Federated Unlearning in the Internet of Vehicles

Guofeng Li*, Xia Feng†, Liangmin Wang*¶, Haiqin Wu‡, Boris Düdder§
*{220235267, 101013133}@seu.edu.cn, School of Cyber Science and Engineering ,Southeast University, Nanjing, China
†xiazio@gmail.com, School of Cyberspace Security, Hainan University, Haikou, China
‡hqwu@sei.ecnu.edu.cn, Software Engineering Institute, East China Normal University, Shanghai, China
§boris.d@di.ku.dk, Department of Computer Science, University of Copenhagen, Copenhagen, Denmark
¶corresponding author

***Abstract*—Model training in the Internet of Vehicles (IoV) requires federated unlearning under three scenarios: (1) vehicles want to erase their historical updates to protect their privacy, (2) servers eliminate the influence of dropout vehicles, and (3) servers recover the global model from the poisoning attack launched by malicious clients. However, the existing federated unlearning methods based on retraining, such as FedRecover[1] and FedEraser[2], cannot be applied directly to the IoV scenarios. Firstly, in these schemes, the server needs to store all the local gradients uploaded by clients, resulting in significant storage space occupation. Secondly, their methods assume that all clients engaged in Federated learning (FL) from the beginning and will not exit, which does not align with the characteristics of FL in IoV where vehicles can join and leave FL at any time. To address these challenges, we propose a federated unlearning scheme suitable for IoV scenarios. We use a backtracking mechanism to achieve unlearning instead of reinitializing like other approaches. Then, we build our function over Cauchy mean value theorem to recover the performance of the global model. To reduce the storage burden of the server, we present a novel method that only saves the direction of the local updates, which can spare approximately 95% of storage overhead. The experimental results proved the effectiveness of our scheme that uses just the direction of historical gradients and historical models. Therefore, our work enables federated unlearning to be applied in practical applications in the IoV.**

***Index Terms*—Internet of Vehicles, Federated Learning, Federated Unlearning, Poisoning Attack, Privacy-preserving**

I. Introduction

The rapid development of the Internet of Vehicles (IoVs) has enabled intelligent transportation systems to monitor, coordinate, and optimize traffic in real-time. However, how to use the vast amounts of data generated by the IoV has raised significant concerns regarding privacy and security. Federated learning (FL) has emerged as a promising solution to address these challenges [3]. FL allows clients to collaboratively train a global model without sharing raw data, thereby preserving privacy while optimizing models [4, 5]. In the IoV, Road Side Units (RSU) and vehicles act as servers and clients in FL respectively [6]. In each training round, vehicles first download model parameters from RSU. They then train the global model based on their local datasets. Finally, they upload local parameters, such as gradients, to the nearby RSU. The RSU aggregates received local gradients to update the global model. This process will continue until the model converges.

With the widespread application of FL in IoVs, the concept of federated unlearning [7] has also garnered attention. The server sometimes needs to unlearn certain gradients aggregated from the global model in the past rounds. This is due to three reasons as follows: Firstly, when data from clients is employed in FL, the model may memorize part of them, potentially exposing clients to the risk of a privacy leakage attack [8–10]. Therefore, clients may want to erase their updates from the global model to protect their privacy [11]. Some data protection legislation grants data owners *the right to be forgotten*, such as the General Data Protection Regulation (GDPR) [12] and the Personal Information Protection and Electronic Documents Act (PIPEDA) [13], which requires servers to find a feasible way to forget (erase) clients' data from the model.

Secondly, servers need to eliminate the influence of dropout vehicles. In the IoV scenarios, vehicles may be disconnected from the RSU due to network connection problems, hardware failures, or other technical reasons, resulting in dropouts. Those dropout vehicles may desire the global model to forget the knowledge learned from their private local training data or even their existence. It is worth noting that leaving FL is different from dropping out. Clients who drop out will disconnect from the RSU, whereas those leaving FL may still maintain communication with the RSU.

Finally, servers need to recover the global model from the poisoning attacks, due to FL being susceptible to poisoning attacks. Specifically, malicious clients may upload poisoning updates that can degrade the model's performance [14–17] or implant the backdoor [18, 19]. Existing methods to defend against poisoning attacks either aim to detect them [20–24] or mitigate their impact [25, 26]. However, these methods are not foolproof and attackers may still compromise the model [22, 27]. Therefore, the safest approach is to erase all updates contributed by the attacker from the global model once the attacker is detected. This is because the server cannot be certain whether the attacker launched an undetected attack prior to being discovered. Compared to existing methods, federated unlearning proves more effective in defending against such attacks by completely erasing the negative influence of

This work was supported by National Natural Science Foundation of China under Grant 62272203, the Leading-edge Technology Program of Jiangsu Natural Science Foundation (BK20202001), and the National Natural Science Foundation of China (62372105).

