Machine Unlearning via Representation Forgetting With Parameter Self-Sharing

Weiqi Wang, *Student Member, IEEE*, Chenhan Zhang, *Student Member, IEEE*, Zhiyi Tian, *Student Member, IEEE*, and Shui Yu, *Fellow, IEEE*

***Abstract*— Machine unlearning enables data owners to remove the contribution of their specified samples from trained models. However, existing methods fail to strike an optimal balance between erasure effectiveness and model utility preservation. Previous studies focused on removing the impact of user-specified data from the model as much as possible to implement unlearning. These methods usually result in significant model utility degradation, commonly called catastrophic unlearning. To address the issue, we systematically consider machine unlearning and formulate it as a two-objective optimization problem that involves forgetting the erased data and retaining the previously learned knowledge, highlighting accuracy preservation during the unlearning process. We propose an unlearning method called representation-forgetting unlearning with parameter self-sharing (RFU-SS) to achieve the two-objective unlearning goal. Firstly, we design a representation-forgetting unlearning (RFU) method that aims to remove the contribution of specified samples from a trained representation by minimizing the mutual information between the representation and the erased data. The representation is learned using the information bottleneck (IB) method. RFU is tailored to the IB structure models for ease of introduction. Secondly, we customize a parameter self-sharing structural optimization method for RFU (i.e., RFU-SS) to simultaneously optimize the forgetting and retention objectives to find the optimal balance. Extensive experimental results demonstrate a significant effectiveness improvement of RFU-SS over the state-of-the-art methods. RFU-SS almost eliminates catastrophic unlearning, reducing model accuracy degradation from over 6% to less than 0.2% on the MNIST dataset with an even better removal effect. The source code is available at https://github.com/wwq5-code/RFU-SS.git.**

***Index Terms*— Machine unlearning, representation forgetting, multi-objective optimization, machine learning.**

I. Introduction

MACHINE unlearning draws lots of research attention as the solution to the "right to be forgotten", which entitles individuals to revoke the trace of their specified data samples from trained machine learning (ML) models. Retraining ML models from scratch with advanced fast retraining methods [1], [2], [3], [4] can be effective. However, these approaches may incur prohibitive costs in terms of storage and computation, especially for complex tasks or frequent data removal requests [5], [6]. Additionally, cumulatively removing data from a model can lead to a significant decrease in accuracy, known as catastrophic unlearning [7]. Therefore, it is a crucial and challenging research question to find an effective and efficient way to eliminate the impact of erased data from trained models while preserving the model utility.

A few works proposed approximate unlearning methods to address the problem of removing data's impact from trained models. For example, Guo et al. [8] and Sekhari et al. [9] introduced certified removal, a technique based on the Hessian matrix and the Newton update approach [10]. Their methods aim to make the unlearned model be $\epsilon-$indistinguishable from the model that retrained based on the remaining dataset except for the erased samples. Fu et al. [11] and Nguyen et al. [7] introduced knowledge removal for unlearning Bayesian inference models [12]. Both the Hessian-based unlearning methods [8], [9], [13] and Bayesian inference unlearning methods [7], [11], [14] treat unlearning as a single-objective optimization problem, often leading to significant utility degradation in the unlearned model. In order to mitigate model utility degradation caused by unlearning, these methods usually use a fixed threshold to limit the extent of unlearning. However, this approach cannot achieve an optimal balance between forgetting effectiveness and keeping model accuracy.

In this paper, we aim to solve the catastrophic accuracy degradation problem in machine unlearning. We systematically address both erasure effectiveness and model utility preservation in unlearning by formulating it as a two-objective optimization problem. The two objectives are to maximize the removal of erased sample contributions from trained models (forgetting) while retraining the knowledge learned from the remaining dataset as much as possible in the models (remembering). The main process of the two-objective unlearning is shown in Figure 1. We propose representation-forgetting unlearning with parameter self-sharing (RFU-SS) to achieve the two-objective unlearning goals. First, we design a representation-forgetting unlearning (RFU) method, which minimizes the mutual information between the learned representation and the specified samples to implement the forgetting objective. Here, the already-learned representation of the original model is trained utilizing the information bottleneck (IB) method [15], [16], and we introduce our RFU method following this IB structure for ease of understanding. Second,

Manuscript received 10 May 2023; revised 5 September 2023 and 29 October 2023; accepted 31 October 2023. Date of publication 8 November 2023; date of current version 23 November 2023. This work was supported in part by Australia under Grant ARC DP200101374 and Grant LP190100676. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Paolo Gasti. *(Corresponding author: Zhiyi Tian.)*

The authors are with the School of Computer Science, University of Technology Sydney, Ultimo NSW 2007, Australia (e-mail: Weiqi.Wang-2@student.uts.edu.au; chenhan.zhang@student.uts.edu.au; zhiyi.tian@student.uts.edu.au; shui.yu@uts.edu.au).

Digital Object Identifier 10.1109/TIFS.2023.3331239

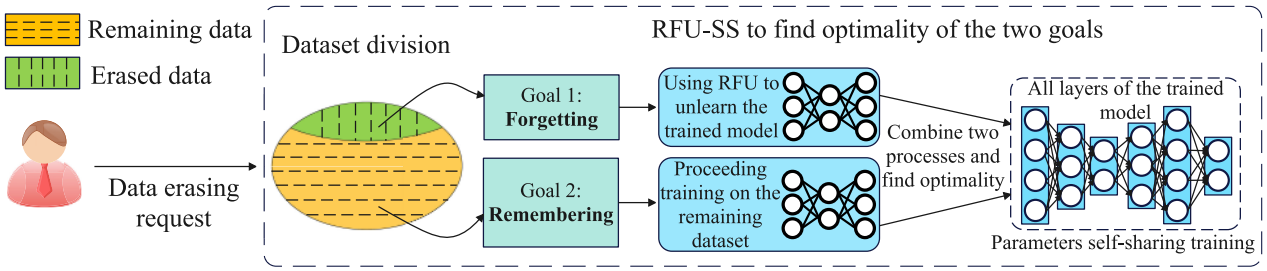


Fig. 1. Process of a two-objective unlearning problem. First, an unlearning process usually begins with a user's data erasure request. Then, the ML server divides the full dataset into the erased data and the remaining data according to the request. Two-objective unlearning has two optimization goals: **forgetting** the contribution of the erased data and **remembering** the knowledge learned from the remaining data. We propose RFU to achieve the forgetting purpose and continue training the model using the samples from the remaining dataset to achieve the remembering purpose. To find the optimal balance between the two goals, we introduce parameters self-sharing into RFU (RFU-SS) to optimize the two goals on the same model simultaneously.

we present a parameter self-sharing structural optimization method to simultaneously optimize the forgetting and remembering objectives to mitigate the accuracy degradation caused by unlearning. This idea is inspired by multi-task learning (MTL) [17], [18]. RFU-SS shares the entire parameters of the model to optimize the forgetting and remembering objectives via gradient descent simultaneously. Since it does not have task-specific layers like MTL, we call it parameter self-sharing. We moreover introduce a simple clipping skill that helps to find the steepest descent direction.

To evaluate the proposed unlearning method, we perform experiments using a well-established approach known as "inference via backdoor" [19], enabling direct verification of whether the model has successfully unlearned the backdoored-erased samples. Specifically, the designed backdoored samples are mixed into the original training dataset for the ML model training, and the unlearning target is to erase these backdoors from the trained model. The effectiveness of unlearning is measured through backdoor attacks. A low backdoor attack accuracy implies a high unlearning performance. Extensive experimental results demonstrate that RFU-SS significantly improves the erasure effect and mitigates the accuracy degradation during unlearning training. To effectively remove the influence of backdoored samples, decreasing the backdoor accuracy to less than 5%, the existing methods easily cause a model accuracy degradation over 6%. However, RFU-SS can mitigate the accuracy degradation within 0.2% on MNIST with an even better removal effect.

To sum up, our contributions are:

- We systematically consider the unlearning problem and transform the traditional unlearning definition into a two-objective optimization problem, including forgetting erased samples and remembering the before-learned knowledge. In this way, we highlight the importance of utility preservation during unlearning to prevent catastrophic unlearning.
- We propose representation forgetting unlearning (RFU) to implement the forgetting objective for trained IB models. Moreover, an unlearning rate is introduced to make RFU adaptive to different unlearning tasks.
- We introduce parameter self-sharing to RFU (i.e., RFU-SS) to find the balance optimality of the formalized two-objective unlearning problem. It effectively removes the impact of backdoored samples (reducing backdoor attack accuracy to less than 5%) with an accuracy loss of less than 0.2% on both MNIST and CIFAR10.
- Extensive experiments have been conducted to validate the proposed RFU-SS. The results demonstrate that RFU-SS is superior in terms of erasure effectiveness and ability to prevent catastrophic unlearning when compared with state-of-the-art solutions.

