR, RR, and SE. How-

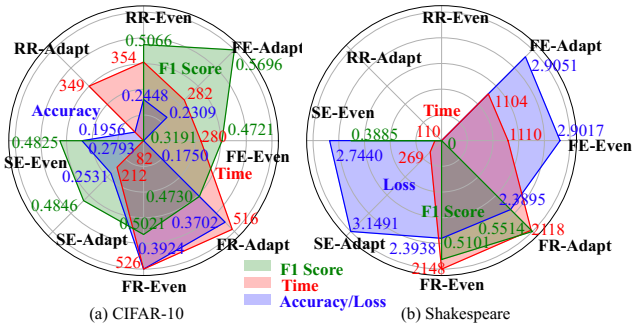

Figure 4: Performance with concurrent unlearning requests.

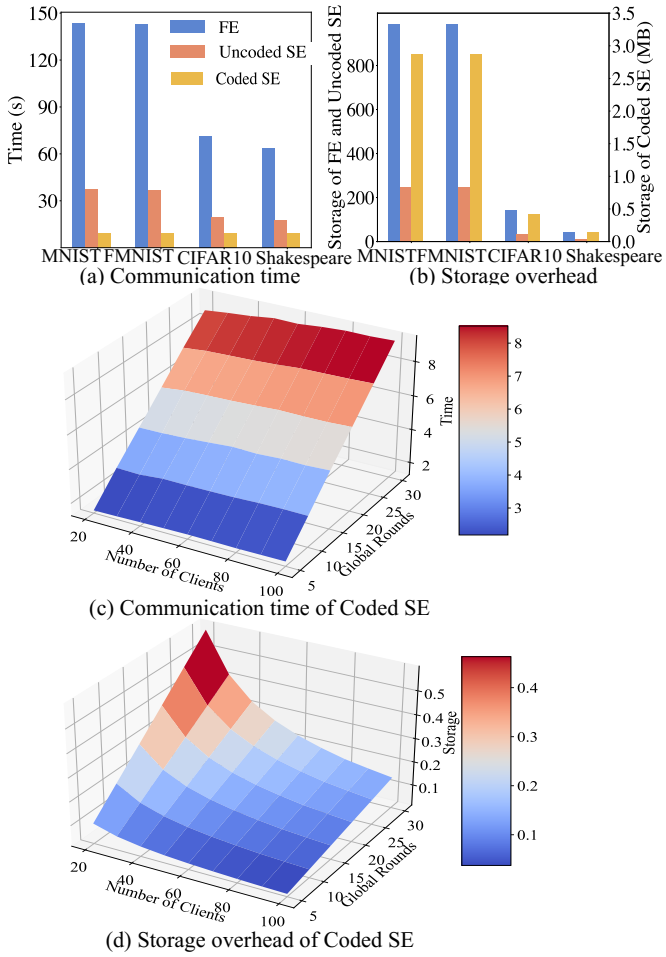

Figure 5: Communication time and storage overhead of different frameworks with concurrent adaptive unlearning requests.

ever, this result is attributed to the reduced accuracy of FE, which cannot be considered as the improved unlearning performance. Instead, our proposed framework SE can achieve comparable F1 scores with FR while maintaining model accuracy. Besides, for the retraining time, our framework SE always surpasses the other three frameworks. For example, in the non-IID case, SE can achieve about 66.41%, 66.39%, 65.16%, and 87.62% time reduction on the MNIST, Fashion-MNIST, CIFAR-10, and Tiny Shakespeare dataset. Therefore, our proposed framework can significantly reduce the retraining time without sacrificing accuracy.

Performance with Concurrent Unlearning Requests. To demonstrate the scalability of the proposed framework, we further consider a scenario in which each retained client has multiple concurrent unlearning requests. In particular, each request can be evenly or adaptively initiated across different shards. The experimental results are illustrated in Figure 4. Specifically, on the CIFAR-10 dataset, the proposed framework SE can outperform FE and RR, and can offer a comparable F1 score to all baseline frameworks. Moreover, it reduces the retraining time by 70% in the even scenario. Similar trends can be observed in the Tiny Shakespear dataset. These gains are attributed to the fact that SE can significantly reduce the number of affected clients in the unlearning process. As for the adaptive scenario, the variance in data quality across different shards may affect the loss of SE on the Tiny Shakespeare dataset. Nevertheless, its retraining time is still the shortest among all frameworks, demonstrating the effectiveness of the proposed framework in terms of time efficiency.