2833-292X/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN-S60304.2024.00034

attackers.

II. Motivations and Challenges

Current federated unlearning methods can be classified into two categories. The first requires initialization of a new global model to fully erase updates of forgotten clients [1, 2, 7, 28]. Then, the server employs historical models and gradients to retrain the new model. The second-class methods can achieve forgetting while still maintaining the performance of the "original" global model on remaining clients, including reverse gradient descent [29], removing gradient residuals[30], channel pruning [31], and so on. However, both of these approaches are not well-suited to the IoVs due to two challenges.

Challenge I: Most methods [1, 2, 28, 30] require servers to store all the gradients uploaded by clients, which may lead to substantial storage overhead, especially in large-scale IoV networks with numerous participating vehicles.

Challenge II: The existing federated unlearning methods do not consider the scenario of clients dynamically joining and exiting FL. Some methods need the assistance of clients to implement federated unlearning and fix errors [1, 2, 28, 32, 33], so they will fail if a substantial number of clients leave FL. Besides, the reinitialization methods for forgetting are impractical in dynamic IoV environments with frequent client participation in FL. When the server needs to erase clients that engaged in FL midway, these methods result in squandering the training outcomes accumulated before the participation of the forgotten clients.

To address **challenge** I, we implement a gradient compression scheme inspired by Li et al. [25]. By preserving the direction of the gradient element which takes up just two bits, we are able to reduce storage overhead by approximately 95%.

To address **challenge** II, we propose a novel approach which can implement federated unlearning in the dynamic environment of IoVs. To the best of our knowledge, we are **the first to introduce** a comprehensive study on the detailed scheme and algorithm of federated forgetting learning in the context of IoVs. We utilize a backtracking-based method as a substitute for re-initialization to erase the updates of the pending forgetting clients from the global model, thereby preserving partial training outcomes. We refer to the backtracked model as the unlearned model. To recover the unlearned model, We then build a recovery function based on the Cauchy mean value theorem, which uses the historical models and gradient direction retained on the server. Our rationale is that during the recovery process, the server estimates the remaining clients' model gradients instead of asking clients to compute and communicate with them.

Besides, we adopt the L-BFGS algorithm [34, 35] to compute an approximate value of the Hessian matrix in the Cauchy mean value theorem, thereby reducing the computational overhead of servers. To ensure the normal initialization of the L-BFGS algorithm, we use the historical information that existed before the forgotten vehicle joined the FL system as the recovered information. Furthermore, we clip the recovered gradients with a threshold, thereby decreasing the error in recovery without the assistance of the online clients.

Our work paves the way for the practical application of federated unlearning in IoV, enabling the safe and efficient unlearning of client updates when necessary. In summary, our main contributions are as follows:

1) With the modified gradient compression scheme of RSA [25], we greatly reduce the requirements of the server storage capacity for federated unlearning in IoVs.
2) We design a forgetting method that can be adapted to scenarios in which vehicles join FL at any time.
3) We propose a recovery scheme that can be conducted on the server side without client engagement, thereby reducing vehicle-side overhead and adapting to scenarios where vehicles leave FL.
4) We conducted experimental simulations to demonstrate that our method can effectively erase updates from forgotten clients and recover the global model.

A. Related Work

Cao et al. [36] first introduced machine unlearning, a method to erase the impact of a data sample on a trained model. They focused on statistical query learning. In this scenario, the unlearning algorithm only needs to recalculate the summations that contain unlearned data to erase their impact from the model. Zhang et al. [37] unitized the Cauchy mean value theorem and the L-BFGS algorithm to retrain the unlearned model as well. Still, they used the same approximate Hessian matrix for all clients, which is ineffective for model recovery in FL [1]. Besides, Bourtoule et al. [11] suggested reducing the cost of model retraining by slicing the training data. However, it is not suited for FL because the servers do not have access to the data. To address the limitations of existing machine unlearning methods, the concept of federated unlearning has emerged. Depending on the implementing methods, they can be divided into two categories: exact unlearning and approximate unlearning.