A. Roadmap

The rest of the paper is structured as follows. We review the related work in Section II. Preliminary knowledge about the tools we used is introduced in Section III. In Section IV, we formalize the two-objective unlearning problem and introduce the solution in detail. Our experimental results and comparison with related work are presented in Section V. Subsequently, in Section VI, we summarize the paper.

II. Related Work

The concept of machine unlearning has been introduced as an effective approach for removing the impact of specific data on a trained model in an efficient manner [1], [20], [21]. A straightforward intuition for implementing machine unlearning is retraining the entire ML model with the dataset after erasing the data that needs to be forgotten. However, this incurs significant computational and storage expenses, particularly in intricate deep-learning scenarios. Some studies, called fast retraining, tried to improve the retraining method by splitting the learning process and training dataset to reduce the unlearning computation cost [2], [3], [22], [23]. However, it is still unsuitable when unlearning requests are frequent or when erased samples are dispersed across multiple split subsets.

Furthermore, approximate unlearning [9], [13], [24], [25] offers an alternative solution. It focuses on unlearning an approximate model solely utilizing the erased dataset, substantially diminishing computational and storage requirements. Guo et al. [8] proposed a certified-removal mechanism similar to differential privacy. The differential privacy ensures the queried outputs cannot be distinguished from the data with or without one specific sample. They hope the unlearning mechanism also guarantees that a model, after executing unlearning, cannot be distinguished from a model that was retrained without the erased data. data. Sekhari et al. [9] introduced a similar solution, which removes the contribution of the erased data from model gradients during the unlearning process. They bounded the model parameters by their differential unlearning definition to mitigate the unlearning

TABLE I
BASIC NOTATIONS

Notations	Descriptions
$D = (X, Y)$	the full training dataset D includes inputs X and labels Y
$D_e = (X_e, Y_e)$	the erased dataset D_e includes inputs X_e and labels Y_e
$D_r = (X_r, Y_r)$	the remaining data D_r includes inputs X_r and labels Y_r
Z	the compressive representation
$p(Z\|X)$	the representation posterior learned based on X
$p(Z\|X_r)$	the naively retrained representation posterior learned based on X_r
$p(Z\|X_{-X_e})$	the unlearned representation posterior based on X_e by RFU
x, z, y	the persample from X, Z, Y
$\mathcal{M}(\theta^r, \theta^a)$	the representer θ^r and the approximator θ^a of an IB model $\mathcal{M}$
θ^r_{fix}	the fixed representer of a trained IB model
θ^a_{fix}	the fixed approximator of a trained IB model
$\mathcal{L}_{rep}$	the representation loss to train the representer θ^r
$\mathcal{L}_{app}$	the approximation loss to train the approximator θ^a
$\mathcal{L}^u_{rep}$	the representation unlearning loss to unlearn the representer θ^r
$\mathcal{L}^u_{app}$	the approximation unlearning loss to unlearn the approximator θ^a
β	the Lagrange multiplier for training an IB model
β_u	the unlearning rate of RFU
α^u	the tradeoff weight in RFU-SS for forgetting and remembering

accuracy reduction. Another Hessian-based method was introduced in [26], where the authors trained a set of linear user weights and a set of non-linear core weights. They implemented model unlearning with strict limits on the amount of remaining information to avoid dramatic accuracy degradation. degradation. Neel et al. [27] proposed unlearning methods allowing arbitrary sequences of updates, including addition and deletion, and defined the corresponding weak and strong unlearning algorithms.

There are also some approximate unlearning methods using methods besides the Hessian matrix. In [7], the authors unlearned an approximate posterior based on the erased data via the variational Bayesian inference [12]. Similarly, Fu et al. [11] proposed a machine unlearning algorithm for Markov Chain Monte Carlo (MCMC) and further defined a knowledge removal estimator to evaluate the knowledge erasing. Nguyen et al. [14] moreover explained the effectiveness of the approximate machine unlearning algorithm from the perspective of Markov Chain Monte Carlo. However, these methods approach machine unlearning as an optimization problem with a single objective and typically rely on bounded-range values or thresholds to restrict the extent of unlearning to avoid significant degradation of utility. It is hard to directly achieve a suitable fixed threshold value to achieve the optimal solution for both the forgetting effect and model accuracy preservation. Therefore, in this paper, we formulate machine unlearning as a two-objective optimization problem and propose a corresponding method to find the optimal solution.

III. PRELIMINARY

Before introducing the background knowledge, we first summarize the primary notations used in the paper, as presented in Table I. In the full training dataset $D = (X, Y)$, X denotes the inputs, Y denotes the labels. We use Z to denote the compressive representation of an IB-trained model, which maximizes the distortion of input information while preserving as much information relevant to the labels. We denote the representation posterior learned from X as $p(Z|X)$, and the unlearned representation posterior, which was learned from X excluding the information of X_e, as $p(Z|X_{-X_e})$. In the training process, we use x, z, y as an example sampled from X, Z, Y. In the learning algorithms, we have two losses: the representation loss $\mathcal{L}_{rep}$ for the representer θ^r, and the predicting approximation loss $\mathcal{L}_{app}$ for the approximator θ^a. We use the Lagrange multiplier β to control the tradeoff between the representer and approximator. In the solution of forgetting objective, RFU, we have the unlearning loss $\mathcal{L}^u$, which is combined by the representation unlearning loss $\mathcal{L}^u_{rep}$ to unlearn the representer and the approximation unlearning loss $\mathcal{L}^u_{app}$ to unlearn the approximator. In RFU, we also introduce an unlearning rate β_u to control the tradeoff between entirely forgetting the contribution of erased samples and not entirely forgetting the original representation trained from the full data in both representation and prediction forgetting loss functions.

A. Information Bottleneck

Our unlearning methods aim to remove the contribution of the erased samples from models trained using Information bottleneck (IB) [15], [16]. IB aims to find a compact encoding distribution sufficient for the target machine learning service while being maximally rate-distortive from the original data. Traditional IB [15], [28] approaches for optimizing the IB objective are based on the iterative Blahut Arimoto algorithm [29]. The IB objective can be described as follows,

$$\mathcal{L}_{IB} = \beta I(Z; X) - I(Z; Y), \tag{1}$$

where $I(Z; X)$ is the mutual information between the encoded representation Z and inputs X and $I(Z; Y)$ denotes the mutual information between Z and Y. β is the Lagrange multiplier that controls the distortion ratio of X. Traditional IB is infeasible to be applied to deep neural networks as it is hard to calculate the exact mutual information [16]. A prevalent solution involves utilizing variational inference to optimize the IB function, known as the Variational Information Bottleneck (VIB) [16], [30], [31]. The VIB method trains the model of the IB objective by proposing a variational distribution of representation to calculate the bounds of the two mutual information terms in Eq. (1). First, it uses a representer θ^r to distort the original input X into a compressive space Z. Second, it uses an approximator θ^a to predict Y based on the compressed Z, which is shown in the upper half of Figure 3. The corresponding approximation loss function is $\mathcal{L}_{app} = -I(Z; Y)$ and representation loss function is $\mathcal{L}_{rep} = \beta I(Z; X)$. And the VIB model is denoted as $\mathcal{M}(\theta^r, \theta^a)$. The experimental optimizing loss function of a VIB model is described as

$$\mathcal{L} = \mathcal{L}_{app} + \mathcal{L}_{rep} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{z\backsim p_{\theta^r}(z|x_i)}[-\log\ p_{\theta^a}(y_i|z)] + \\ + \beta D_{KL}[p_{\theta^r}(z|x_i)||\prod_{i}^{|z|} q_i(z_i)]. \tag{2}$$

The Eq. (1) of IB objective can be optimized using Eq. (2) in a variational form, where D_{KL} is the Kullback-Leibler divergence (KLD) [32]. In this paper, we focus on finding the solution to unlearn these models trained based on the IB objective. The IB-trained models aim to find a representation Z

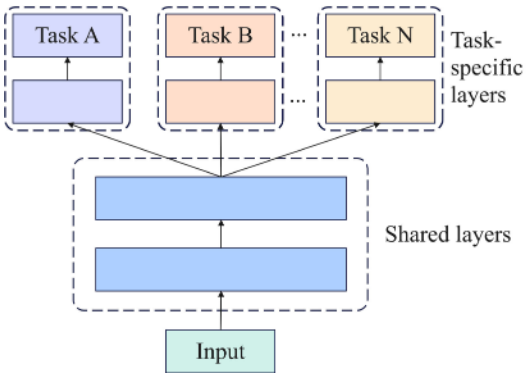


Fig. 2. Hard parameter sharing for multi-task learning. This method shares most of the model, all the hidden layers, as the shared parameters. After the hidden layers that are shared across tasks, there are multiple output layers, each specific to a particular task.

that maximizes the distortion of original data while preserving sufficient information for the target. Our solutions will further remove the information about the erased dataset from the learned representation and will be introduced later.