Storage Overhead with Concurrent Unlearning Requests. Since FR and RR do not utilize storage resources to improve unlearning performance, we only compare the proposed framework SE with the baseline framework FE. We take the concurrent adaptive scenario for experiments. Therein, the communication time for transmitting model parameters includes two parts: 1) Base network delay, which is set as 0.1 second; 2) Model transmission time, which is computed as the ratio of the model size (in bits) and the network data rate (in bit/s). Moreover, to demonstrate the benefit of the coded computing mechanism, we define Coded SE as the framework with isolated and coded sharding, and denote Uncoded SE as the framework with only isolated sharding. In both frameworks, the storage overhead includes the model parameters stored in one shard.

Due to that the storage overhead is independent of data distribution, we take the IID scenario as an example. Figure 5(a) and Figure 5(b) show the communication time and storage overhead of FE, Uncoded SE, and Coded SE with concurrent adaptive unlearning requests. We can observe that Coded SE achieves the smallest storage overhead and the minimum communication time among all frameworks. For example, the storage overhead can be reduced by almost 98%. On the other hand, we perform experiments on the Shakespeare dataset to investigate the impact of the total number of clients and global rounds on the storage overhead. The results are depicted in Figures 5(c)-(d). From these figures, we can see that Coded SE shows a sharp reduction in storage overhead with a slight increase in communication time as the number of clients increases. This result is attributed to the increased division of model parameters into more shards for storage when the number of clients increases. Also, there exists a marginal increase in the distribution and retrieval time when the server utilizes the model parameters for unlearning.

6 Conclusion

In this paper, we have developed a scalable federated unlearning framework by introducing two mechanisms based on isolated sharding and coded computing. The isolated sharding mechanism divides distributed clients into multiple shards and removes the data effects according to sequential or concurrent unlearning requests. The coded computing mechanism extends the scalability of the framework by encoding, decoding, distributing, and retrieving intermediate model parameters. By doing so, the storage overhead of the central server can be significantly reduced. Theoretical analysis and experimental results demonstrate the effectiveness of the proposed framework as compared with three baseline frameworks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62372050) and Beijing Natural Science Foundation (4232029). This work is also supported in part by NSFC with Grant No. 62293482, the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, the National Key R&D Program of China with grant No. 2018YFB1800800, by Shenzhen Outstanding Talents Training Fund 202002, by Guangdong Research Projects No. 2017ZT07X152 and No. 2019CX01X104, by Key Area R&D Program of Guangdong Province (Grant No. 2018B030338001), by the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), and by Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. ZDSYS201707251409055). This work is supported by the National Research Foundation, Singapore, and Infocomm Media Development Authority under its Future Communications Research & Development Programme, Defence Science Organisation (DSO) National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-019 and FCP-ASTAR-TG-2022-003), and Singapore Ministry of Education (MOE) Tier 1 (RG87/22).

References

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

[Chen and Yang, 2023] Jiaao Chen and Diyi Yang. Unlearn what you want to forget: Efficient unlearning for llms. In *EMNLP 2023-Empirical Methods in Natural Language Processing*, 2023.

[Jang *et al.*, 2022] Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. 2022.

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

[Kurmanji *et al.*, 2023] Meghdad Kurmanji, Peter Triantafillou, and Eleni Triantafillou. Towards unbounded machine unlearning. *arXiv preprint arXiv:2302.09880*, 2023.

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

[Li *et al.*, 2020] Songze Li, Mingchao Yu, Chien-Sheng Yang, Amir Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Polyshard: Coded sharding achieves linearly scaling efficiency and security simultaneously. *IEEE Transactions on Information Forensics and Security*, 16:249–261, 2020.

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

[Liu *et al.*, 2022] Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1749–1758. IEEE, 2022.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Roth, 2006] Ron M Roth. Introduction to coding theory. *IET Communications*, 47(18-19):4, 2006.

[Salton and Buckley, 1988] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[Shokri *et al.*, 2017] 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)*, pages 3–18. IEEE, 2017.

[Su and Li, 2023] Ningxin Su and Baochun Li. Asynchronous federated unlearning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wang *et al.*, 2020] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE, 2020.

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

[Wang *et al.*, 2023] Weiqi Wang, Zhiyi Tian, Chenhan Zhang, An Liu, and Shui Yu. Bfu: Bayesian federated unlearning with parameter self-sharing. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 567–578, 2023.

[Wu *et al.*, 2022] Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Yaohong Ding. Federated unlearning: Guarantee the right of clients to forget. *IEEE Network*, 36(5):129–135, 2022.

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

[Xu *et al.*, 2023] Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S Yu. Machine unlearning: A survey. *ACM Computing Surveys*, 56(1):1–36, 2023.

[Yan *et al.*, 2022] Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. Arcane: An efficient architecture for exact machine unlearning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4006–4013, 2022.

[Zhang *et al.*, 2023] 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*, 2023.