Exact unlearning. These methods destroy the global model to achieve total forgetting [1, 2, 28, 32]. They then reinitialize a global model and recover it with the historical models and gradients. Liu et al. [2] proposed FedEraser and recovered the global model by adjusting the historical parameter updates of the participating clients. Wu et al. [28] utilized knowledge distillation to recover the global model. Cao et al. [1] also leveraged the Cauchy mean value theorem and the L-BFGS algorithm to recover the global model. Still, they relied on online clients to initialize the L-BFGS algorithm and correct the error. Those proposals will consume significant storage space and do not work when clients leave FL, so they are unsuitable for applications in IoV. Wei et al. [32] achieved unlearning by a rollback mechanism and saved the storage overhead of servers based on a negative sampling method and an importance-based update selection mechanism. However, they realized recovery based on FedEraser [2], so their approach is also unsuitable for scenarios where clients exit the FL.

Approximate unlearning. These methods can achieve forgetting while maintaining the performance of the global model [30, 31]. Wang et al. [31] leveraged TF-IDF [38] to evaluate the relevance between classes and channels, and erased the discriminative channels of the category to unlearn. Zhang et al. [30] erased the impact of a client by removing a weighted sum of gradient residuals from the global model. However, these methods also consume a lot of computational [31] or storage [30] resources. Moreover, they pose a risk whereby attackers may still restructure some information from the unlearned model [39].

III. Preliminaries

A. Federated Learning

Federated Learning (FL) allows distributed clients to collaboratively train a sharing model without exchanging training data [4, 5]. This is achieved by each client training the global model on their local dataset and then sending only the model updates, rather than the data itself, to a central server. A typical FL setup has n clients, each with a local dataset $D_i, i = 1, 2, \cdots, n$. In the tth round, the ith client trains the global model on its local dataset D_i and sends its model updates, such as gradients g_t^i, to the server. Then, the server aggregates the clients' gradients according to an *aggregation rule* $\mathcal{A}$. In our scheme, we use the stochastic gradient descent (SGD)[40] algorithm to calculate the gradients, and FedAvg [4] as our aggregation rule. The server computes the weighted average of the received gradients as the aggregated gradient. Formally, given the model updates $g_t^1, g_t^2, \cdots, g_t^n$ in the tth round, the update of the global model is as in:

$$\mathcal{A}(g_t^1, g_t^2, \cdots, g_t^n) = \frac{1}{\sum_{i=1}^{n} \|D_i\|} \cdot \sum_{i=1}^{n} \|D_i\| \cdot g_t^i \quad (1)$$

where $\|\cdot\|$ represents the size of a dataset. After aggregation, the server updates the tth round global model parameters w_t, as in:

$$w_{t+1} = w_t - \eta \cdot \mathcal{A}(g_t^1, g_t^2, \cdots, g_t^n) \quad (2)$$

where η is the learning rate. In the IoV, RSU serves as the server, while vehicles act as the clients in FL.

B. Federated Unlearning

Federated unlearning is an emerging technique that can erase the updates of clients from a model without accessing the training data [1, 2]. This method serves dual purposes: recovering models from poisoning attacks and ensuring *the right to be forgotten*. The model after unlearning should resemble the one that has been trained for the same number of rounds on remaining clients [39]. Let C denote the set of clients and the forgotten client $i \in C$. We use $C \setminus \{i\}$ to represent the remaining set that does not include the client i after unlearning. Suppose the model trained on the set C is denoted by M^C. After unlearning, the model M^C should be close to the model $M^{C\setminus\{i\}}$ [30]. That means the global model, after unlearning, should maintain a similar prediction accuracy for the remaining clients.

Currently, achieving federated unlearning rapidly and resource-efficiently is an open challenge. A naive way of implementing federated unlearning is to reinitialize a global model and retrain the model from scratch, depending on the remaining clients. However, it is prohibitively costly for clients. Consequently, researchers suggest shifting more of the computational burden of federated unlearning to the server [1, 2, 29]. To this end, some researchers propose that the server should preserve the model and gradients at each round for retraining [1]. Nevertheless, although these methods effectively alleviate the computational load on clients, they lead to a considerable increase in the server storage overhead. Additionally, storing user gradients on the server will introduce extra security vulnerabilities. Hackers could attack the server, access the gradients, and use them to reconstruct sensitive client data [41–43]. Therefore, to effectively apply federated unlearning in the context of IoV, it is crucial to address the aforementioned challenges.