B. Hard Parameters Sharing

Our parameter self-sharing technique is inspired by the hard parameter sharing used in multi-task learning (MTL) [18], [33], [34]. There are two main solutions in the MTL domain: hard parameter sharing [35], [36] and soft parameter sharing [37], [38]. Hard parameter sharing is commonly implemented by sharing the hidden layers of the model across different learning tasks while maintaining independent task-specific output layers, which is shown in Figure 2. We consider a multi-task learning problem defined over an input space $\mathcal{X}$ and a set of task spaces $\{\mathcal{Y}^t\}_{t\in[T]}$. MTL involves learning multiple tasks based on a large dataset consisting of N data points, such as $\{x_i, y_i^1, \ldots, y_i^T\}_{i\in[N]}$, where x_i is the input and y_i^t is the label of the t-th task. When solving the task t, the hard parameter sharing can be considered as the optimizing task $f^t(x;\theta^{sh},\theta^t) : \mathcal{X} \to \mathcal{Y}^t$, where the parameters (θ^{sh}) are the shared layers used by different tasks together. Task predict layer parameters (θ^t) are the task-specific output layers. The hard parameter sharing solution of MTL can be generally described as the following empirical risk minimization:

$$\min_{\theta^{sh},\theta^1,\ldots,\theta^T} \sum_{t=1}^{T} \alpha^t \mathcal{L}^t(\theta^{sh},\theta^t), \tag{3}$$

where α^t are some static or dynamic weights computed for the empirical loss $\mathcal{L}(\theta^{sh},\theta^t)$ of the task t. Each task loss function can be expanded as $\mathcal{L}(\theta^{sh},\theta^t) \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f^t(x_i;\theta^{sh},\theta^t), y_i^t)$.

C. Multiple Gradient Descent Algorithm

It is possible to convert a MTL problem into a multi-objective optimization problem [18] and to deal with it via gradient descent as in the single-objective case to local optimality. This solution is named the multiple gradient descent algorithm (MGDA) [18], [39]. It is based on the Karush-Kuhn-Tucker (KKT) [40] conditions, which are necessary for optimality [39], [41], [42]. We can state the KKT condition for both task-specific and shared parameters in the MTL problem as,

- There exist $\alpha^1, \ldots, \alpha^T \leq 0$ such that $\sum_{t=1}^{T}\alpha^t = 1$ and $\sum_{t=1}^{T}\alpha^t \nabla_{\theta^{sh}}\mathcal{L}^t(\theta^{sh},\theta^t) = 0$
- For all tasks t, $\nabla_{\theta^t}\mathcal{L}^t(\theta^{sh},\theta^t) = 0$

Any solution that meets these conditions is referred to as a Pareto stationary point [18], [39]. It should be noted that although every Pareto optimal point is Pareto stationary, the converse may not hold. To meet these conditions, we can consider it as an optimization problem

$$\begin{aligned} \min_{\alpha^1,\ldots,\alpha^T} \quad & ||\textstyle\sum_{t=1}^{T}\alpha^t \nabla_{(\theta^{sh},\theta^t)}\mathcal{L}^t(\theta^{sh},\theta^t)||_2^2, \\ \text{s.t.} \quad & \textstyle\sum_{t=1}^{T}\alpha^t = 1, \alpha^t \geq 0 \quad \forall t. \end{aligned} \tag{4}$$

Sener and Koltun [18] and Désidéri [39] demonstrated that either the solution to this optimization problem is 0 and the result satisfies the KKT conditions, or the solution gives a descent direction that can improve all tasks. Based on these former studies, we proposed the parameter self-sharing method to optimize the two goals of unlearning.

IV. Representation Forgetting Unlearning With Parameter Self-Sharing

In this section, we introduce the proposed method of RFU-SS. We organize the section as follows: First, we formalize the machine unlearning problem as a two-objective optimization problem in section IV-A. Second, we present an overview of our method, RFU-SS, in Section IV-B. We then delve into RFU, addressing the forgetting objective for IB-trained models, in Section IV-C. Additionally, we introduce RFU-SS, which tackles the two-objective optimization unlearning problem, in Section IV-D.

A. Problem Definition of Two-Objective Unlearning

Machine unlearning removes the contribution of user-specified samples from the already-trained models. We follow the IB model structure to train the original models, which divides the model into a representer and an approximator. And we design separate unlearning losses for removing the information of the erased data from the learned representation Z from both the representer and approximator on an IB-trained model. Implementing model learning and unlearning based on the IB structure can help us understand the training process more clearly.

When an unlearning request comes, the full training dataset $D = (X, Y)$ can be divided into the erased dataset $D_e = (X_e, Y_e)$ and the remaining dataset $D_r = (X_r, Y_r)$. We assume $(X, Y) = (X_r, Y_r)\bigcup(X_e, Y_e)$ and $(X_r, Y_r)\bigcap(X_e, Y_e) = \emptyset$. We also assume the IB learning algorithm $\mathcal{A}$ to learn the model $\mathcal{M}(\theta^r, \theta^a)$ with a representer θ^r and an approximator θ^a based on the full dataset D. The model learns a representation Z, which minimizes $I(X;Z)$ and maximizes $I(Y;Z)$, as we introduced in preliminary. Normally, after receiving a user's erasure request of the specified dataset D_e, the ML server will execute an unlearning algorithm $\mathcal{U}$ to remove the trace of D_e while keeping the knowledge learned from the remaining dataset D_r. After unlearning, the unlearned model $\mathcal{M}_u(\theta^r, \theta^a)$'s distribution is expected to equal the distribution

of the model that retrained solely based on the remaining dataset, i.e., $\mathcal{M}_u = \mathcal{U}(D, D_e, \mathcal{A}(D))$ is hoped to equal $\mathcal{M}_u = \mathcal{A}(D_r)$. For an IB-trained model, since the key of the model is to learn the representation Z to satisfy the IB objective, the unlearning problem of IB-trained models can be moreover described as

$$\begin{cases} p(Z|X_{-X_e}) \text{ unlearned by } \mathcal{M}_u = \mathcal{U}(D, D_e, \mathcal{A}(D)), \\ p(Z|X_r) \text{ retrained by } \mathcal{M}_u = \mathcal{A}(D_r), \\ p(Z|X_r) = p(Z|X_{-X_e}), \end{cases} \tag{5}$$

where $p(Z|X_{-X_e})$ denotes the representation is unlearned through removing the information of the erased dataset $D_e = (X_e, Y_e)$ from the trained representation, and $p(Z|X_r)$ denotes the representation is achieved by retraining based on X_r.

Current solutions [7], [11] tried to solve the unlearning problem by minimizing the difference between the model unlearned based on the erased dataset $\mathcal{M}_u = \mathcal{U}(D, D_e, \mathcal{A}(D))$ and the model retrained without the erased dataset $\mathcal{M}_u = \mathcal{A}(D_r)$. Usually, they use the L_2-norm or Kullback-Leibler divergence (KLD) to evaluate the distance between two model parameters' distributions and minimize the distance as unlearning optimization. However, if we do not retrain the model, we cannot achieve the exact $\mathcal{M}_u = \mathcal{A}(D_r)$. Therefore, the certified-removal methods [8], [9] tried to estimate the influence of the erased dataset (usually using the Hessian matrix) and subtract this influence estimation from the original model. The variational Bayesian unlearning methods [7], [11] had an unlearning term comprised of minimizing the log posterior probability of the erased dataset (which is maximized in the learning) and the KLD between the assumed distribution and the original posterior. Directly optimizing unlearning like the above methods can easily cause catastrophic unlearning.

In order to avoid dramatic accuracy degradation, existing approximate unlearning methods usually propose a threshold or a bounded range to limit the unlearning extent. For example, Nguyen et al. [7] set a posterior threshold to ensure their unlearning method focused on values of the model parameters with sufficiently larger posterior. However, it is hard to obtain a suitable fixed threshold value to balance both the erasure effect and model accuracy preservation optimally. To effectively unlearn the erased sample meanwhile keeping the performance on the remaining dataset for IB-trained models, we redefine machine unlearning as a two-objective optimization problem as

$$\textbf{Forgetting:} \quad \min_{(\theta^r, \theta^a)} D_{KL}[p(Z|X_{-X_e})||p(Z|X_r)], \tag{6}$$

$$\textbf{Remembering:} \quad \min_{(\theta^r, \theta^a)} \mathcal{L}(D_r; (\theta^r, \theta^a)), \tag{7}$$

The Eq. (6) is the forgetting purpose that aims to minimize the distance between the model before and after unlearning. The KL distance D_{KL} can also be changed to other distance metrics, such as L_2-norm. In Eq. (7), the remembering objective, $\mathcal{L}(.)$ is the model's empirical loss on the remaining dataset, and the purpose is to keep the minimum loss. Retraining from scratch is still a solution to this problem definition. First, if the unlearning algorithm $\mathcal{U}$ is retraining, the retraining process is to make the model approach to the retrained model, which is minimizing the first objective, Eq. (6). Second, the retraining process minimizes the model loss on the remaining dataset, which optimizes the second objective. However, as we explained above, the storage and computation costs of retraining are expensive. Therefore, we solve this problem by further minimizing the mutual information between the learned representation and the erased dataset, named representation forgetting unlearning with parameter self-sharing.

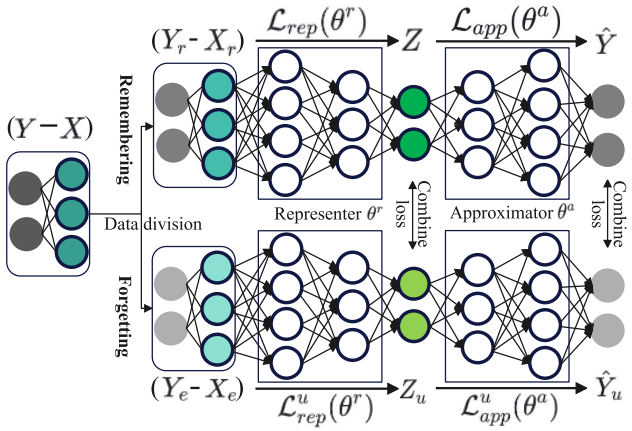


Fig. 3. In the RFU-SS training process, first, we divide the full training dataset into the erased dataset (X_e, Y_e) and the remaining dataset (X_r, Y_r). We sample the same minibatch size from the erased dataset for RFU to implement the forgetting objective and from the remaining dataset for normal learning training loss calculation to implement the remembering objective. Then, we use parameter self-sharing to combine the loss of both forgetting and remembering objectives to optimize them together based on the same model.

B. Overview of RFU-SS

We here briefly introduce the primary concept and process of our approach, followed by a detailed discussion of the associated technologies. The main procedure of RFU-SS is depicted in Figure 3, which contains two main components (**forgetting** and **remembering**) that achieve the two objectives of unlearning. We propose representation forgetting unlearning (RFU) based on the erased samples to tackle the forgetting objective. This operation removes the contribution of erased samples from the IB-trained models, which implements unlearning. Traditional machine unlearning methods [9], [13], [24], [25] only contain this data removal part, and they only use a fixed threshold to limit catastrophic unlearning. To mitigate the utility degradation caused by unlearning, we add the remembering objective and train the model as normal based on the several remaining samples. We propose a method called parameter self-sharing, which combines the loss of two components and optimizes them simultaneously to achieve the best balance of the two objectives. The entire solution is called RFU-SS.

C. Representation Forgetting Unlearning

As we introduced in the preliminary, an IB-trained model learns a representation Z, which minimizes $I(X; Z)$ and maximizes $I(Y; Z)$ based on full training dataset $D = (X, Y)$ by the representer θ^r and approximator θ^a. We assume the model $\mathcal{M}(\theta^r, \theta^a)$ is well trained, and we now execute unlearning based on this already-trained model. Our basic idea is further minimizing the mutual information $I(X_e; Z)$ of the erased inputs X_e and representation Z to eliminate the information of X_e from the trained representation Z and representer θ^r.

We also minimize the mutual information $I(Y_e; Z)$ of the erased labels Y_e and representation Z to remove the information of Y_e from the representation Z and approximator θ^a.

In the previous IB training process, we minimized $I(X; Z)$ but with a constraint maximizing $I(Y; Z)$ to guarantee the model utility. It means that previous training does not make $I(X_e; Z) = 0$, and it only minimizes $I(X_e; Z)$ with a constraint of maximum $I(Y_e; Z)$. Since $I(X_e; Y_e) \geq 0$ and $I(Y_e; Z)$ is maximized, the $I(X_e; Z)$ is also larger than 0 in previous training. Therefore, we need to minimize both $I(X_e; Z)$ and $I(Y_e; Z)$ to implement unlearning. The optimal unlearned representation contains no information of (X_e, Y_e), i.e., $I(X_e; Z) = 0$ and $I(Y_e; Z) = 0$. Directly minimizing $I(X_e; Z)$ and $I(Y_e; Z)$ can easily achieve this purpose but also make the representation forget everything, including the knowledge that Z learned on the remaining dataset before. Therefore, we set a limitation term when minimizing $I(X_e; Z)$ and $I(Y_e; Z)$ in RFU to ensure the unlearned representation is not significantly different from the original one. A simple solution to implement the limitation term is to minimize the Kullback-Leibler divergence D_{KL} [32] between the unlearned representation and originally trained representation before unlearning. Specifically, the unlearning loss for the representer can be described as

$$\mathcal{L}^u_{rep}(\beta_u) = \beta_u \cdot \underbrace{I_{\theta^r}(X_e; Z)}_{\text{Erasure term}} + \underbrace{D_{KL}[p_{\theta^r}(Z|X_{-X_e})||p_{\theta^r_{fix}}(Z|X_{-X_e})]}_{\text{Limitation term}}, \quad (8)$$

and the unlearning loss for approximator can be described as

$$\mathcal{L}^u_{app}(\beta_u) = \beta_u \cdot \underbrace{I_{\theta^a}(Y_e; Z)}_{\text{Erasure term}} + \underbrace{D_{KL}[p_{\theta^a}(Y_{-Y_e}|Z)||p_{\theta^a_{fix}}(Y_{-Y_e}|Z)]}_{\text{Limitation term}}, \quad (9)$$

where β_u is an unlearning rate parameter that we introduced to control the data erasure extent during RFU training. θ^r_{fix} and θ^a_{fix} are the representer and approximator that we fixed using the trained model before unlearning to calculate the limitation term to ensure the unlearned representation not forgetting everything.

By combining these two loss functions, we achieve the final RFU optimization loss to achieve the forgetting objective as

$$\begin{aligned}\mathcal{L}^u &= \mathcal{L}^u_{rep}(\beta_u) + \mathcal{L}^u_{app}(\beta_u) \\ &= \beta_u \cdot I_{\theta^r}(X_e; Z) + D_{KL}[p_{\theta^r}(Z|X_{-X_e})||p_{\theta^r_{fix}}(Z|X_{-X_e})] \\ &\quad + \beta_u \cdot I_{\theta^a}(Y_e; Z) + D_{KL}[p_{\theta^a}(Y_{-Y_e}|Z)||p_{\theta^a_{fix}}(Y_{-Y_e}|Z)].\end{aligned} \quad (10)$$

Minimizing the loss Eq. (10) to unlearn the information of the erased data $D_e = (X_e, Y_e)$ from the trained IB model can be proved equivalent to minimizing the retraining IB model's loss, $\mathcal{L}^r = I(X_r; Z) - I(Y_r; Z)$. Usually, the loss functions in Eqs. (8) to (10) can be optimized in a variational form like [16], [30], and [31]. Alternatively, we can also use MINE [43] to calculate the mutual information estimation for the optimization of these loss functions. We will show an example of the variational optimization form of these equations similar to Eq. (2) in our Algorithm 1 later. Moreover, optimizing Eqs. (8) and (9) by grid search for a suitable unlearning rate β_u, RFU can already perform better than existing approximate unlearning methods.