C. RSA for Robust Distributed Learning

Li et al. [25] propose Byzantine-Robust Stochastic Aggregation (RSA) Methods to restrict the negative influence of Byzantine attackers. Suppose each client possesses a local model m_i, while the master server holds the global model m_0. In the tth round, the server and clients update their model as (3) and (4), respectively:

$$m_0^{t+1} = m_0^t - \eta \left(\nabla f_0(m_0^t) + \lambda \left(\sum_{i=1}^{n} sign(m_0^t - m_i^t) \right) \right) \quad (3)$$

$$m_i^{t+1} = m_i^t - \eta \left(\nabla L(m_i^t, \xi_i) + \lambda sign(m_i^t - m_0^t) \right) \quad (4)$$

where ∇L denotes the local gradient and ξ_i represents a data point sampled from the local dataset D_i of client i, $f_0(\cdot)$ is the regularization term, λ is a positive constant, and $sign(\cdot)$ is the element-wise sign function that returns the sign of the input. When the input to the sign function is greater than 0, it returns 1, and when it is less than 0, it returns -1; otherwise, it returns 0. These signs indicated the direction of updates. The server obtains updating directions from clients and controls the step size of global model update through the learning rate η. Li et al. [25]theoretically proved that RSA algorithm can converge to the desirable optimality.

IV. Our Scheme

In this paper, we propose an efficient federated unlearning method suited for IoV. When training, the server needs to record the number of rounds each vehicle participated in FL and save the global model parameters and gradients' direction for each round. Specifically, we defined the direction of a gradient element as 1 when it is greater than a threshold δ, -1 when it is less than the threshold $-\delta$, and 0 when it is between the thresholds. We use a threshold δ to prevent gradient elements with tiny values from being over-amplified

to 1 or -1, which would otherwise lead to degradation of the global model performance after recovery.

When unlearning, the server firstly backtracks the global model to its state prior to the participation of the pending forgetting client in FL. Then it utilizes the L-BFGS algorithm to compute the approximate Hessian matrix and adopt Cauchy mean value theorem to estimate the recovered gradient for other vehicles. Subsequently, the server clips the recovered gradients to limit its error. Finally, the server updates the current unlearned model using (2). Next are the detailed steps.

A. Performing forgetting by backtracking

The server needs to erase the updates of a client from the global model under three circumstances: 1) the vehicle requires erasure, 2) the vehicle drops out from the IoV, and 3) the vehicle is detected to have performed a poisoning attack. Suppose the pending erasure vehicle joined the FL at round F, the current round is T, and the unlearned model is $\overline{w}$. To erase the updates of this client, the server backtracks the model parameters from the current state w_T to the state w_F according to records. Then the server assigns the historical model w_F to the unlearned model $\overline{w}$ as in:

$$\overline{w} = w_F \tag{5}$$

By doing so, the server fully erases the updates uploaded by the client from the global model and keeps the training results from rounds 1 through F.

B. Recovering the global model

To recover the unlearned model w_F, we apply the integral form of the Cauchy mean value theorem[44] to estimate the gradients of other clients in the recovery stage based on historical information. Specifically, set the symbol w_t denotes the historical global model parameters and g_t^i denotes the direction of historical gradients reported by the ith client in the tth round, where $i = 1, 2, \cdots, n$ and $t = 1, 2, \cdots, T$. When retraining, we use $\overline{\mathbf{w}}_t$ to denote the recovered global model parameters and $\overline{g}_t^i$ to denote the recovered gradients of ith client in the tth round. The server estimates the $\overline{g}_t^i$ as in:

$$\overline{g}_t^i = g_t^i + H_t^i \cdot (\overline{w}_t - w_t) \tag{6}$$

where $H_t^i = \int_0^1 H(w_t + z(\overline{\mathbf{w}}_t - w_t))dz$ is an integrated Hessian matrix for the ith client in the tth round and $H(\cdot)$ is a function that computes the second-order derivatives of inputs. Equation (5) reveals the relationship between $\overline{\mathbf{w}}_t - w_t$ and $\overline{g}_t^i - g_t^i$. However, the exact value of the Hessian matrix is hard to calculate, so we leverage the L-BFGS algorithm[34] to compute an approximate value.

Calculating the approximate value of the Hessian matrix. To reduce computational memory overhead, we adopt the L-BFGS algorithm[34] to calculate an approximate value of the Hessian matrix H_t^i. This algorithm requires the differences between the global models and the model updates in the past rounds, referred to as vector pairs[35], to make the approximation in the current round. Specifically, the server needs to compute the difference of the global model parameters as $\Delta w_t = \overline{w}_t - w_t$ and the difference of model updates as $\Delta g_t^i = \overline{g}_t^i - g_t^i$. We input the vector pair $(\Delta W, \Delta G^i)$ to the L-BFGS algorithm which outputs an approximate Hessian matrix $\widetilde{H}_t^i$, where $\Delta W = [\Delta w_{t_1}, \Delta w_{t_2}, \cdots, \Delta w_{t_s}]$ and $\Delta G^i = [\Delta g_{t_1}^i, \Delta g_{t_2}^i, \cdots, \Delta g_{t_s}^i]$. The parameter s is the size of the vector pair.