D. RFU With Parameter Self-Sharing

We have introduced the solution of optimizing the forgetting objective. Directly optimizing RFU yields a slightly superior unlearning outcome compared to existing approaches. Yet, it is hard to achieve the optimal balance between forgetting and remembering, i.e., erased data unlearning and model utility preservation. Therefore, we add a remembering objective and propose a multi-objective optimization method to solve the problem. To optimize the remembering objective of the two-objective unlearning, we continue training the unlearned model with the loss Eq. (1) on the remaining dataset (X_r, Y_r) to maintain the model accuracy. For clearness, we denote the corresponding loss in Eq. (1) on the remaining dataset (X_r, Y_r) as $\mathcal{L}^r_{rep}$ and $\mathcal{L}^r_{app}$. In particular, we can optimize them as multi-task learning in Eq. (3). The transformed minimization for the two-objective unlearning can be described as the following formulation:

$$\min_{(\theta^r, \theta^a)} \alpha^u \cdot (\mathcal{L}^u_{rep}(\beta_u) + \mathcal{L}^u_{app}(\beta_u)) + \alpha^r \cdot (\mathcal{L}^r_{rep} + \mathcal{L}^r_{app}), \quad (11)$$

where α^u and α^r are static or dynamically computed weights for forgetting and remembering tasks.

It is difficult and time-consuming to achieve the optimal settings of Eq. (11) if we use heuristic algorithms or grid search over various scalings [17], [18]. Therefore, we formulate the proposed two-objective machine unlearning problem as a multi-objective optimization problem, which aims to achieve Pareto optimality of the forgetting and remembering tasks.

Definition 1 (Pareto optimality for unlearning):

a) A solution (θ^r, θ^a) dominates a solution $(\bar{\theta}^r, \bar{\theta}^a)$ if $\mathcal{L}^u(\theta^r, \theta^a) \leq \mathcal{L}^u(\bar{\theta}^r, \bar{\theta}^a)$ and $\mathcal{L}^r(\theta^r, \theta^a) \leq \mathcal{L}^r(\bar{\theta}^r, \bar{\theta}^a)$ for both forgetting objective and remembering objective.

b) A solution $(\theta^{*,r}, \theta^{*,a})$ is called Pareto optimal if there exists no solution (θ^r, θ^a) that dominates $(\theta^{*,r}, \theta^{*,a})$.

The set of Pareto optimal solutions is named the Pareto set $(\mathcal{P}_{(\theta^r, \theta^a)})$. Usually, we can achieve the local optimality of multi-objective optimization problems via gradient descent [39], [41], [42], [44], [45], as we introduced in the preliminary section. They attempt to identify the Pareto stationarity, which is a necessary condition for Pareto optimality. Similarly, we also give a definition of Pareto stationarity for the optimization of unlearning.

Definition 2 (Pareto stationarity for unlearning): For the unlearning objective and remembering objective, (θ^r, θ^a) is Pareto stationary if there exists a scalar α^u, such that

$$\begin{cases} \alpha^u \nabla_{(\theta^r, \theta^a)}(\mathcal{L}^u(\theta^r, \theta^a)) + (1 - \alpha^u)\nabla_{(\theta^r, \theta^a)}(\mathcal{L}^r(\theta^r, \theta^a)) = 0, \\ \alpha^u \in [0, 1]. \end{cases} \quad (12)$$

Any solution that meets these conditions is called a Pareto stationary point for unlearning, and every Pareto optimal point is considered Pareto stationary. In order to find these points, we transform the two-objective unlearning problem into the following optimization problem,

$$\min_{\alpha^u} ||\alpha^u \nabla_{(\theta^r,\theta^a)}(\mathcal{L}^u) + (1-\alpha^u)\nabla_{(\theta^r,\theta^a)}(\mathcal{L}^r)||_2^2,$$
$$\text{s.t.} \quad \alpha^u \in [0,1]. \tag{13}$$

According to [18] and [39], we can derive that the solution to this equation will be either a Pareto stationarity for unlearning when it is 0, or it will give a direction to move in to improve both forgetting and remembering goals when the result is not 0. The problem of Eq. (13) to find an optimal α^u now is a convex quadratic problem with linear constraints as the gradients of the two objectives can be calculated as fixed values before each round update. We can achieve the results of this one-dimensional quadratic function about α^u via an analytical solution:

$$\hat{\alpha^u} = \left[\frac{(\nabla_{(\theta^r,\theta^a)}(\mathcal{L}^r) - \nabla_{(\theta^r,\theta^a)}(\mathcal{L}^u))^\intercal (\nabla_{(\theta^r,\theta^a)}(\mathcal{L}^r))}{||\nabla_{(\theta^r,\theta^a)}(\mathcal{L}^u) - \nabla_{(\theta^r,\theta^a)}(\mathcal{L}^r)||_2^2} \right]_{+,\frac{1}{\intercal}}, \tag{14}$$

where $[\cdot]_{+,\frac{1}{\intercal}}$ denotes clipping to $[0,1]$ as $[a]_{+,\frac{1}{\intercal}} = \max(\min(a,1),0)$. The value of $\hat{\alpha^u}$ is highly related to the numerator, the dot product between $\nabla_{(\theta^r,\theta^a)}(\mathcal{L}^r) - \nabla_{(\theta^r,\theta^a)}(\mathcal{L}^u)$ and $\nabla_{(\theta^r,\theta^a)}(\mathcal{L}^r)$. If the dot product is a large positive value, it implies that the difference between the two gradient vectors aligned with the gradients of $\mathcal{L}^r$, the loss of remembering. It means that the changes in $\mathcal{L}^r$ and $\mathcal{L}^u$ are generally in the same direction, and thus $\mathcal{L}^u$ might serve as a good approximation for $\mathcal{L}^r$ in that region of the parameter space. Hence, assigning a large $\hat{\alpha^u}$ weight to update the model with the forgetting loss gradients in this situation will not likely cause catastrophic unlearning. When the dot product is approaching zero, it means the gradient difference vector is almost orthogonal to $\nabla_{(\theta^r,\theta^a)}(\mathcal{L}^r)$, meaning that changes in $\mathcal{L}^r$ and $\mathcal{L}^u$ are unrelated in that particular point in the parameter space. Therefore, the weight $\hat{\alpha^u}$ is small in this situation and can effectively prevent catastrophic unlearning.

By dynamically and continuously adjusting the weight $\hat{\alpha^u}$ during the unlearning update process for both forgetting and remembering using gradient descent, this algorithm has been proven to converge to a point on the Pareto set [39]. Since our method shares the entire model parameters (θ^r, θ^a) during the updating and does not have additional task-specific layers as MTL, we call our method parameter self-sharing. The pseudocode about RFU-SS is presented in Algorithm 1.

In Algorithm 1, RFU-SS needs to optimize forgetting objective using RFU based on $D_e = (X_e, Y_e)$, and needs several samples of $D_r = (X_r, Y_r)$ to participate remembering objective calculation process. It first set a fixed temp model $\mathcal{M}_{fix}(\theta^r_{fix}, \theta^a_{fix})$ using $\mathcal{M}(\theta^r, \theta^a)$ for the later calculating the posterior of full training data used in the unlearning function. The unlearning training process is in Lines 2 to 10. During E epochs training, we first sample m minibatch data points $\{(x_i, y_i)\}_{i=1}^m$ from the erased dataset D_e for forgetting and m minibatch data points $\{(x_j, y_j)\}_{j=1}^m$ from the remaining dataset D_r for remembering objective calculation. Then, we generate the corresponding representation z_i and z_j using the representer θ^r based on the erased x_i and remaining x_j. After these preparations, we calculate the loss functions and the optimal α^u based on Eqs. (1), (8) to (10) and (14), and the corresponding calculations are shown from lines 6 to 9. All these equations are extended and expressed in a variational empirical form like Eq. (2). Finally, we update the model by combining the two optimization objectives together according to Eq. (14), as in line 10.

Algorithm 1 RFU With Parameter Self-Sharing

Input: $\mathcal{M}(\theta^r, \theta^a)$, $D_e = (X_e, Y_e)$, $D_r = (X_r, Y_r)$, and E

Output: Unlearned model $\mathcal{M}_u(\theta^r_u, \theta^a_u)$

1 Set the fixed temp model:
$\mathcal{M}_{fix}(\theta^r_{fix}, \theta^a_{fix}) \leftarrow \mathcal{M}(\theta^r, \theta^a)$
2 **for** E *epochs* **do**
3 Sample minibatch of m data points $\{(x_i, y_i)\}_{i=1}^m$ from the erased dataset D_e;
4 Sample minibatch of m data points $\{(x_j, y_j)\}_{j=1}^m$ from the remaining dataset D_r;
5 Generate $z_i \backsim p_{\theta^r}(\cdot|x_i)$, $z_j \backsim p_{\theta^r}(\cdot|x_j)$ based on input and the representer;
6 Calculate the unlearned representation loss function Eq. (8) in a per-sample form for each minibatch of size m as $\mathcal{L}^u_{rep}(\beta_u) =$
$\beta_u \cdot \frac{1}{m}\sum_{i=1}^m D_{KL}[p_{\theta^r}(z|x_i)||\prod_i^{|z|} q_i(z_i)]$
$+\frac{1}{m}\sum_{i=1}^m D_{KL}[p_{\theta^r_{fix}}(z|x_i)||p_{\theta^r}(z|x_i)]$
7 Calculate the unlearned approximation loss function Eq. (9) in a per-sample form for each minibatch of size m as
$\mathcal{L}^u_{app}(\beta_u) = \beta_u \cdot \frac{1}{m}\sum_{i=1}^m \log\ p_{\theta^a}(y_i|z_i)$
$+\frac{1}{m}\sum_{i=1}^m D_{KL}[p_{\theta^a_{fix}}(y_i|z_i)||p_{\theta^a}(y_i|z_i)]$
8 Calculate the representation and approximation loss based on the remaining data as Eq. (2)
$\mathcal{L}^r_{app} + \mathcal{L}^r_{rep} = -\frac{1}{m}\sum_{j=1}^m \log\ p_{\theta^a}(y_j|z_j)$
$+\frac{\beta}{m}\sum_{j=1}^m D_{KL}[p_{\theta^r}(z_j|x_j)||q_{\theta^r}(z_j)]$
9 Calculate $\hat{\alpha^u}$ using Eq. (14);
10 We update the model (both the representer and approximator) according to the gradients of the combined loss functions Eq. (13) as
$(\theta^r, \theta^a) \leftarrow (\theta^r, \theta^a) - \eta\nabla_{(\theta^r,\theta^a)}(\hat{\alpha^u} \cdot (\mathcal{L}^u_{rep}(\beta_u) + \mathcal{L}^u_{app}(\beta_u)) + (1-\hat{\alpha^u}) \cdot (\mathcal{L}^r_{rep} + \mathcal{L}^r_{app}))$
11 Return $\mathcal{M}_u(\theta^r_u, \theta^a_u) \leftarrow \mathcal{M}_u(\theta^r, \theta^a)$;

V. Performance Evaluation

In this section, we introduce the detailed evaluation of RFU-SS and comparisons between RFU-SS and the state-of-art methods [7], [8], [9], [11]. To effectively evaluate unlearning methods, most experiments are conducted to unlearn the backdoored samples as [19]. We also conduct experiments on unlearning normal data. Experimental results demonstrate that RFU-SS achieves a significant improvement

TABLE II
GENERAL EVALUATION RESULTS ON MNIST AND CIFAR10

Evaluation Metrics	MNIST, $EDR = 6\%$					CIFAR10, $EDR = 6\%$				
	Origin	HBU	VBU	RFU-SS	Retrain	Origin	HBU	VBU	RFU-SS	Retrain
Running Time (s)	44	2.42	**0.12**	0.33	41.36	552	28.01	**1.10**	4.6	518.88
KLD to retrain	221.32	214.21	216.21	**213.07**	-	24.84	20.22	21.87	**18.56**	-
Acc. on test dataset	97.6%	87.99%	89.37%	**97.44%**	97.63%	81.16%	75.32%	74.08%	**83.36%**	86.65%
Backdoor Acc.	100%	2.04%	2.83%	**0.2%**	0.08%	99.83%	0.1%	3.53%	**0.07%**	0.03%

in both unlearning effect and accuracy preservation than existing methods, regardless of whether unlearning backdoored or normal samples.

A. Experiment Setup

Our experiments are conducted on two representative datasets, MNIST and CIFAR10. We evaluate RFU-SS and compare it with state-of-the-art unlearning methods. There are two representative approximate unlearning methods, Hessian-matrix-based unlearning (HBU) [8], [9] and variational Bayesian unlearning (VBU) [7], [11]. To effectively evaluate unlearning methods, we refer to a common method [19] to conduct most experiments, adding the backdoor triggers to the erased data samples for the original ML model training. Then, we execute the unlearning methods to forget the backdoor. After unlearning, we verify whether a triggered input can still activate the previously injected backdoor of the unlearned model to evaluate the unlearning effect. We evaluate both the effectiveness and efficiency of different unlearning methods. Specifically, we use models' accuracy on the test dataset and backdoor inference success accuracy [19] on the erased dataset to evaluate the unlearning effectiveness. We use the models' running time to evaluate the unlearning efficiency.

We train an IB model with two linear models (one representer and one approximator) on MNIST, each with three hide layers, with a learning rate $\eta = 0.001$. And we train an IB model with one Resnet18 as the representer and one linear model as approximator on CIFAR10 with $\eta = 0.0005$. To ensure consistency and ease of implementation, during unlearning, we set the minibatch size to 100, which keeps the same when sampling from the remaining and erased datasets. All models are implemented using Pytorch, and experiments are done on a cluster with four NVIDIA 1080ti GPUs.

B. Overall Evaluation

We first demonstrate the overall evaluation results of different methods on MNIST and CIFAR10, shown in Table II. On both MNIST and CIFAR10, we set the erased data ratio (*EDR*) of the full original dataset $EDR = 6\%$, and the proposed unlearning rate $\beta_u = 0.1$. We evaluate efficiency by calculating the model's running time, which involves recording the time used in each training batch and multiplying it by the number of training epochs. We evaluate the effectiveness from three aspects, including the KLD between the unlearned and the retrained models, the model accuracy on the test dataset, and the backdoor accuracy on the erased dataset.

1) Evaluation of Efficiency: From the running time shown in Table II, all unlearning methods achieve a speedup of more than 10× compared with retraining. HBU performs worse than the other two unlearning methods because it needs to calculate the Hessian matrix based on the remaining dataset. VBU achieves the best efficiency performance. RFU-SS consumes more running time than VBU but much less than HBU on both MNIST and CIFAR10.

2) Evaluation of Effectiveness: In Table II, all unlearning methods have moved the distribution of unlearned models closer to the distribution of the retrained model than the original one. It is reflected in the closer KLD to the retrained model, where RFU-SS makes the unlearned model closest to the retrained model on both MNIST and CIFAR10. In the evaluation of accuracy, catastrophic unlearning appears in HBU and VBU on MNIST, which has a huge accuracy degradation. Only RFU-SS almost eliminates the accuracy degradation, which performs similarly to retraining.

On CIFAR10, HBU and VBU also have around 6% accuracy degradation from the original model accuracy. RFU-SS achieves the best accuracy at around 83.36%, which increases 2% than the original model accuracy 81.16%, but still less than the accuracy of the retrained model, 86.65%. It is because the original model is trained with several mixed backdoored samples, which harms the model accuracy to 81.16%. In contrast, the retrained model is trained based on the remaining clean dataset, which removes these backdoored samples. Since CIFAR10 is a more complex dataset than MNIST, the backdoors have a greater negative influence on the trained model, reducing its accuracy from around 86% to 81% on CIFAR10. However, the backdoors do not have as much of an effect on the model's performance on MNIST. The model accuracy recovery is also clearly demonstrated on CIFAR10. We will show the detailed results of recovery later. Moreover, all unlearning methods can reduce the backdoor accuracy on the erased dataset to less than 10%, lower than randomly selecting. RFU-SS also achieves the best performance in removing the backdoored samples.

C. Evaluation of the Influence of Different Variables

1) Impact of the Erased Data Ratio: In this paper, there are two main variables, *EDR* and β_u, that may have a considerable influence on the model performances. We first evaluate the impact of *EDR*. And when we conduct experiments to evaluate one variable, we will set a fixed other variable. The results on MNIST and CIFAR10 about various *EDR* are shown in Figure 4.