Note that we need to get s sets of $\overline{w}_t$ and $\overline{g}_t^i$ before estimation. Thus, at the beginning of recovery, we regard the historical information from the $F - s$ round to the $F - 1$ round as the "recovered information". Therefore, the $\Delta W = [w_{F-s} - w_F, w_{F-s+1} - w_F, \cdots, w_{F-1} - w_F]$ and the $\Delta G^i = [g_{F-s}^i - g_F^i, g_{F-s+1}^i - g_F^i, \cdots, g_{F-1}^i - g_F^i]$. This strategy has an advantage over other methods because if vehicles exited the FL before recovery, the server can also estimate their recovered gradients. If some vehicles do not submit enough gradients in rounds from $F-s$ to $F-1$ and are still online in FL, the server could dispatch historical models that correspond with the rounds of the missing gradients to these vehicles, enabling the acquisition of the "recovered gradients".

Moreover, as recovery proceeds, the utilization of outdated vector pairs in the L-BFGS algorithm leads to a gradual divergence between the recovered gradients and the real gradients. This results in inaccurate parameter updates and, consequently, a decline in the model's performance. Therefore, when the model accuracy continuously diminishes, the server must update the vector pairs.

Limiting the error. To inhibit the degradation of model performance caused by the error in the estimation, we propose a gradient clipping method. The estimated gradient $\overline{g}_t^i$ is limited by the following formula:

$$\widetilde{g}_t^i = \overline{g}_t^i / \max\left(1, \frac{|\overline{g}_t^i|}{L}\right) \tag{7}$$

where L is a predefined clipping threshold, and $|\cdot|$ denotes the absolute value of gradient elements. This clipping ensures that if $|\overline{g}_t^i| \leq L$, then $|\overline{g}_t^i|$ is preserved, whereas if $|\overline{g}_t^i| > L$, it gets scaled down to be of threshold L. By applying gradient clipping to mitigate the effect of errors in the recovered gradients on the magnitude of the model update step, we could improve the performance of the recovered model.

Algorithm 1 shows our complete scheme of federated unlearning and algorithm 2 shows the details of the L-BFGS algorithm.

V. Evaluation

In this section, we evaluate the validity of our proposal[1]. The experiments have three goals: **(i)** Check whether our approach can effectively recover the performance of the unlearned model; **(ii)** Verify that our scheme does not introduce new poisoned updates during the process of recovering the poisoned model; **(iii)** Investigate the effect of hyperparameters L and δ on model recovery, respectively. We simulate the scenario of IoVs and set the number of clients $n = 100$.

[1]Our code is available in https://github.com/weixiyuluo/FUIOV

Algorithm 1: Federated Unlearning in IoVs

Input: Vehicles set $\mathcal{V}$; historical global model parameters $w_F, w_{F+1}, \ldots, w_T$; historical gradient direction $g_F^i, g_{F+1}^i, \ldots, g_T^i$, where i in $\mathcal{V}$; the vector pairs $(\Delta W, \Delta G^i)$; clipping threshold L; aggregation rules $\mathcal{A}$; learning rate η.

Output: Recovered model $\overline{w}_T$

$\overline{w}_F \leftarrow w_F$; // backtrack the global model

for $t = F, F+1, \ldots, T$ **do**

  // update the vector pairs if needed

  for i *in* $\mathcal{V}$ **do**

    $\widetilde{H}_t^i \leftarrow$ L-BFGS$((\Delta W, \Delta G^i)$

    $\overline{g}_t^i = g_t^i + \widetilde{H}_t^i(\overline{w}_t - w_t)$

    $\widetilde{g}_t^i \leftarrow \overline{g}_t^i / \max\left(1, \frac{|\overline{g}_t^i|}{L}\right)$

  end

  $\overline{w}_{t+1} \leftarrow \ w_t - \eta \cdot \mathcal{A}\left(\widetilde{g}_t^1, \widetilde{g}_t^2, \ldots, \widetilde{g}_t^n\right)$

end

return $\overline{w}_T$

Algorithm 2: L-BFGS

Input: Vector pair $(\Delta W, \Delta G^i)$.

Output: Approximated Hessian matrix $\widetilde{H}_t^i$.