Our effectiveness evaluations include accuracy on the test dataset (Figures 4(a) and 4(b)) and backdoor accuracy on the erased dataset (Figures 4(c) and 4(d)). The accuracy of unlearned models on the test dataset and backdoored samples reflects the utility preservation and the forgetting effectiveness

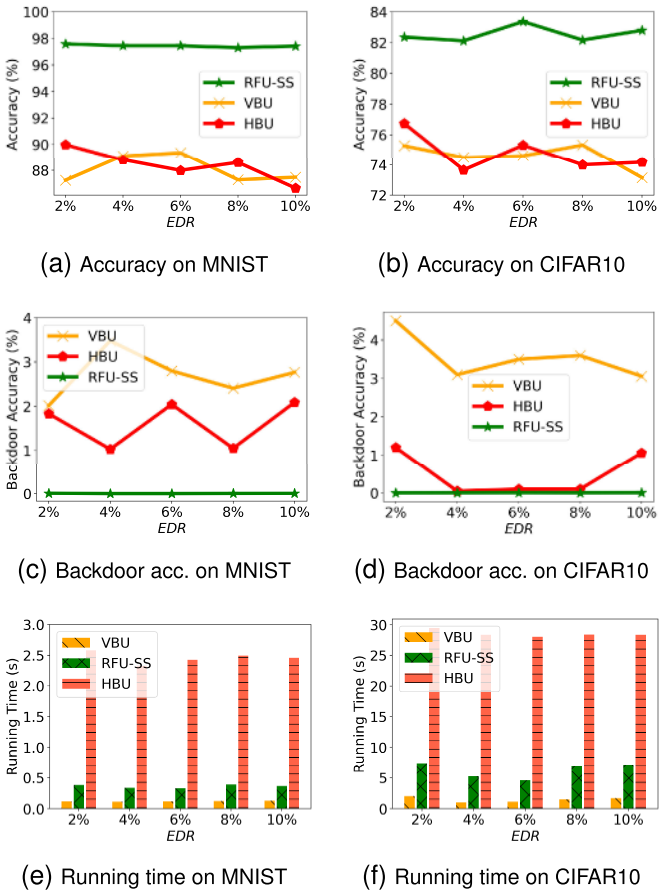


Fig. 4. The running time, acc. on the test dataset, and backdoor acc. on the erased dataset of various *EDR*.

of unlearning methods, respectively. Since VBU and HBU are easy to cause accuracy degradation as the training proceeds, we set a stop threshold that when the backdoor accuracy decreases lower than 5%, we stop the VBU and HBU training. Since RFU-SS can automatically adjust the tradeoff weight α^u, we can continue the model training without worrying about the accuracy degradation and stop the training at any time when the backdoor accuracy decreases satisfied.

On both MNIST and CIFAR10, shown in Figures 4(a) and 4(b), as *EDR* increases, the accuracy of all unlearned models decreases slightly. In this comparison, we focus on three unlearning methods for clarity. The accuracy of the original model is roughly the same as RFU-SS for MNIST and slightly lower than RFU-SS for CIFAR10. Notably, both HBU and VBU experience substantial utility degradation, exceeding 5% accuracy drop compared with the model accuracy before unlearning when attempting to remove the impact of backdoored samples. By contrast, RFU-SS almost has no accuracy degradation on both MNIST and CIFAR10. On the backdoor removal aspect, shown in Figures 4(c) and 4(d), all unlearning methods successfully erase the influence of the backdoor (decreasing backdoor accuracy lower than randomly selecting 10%), where RFU-SS achieves the best forgetting effect, reducing the backdoor accuracy to approaching 0%.

We evaluate the efficiency from running time shown in Figures 4(e) and 4(f). The *EDR* influences the running time slightly. HBU consumes the most running time in unlearning, more than 5× compared with the other two unlearning methods because HBU needs to calculate Hessian estimation based on the remaining dataset. VBU performs best in reducing running time. RFU-SS consumes around double the running time as VBU because it optimizes two tasks together, but it is still much faster than HFU.

Figure 5 shows the changes in accuracy on the remaining dataset and the backdoor (abbreviated as bac.) accuracy on the erased dataset of all unlearning methods during the training process on MNIST and CIFAR10, respectively. Here, we will not stop the model training even if the backdoor accuracy decreases below 5%. As the *EDR* increases, it slightly slows down both accuracy and backdoor accuracy degradation speed. It means the model has been backdoored deeper as the backdoored samples increase, and it will take more time for all unlearning methods to remove these backdoors' influence.

In Figure 5(a), on MNIST, it is obvious that HBU consumes more epochs of training to reduce the backdoor accuracy as *EDR* increases from 2% to 8%. In Figure 5(b), on CIFAR10, the backdoor accuracy in the initial training epoch increases as the *EDR* increases. Moreover, on CIFAR10, the accuracy degradation of both HBU and VBU slows down as the *EDR* increases. The main reason is that a bigger *EDR* means the model is backdoored deeper in the former learning process. Therefore, it increases the backdoor accuracy at the initial training round on CIFAR10 and takes more time to forget these backdoored samples. Although all unlearning methods can remove the influence of backdoored samples, only RFU-SS can keep model accuracy on the remaining dataset during the training process. The accuracy of BFU and HBU on the remaining dataset decreases as the unlearning training continues.

2) Impact of the Unlearning Rate: Another essential variable is the unlearning rate, and we also apply our proposed unlearning rate β_u to VBU and gladly find that it can help VBU improve performance. Since HBU does not have this unlearning rate β_u parameter and it is hard to add β_u in HBU easily, here we only add the β_u to VBU to control the unlearning extent and compare our method with VBU. We demonstrate the changes of accuracy on the remaining dataset and the backdoor accuracy on the erased dataset during unlearning training process on MNIST and CIFAR10 in Figure 6.

As we analyzed before, the unlearning rate will influence the unlearning speed during the training process, and the results also support our analysis. Specifically, when we choose a larger unlearning rate, it will speed up the training process, i.e., it decreases both the backdoor accuracy on the erased dataset and accuracy on the remaining dataset with fewer training epochs. On MNIST, in Figure 6(a), when β_u is small, 0.01 and 0.1, the backdoor accuracy of two methods on the erased dataset and the accuracy of VBU drop slower than when β_u is big, 1 and 10. Since RFU-SS optimizes two objectives together, it can maintain model accuracy on the remaining dataset during unlearning training, which effectively prevents accuracy degradation compared with VBU. Figure 6(b) also shows similar results. When β_u is small, $\beta_u = 0.001$ and $\beta_u = 0.01$, the backdoor accuracy of RFU-SS obviously drops slower than when β_u is big, $\beta_u = 0.1$ and $\beta_u = 1$.

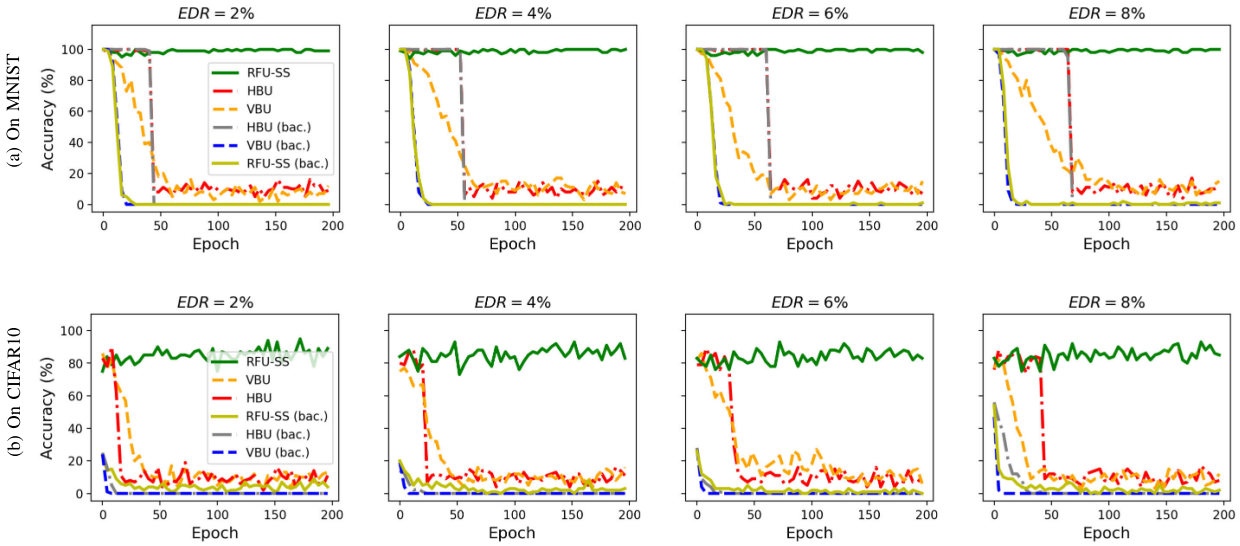


Fig. 5. The changes of accuracy and backdoor accuracy on various *EDR* during training on MNIST and CIFAR10.