$A = \Delta W^T \Delta G^i$

$L = tril(A)$; // Lower triangular matrix of A

$D = diag(A)$; // Diagonal matrix of A

$\sigma = (\Delta g_{s-1}^T \Delta w_{s-1}) \setminus (\Delta w_{s-1}^T \Delta w_{s-1})$

$$p = \begin{bmatrix} -D & L^T \\ L & \sigma \Delta W^T \Delta W \end{bmatrix}^{-1} \begin{bmatrix} (\Delta G^i)^{\mathrm{T}} \\ \sigma \Delta W^T \end{bmatrix}$$

$\widetilde{H}_t^i = \sigma - \begin{bmatrix} \Delta G & \sigma \Delta W \end{bmatrix} p$

return $\widetilde{H}_t^i$

A. Experiment Setup

1) *Datasets and Models*: In this paper, we utilize Convolutional Neural Networks (CNN) models to experiment with two real-world datasets, namely MNIST[45] and GTSRB[46]. The MNIST dataset is one of the most widely used benchmarks in FL. The GTRSB dataset serves as a widely recognized benchmark for the task of traffic sign recognition, which plays a critical role in the domain of connected vehicles and autonomous driving systems.
   - **MNIST [45].** Each image in this dataset is categorized into one of ten classes representing the digits 0 through 9. We adopt the CNN model as our global model architecture. The model consists of two convolutional layers and two fully-connected layers. We train 100 rounds with a learning rate 1×10^{-4} and batch size 128.
   - **GTRSB** [46]. Each image in this dataset shows a traffic sign in real-world conditions, varying in angle, lighting, and seasonal changes. We adopt a CNN model as our global model architecture comprising two convolutional layers and one fully connected layer. We train 100 rounds with a learning rate 1×10^{-3} and batch size 128.
2) *Attack Setting*: We randomly sample 20% of clients as malicious ones. Moreover, we perform two types of poisoning attacks, the label flip attack and the backdoor attack, based on the MNIST dataset.
   - **Label flip attack** [17]. In this attack, adversaries change the labels of a subset of the training data, essentially "flipping" them to incorrect values. Specifically, we altered the labels for images that originally represented the number '7' to a target label '1'.
   - **Backdoor attack** [18]. The backdoor attacker inserts a trigger in the training data. This hidden trigger results in the model making incorrect predictions when the trigger appears, but performs normally otherwise. We introduced a 3x3 pixel-sized black square as a trigger into a random selection of images from the MNIST dataset. These images were then relabeled with the target class '2'.
3) *Recovery Setting and Comparison Metrics*: We apply the same settings as the original FL training in recovery, such as the learning rate and the aggregation rule. Our scheme has the following parameters: the round F that the forgotten client joined FL is 2, the current round T is 100, the size s of vector pairs is 2, the threshold for the preservation phase is $1 \times e^{-6}$ and the clip threshold L is 1. The number of retraining rounds in recovery equals $T - F$ and the vector pairs are updated every 21 rounds. To evaluate the quality of recovery, we compare our method with three baselines:
   - **Retraining.** The server removes the pending forgetting client and retrains a new model from scratch. The training process will last 100 rounds to ensure a robust comparison.
   - **FedRecover [1].** This method also leverages the Cauchy mean value theorem and the L-BFGS algorithm to recover the global model. However, when estimation, the server uses the complete gradients rather than just the direction of gradients. We set the server to get the real gradients from the online clients every 20 rounds.
   - **FedRecovery [30].** This method removes a weighted sum of gradient residuals from the global model to achieve unlearning and adds the Gaussian noise to make the unlearned model and retrained model statistically indistinguishable.

   We assume that vehicles do not exit FL in the comparison methods. The recovery metrics are *test accuracy* and *attack success rate*. The *test accuracy* is the prediction accuracy rate of a model on test images and the *attack success rate* is the probability that the model recognizes the poisoned image as the target label of the malicious attacker. Furthermore, when investigating one hyperparameter, the other hyperparameter remains unchanged.

B. Results and Analysis

1) *The Effectiveness of Recovery*. We compare our scheme with other methods after achieving unlearning, and the

results are shown in Table I. It displays that our scheme can effectively restore the performance of the unlearned model. From this table, we can observe that the recovered model performance of our scheme is slightly lower than that of retraining and FedRecover[1], but marginally superior to FedRecovery[30]. It shows that just using the historical models and the direction of gradients is enough to recover the unlearned model. We believe that this minor reduction in performance is acceptable, considering the significant storage overhead savings provided by our approach.

2) *The Effectiveness of Unlearning*. Fig. 1 exhibits that the poisoned model is restored to a normal form after forgetting and recovery. Before performing unlearning, the attack success rates are 56% and 41%, respectively. After forgetting, the attack success rates both drop to less than 1%. It illustrates that our scheme can effectively erase client updates from the global model. Moreover, the attack success rates do not exhibit an obvious increase after the recovery process. It demonstrates that our scheme does not introduce poisoned updates during recovery.
3) *The Hyperparameter Analyses of Thresholds*. Fig. 2 showcases the effect of varying values of the threshold L on the model performance post-recovery. The experiment suggests that setting the parameter L to 1 achieves the optimal model performance, with an accuracy of 86%. When L is smaller, it restricts the step size during model updates, which will slow the model's recovery process. On the other hand, a larger L value intensifies the effect of any errors in the recovered gradients, thereby reducing the model's accuracy. Fig. 3 presents the effect of different values of the threshold δ on the model performance after recovery, indicating that the optimal value for δ is 10^{-6} where the accuracy of the model is 86%. The observed decline in model performance with increasing values of δ can be attributed to the substantial loss of updated information during the gradient-saving process. Essentially, larger δ values result in more gradient elements being saved as 0, thereby discarding their updated information and affecting the model negatively. Conversely, as the value of δ decreases, elements of the gradient with a negligible influence on the model update are assigned a value of 1 or -1. This amplifies their original impact on the model update, which can also consequently lead to a decrease in model performance.

TABLE I
THE ACCURACY OF UNLEARNING METHODS, COMPARISON

Datasets	Global model's accuracy			
	Retraining	FedRecover	FedRecovery	Ours
MNIST	0.873	0.869	0.825	0.859
GTRSB	0.837	0.766	0.702	0.747

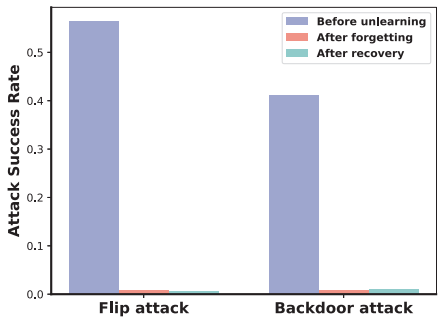

Fig. 1. The attack success rate on the MNSIT dataset before unlearning, after forgetting, and after recovery

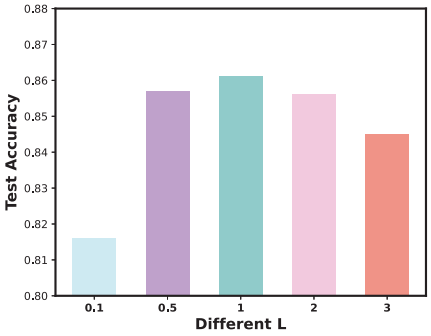

Fig. 2. Impact of different hyperparameter L values on MNIST dataset model accuracy when $\delta = 10^{-6}$.

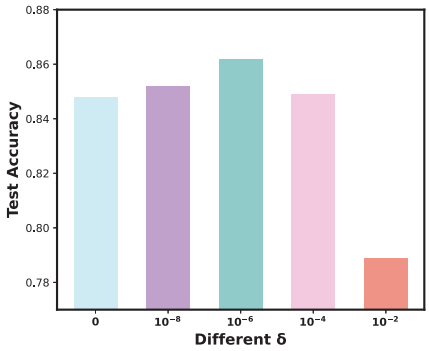

Fig. 3. Impact of distinct hyperparameter δ values on MNIST dataset model accuracy when $L = 1$

VI. CONCLUSION

In this paper, we propose a storage method that preserves the gradient direction, which can reduce the storage overhead of servers by up to 95%. Moreover, by recording only the gradient direction instead of the complete gradients, we can prevent hackers from reconstructing clients' personal information using stored information. Besides, we propose a forgetting method and a recovery scheme to achieve federated unlearning, which can adapt to the dynamic characteristics of IoVs. The experimental results proved the effectiveness of our scheme. To further validate the scalability and practicality of our proposed approach, we plan to evaluate its performance in the Internet of Things(IoTs) scenarios. In conclusion, our work paves the way for the industrial application of Federated Unlearning in IoV by providing a practical solution to the storage and dynamicity challenges.

References

[1] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "Fedrecover: Recovering from poisoning attacks in federated learning using historical information," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 1366–1383.

[2] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, "Federaser: Enabling efficient client-level data removal from federated learning models," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 2021, pp. 1–10.

[3] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5341–5351, 2021.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, jan 2019.

[6] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5341–5351, 2021.

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

[8] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 739–753.

[9] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 691–706.

[10] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2430–2444, 2020.

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

[12] A. Mantelero, "The eu proposal for a general data protection regulation and the roots of the 'right to be forgotten'," *Computer Law & Security Review*, vol. 29, no. 3, pp. 229–235, 2013.

[13] Office of the Privacy Commissioner of Canada, "Announcement: Privacy commissioner seeks federal court determination on key issue for canadians' online reputation," Oct. 2018. [Online]. Available: https://www.priv.gc.ca/en/opc-news/news-and-announcements/2018/an_181010/

[14] J. Chen, H. Zheng, M. Su, T. Du, C. Lin, and S. Ji, "Invisible poisoning: Highly stealthy targeted poisoning attack," in *Information Security and Cryptology*, Cham, 2020, pp. 173–198.

[15] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning(ICML)*. International Machine Learning Society, 2019.

[16] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proceedings of the 29th USENIX Conference on Security Symposium(USENIX Security)*, USA, 2020.

[17] E. Rosenfeld, E. Winston, P. Ravikumar, and J. Z. Kolter, "Certified robustness to label-flipping attacks via randomized smoothing," in *Proceedings of the 37th International Conference on Machine Learning(ICML)*, 2020.

[18] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2088–2105, 2021.

[19] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 11 957–11 965.

[20] S. Shen, S. Tople, and P. Saxena, "Auror: defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications(ACSAC)*, New York, NY, USA, 2016, p. 508–519.

[21] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining(KDD)*, New York, NY, USA, 2022, p. 2545–2555.

[22] X. Mu, K. Cheng, Y. Shen, X. Li, Z. Chang, T. Zhang, and X. Ma, "Feddmc: Efficient and robust federated learning via detecting malicious clients," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–16, 2024.

[23] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems(NIPS)*, Red Hook, NY, USA, 2017, p. 118–128.

[24] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrap-

ping," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2021.

[25] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[26] X. Cao, J. Jia, and N. Z. Gong, "Provably secure federated learning against malicious clients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 8, 2021, pp. 6885–6893.

[27] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proceedings of the 29th USENIX Conference on Security Symposium(USENIX Security)*, USA, 2020.

[28] C. Wu, S. Zhu, and P. Mitra, "Federated unlearning with knowledge distillation," *arXiv preprint arXiv:2201.09441*, 2022.

[29] P. Wang, Z. Yan, M. S. Obaidat, Z. Yuan, L. Yang, J. Zhang, Z. Wei, and Q. Zhang, "Edge caching with federated unlearning for low-latency v2x communications," *IEEE Communications Magazine*, pp. 1–7, 2023.

[30] L. Zhang, T. Zhu, H. Zhang, P. Xiong, and W. Zhou, "Fedrecovery: Differentially private machine unlearning for federated learning frameworks," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4732–4746, 2023.

[31] J. Wang, S. Guo, X. Xie, and H. Qi, "Federated unlearning via class-discriminative pruning," in *Proceedings of the ACM Web Conference 2022(WWW)*, New York, NY, USA, 2022, p. 622–632.

[32] W. Yuan, H. Yin, F. Wu, S. Zhang, T. He, and H. Wang, "Federated unlearning for on-device recommendation," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining(WSDM)*, New York, NY, USA, 2023, p. 393–401.

[33] A. Halimi, S. R. Kadhe, A. Rawat, and N. B. Angel, "Federated unlearning: How to efficiently erase a client in fl?" in *International Conference on Machine Learning(ICML)*, 2022.

[34] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[35] H. Tankaria, S. Sugimoto, and N. Yamashita, "A regularized limited memory bfgs method for large-scale unconstrained optimization and its efficient implementations," *Computational Optimization and Applications*, vol. 82, no. 1, pp. 61–88, 2022.

[36] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 463–480.

[37] Y. Wu, E. Dobriban, and S. B. Davidson, "DeltaGrad: rapid retraining of machine learning models," in *Proceedings of the 37th International Conference on Machine Learning(ICML)*, 2020.

[38] J. H. Paik, "A novel tf-idf weighting scheme for effective ranking," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR)*. New York, NY, USA: Association for Computing Machinery, 2013, p. 343–352.

[39] H. Xu, T. Zhu, L. Zhang, W. Zhou, and P. S. Yu, "Machine unlearning: A survey," *ACM Comput. Surv.*, vol. 56, no. 1, aug 2023.

[40] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Heidelberg, 2010, pp. 177–186.

[41] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems(NIPS)*, vol. 32, 2019.

[42] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security(CCS)*, New York, NY, USA, 2017, p. 603–618.

[43] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 691–706.

[44] S. Lang, *Real and Functional Analysis*. Springer, 1993, ch. 4, p. 341.

[45] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[46] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *International Joint Conference on Neural Networks*, no. 1288, 2013.