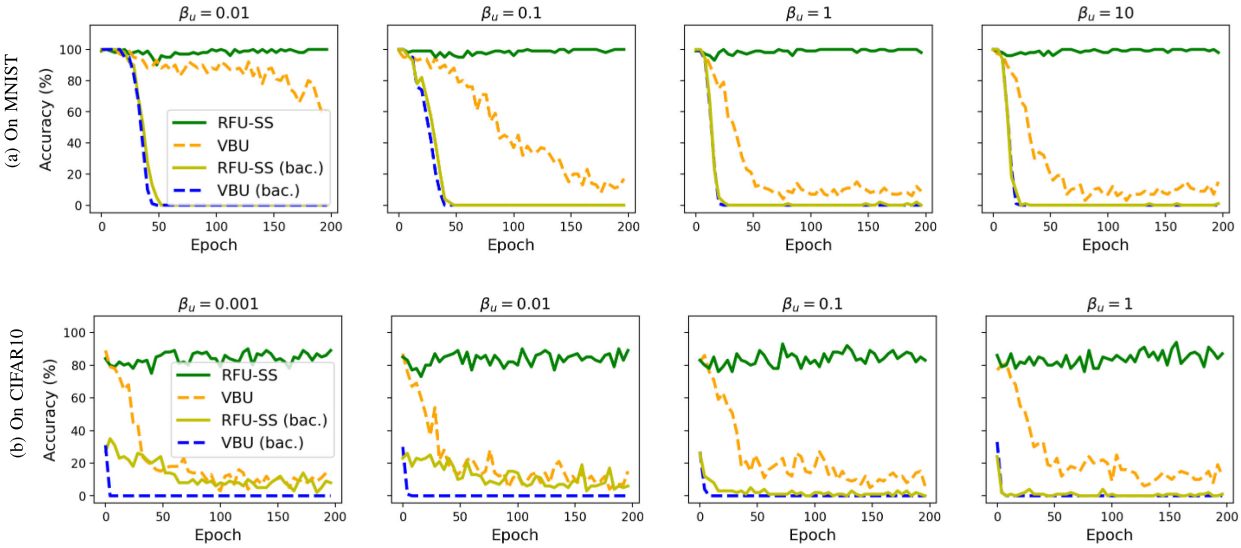


Fig. 6. The changes of accuracy and backdoor accuracy on various β_u during training on MNIST and CIFAR10.

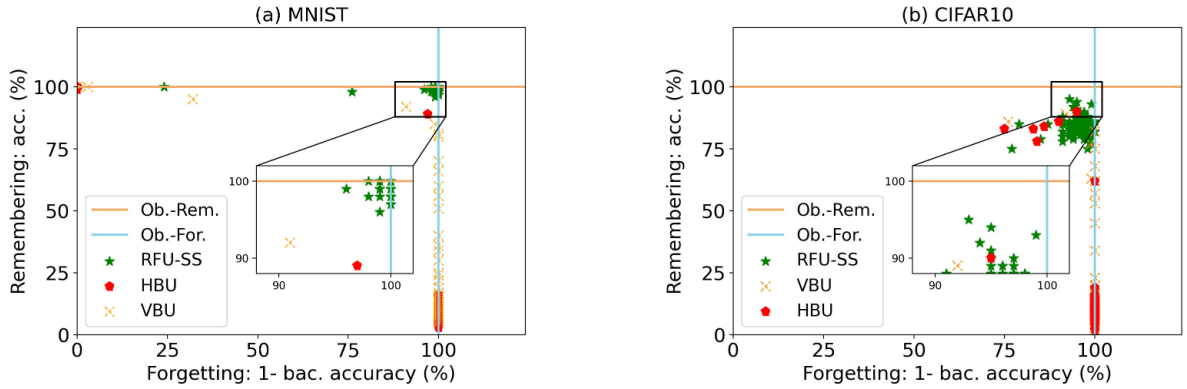


Fig. 7. Results of two-objective optimization unlearning on MNIST and CIFAR10. The two straight lines "Ob.-Rem." and "Ob.-For." are abbreviations for "objective-remembering" and "objective-forgetting", meaning the optimization targets for remembering and forgetting. And the remembering effectiveness is presented using accuracy on the remaining dataset, and the forgetting effectiveness is presented using 1 − backdoor accuracy on the erased dataset.

D. Optimization Results of Unlearning

Figure 7 shows the optimization results of all unlearning methods for forgetting and remembering objectives of unlearning on MNIST and CIFAR10. In this part, we show the remembering effectiveness using the accuracy on the remaining dataset and the forgetting effectiveness using 1 − backdoor accuracy on the erased dataset. On MNIST, in Figure 7, RFU-SS can achieve the best remembering and

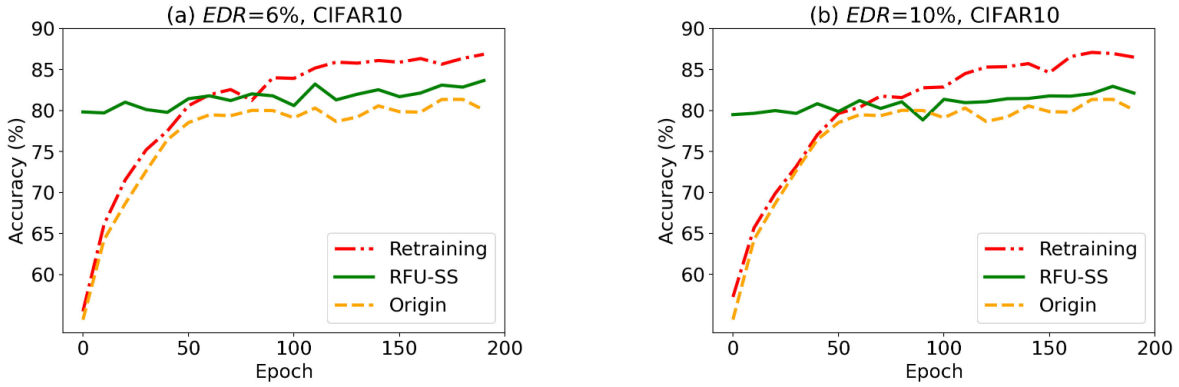


Fig. 8. RFU-SS recovers the model knowledge harmed by backdoors on CIFAR10.

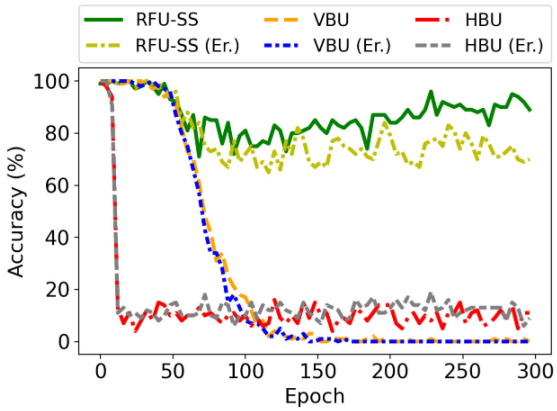


Fig. 9. Training of unlearning independent and identical distribution (IID) normal data. Changes of accuracy on the remaining dataset and on the erased (abbreviated as Er.) dataset during training on MNIST.

forgetting effectiveness, 100% of both. Only a few results of VBU and HBU can achieve a good balance of forgetting and remembering, but these results are not as optimal as RFU-SS. VBU and HBU are hard to achieve good optimization of the two objectives as most of their optimization results exist on the most remembering with least forgetting or the most forgetting with least remembering. Figure 7(b) also shows similar optimization results on CIFAR10, where only RFU-SS can achieve the best balance between forgetting and remembering. Here, we should note that although a few result points of HBU and VBU can achieve acceptable performance, it is hard to precisely stop the model training at that result round in practice. In contrast, because RFU-SS dynamically adjusts the updating weight between forgetting and remembering, it can converge after several epochs of training, making it easier to obtain optimal results.

E. Recover the Knowledge Harmed by Backdoor

We know that adding backdoors will harm model accuracy on the test dataset that has an independent and identical distribution (IID) as the clean full training dataset, e.g., a normally trained model on CIFAR10 has around 86% accuracy; after adding backdoors, the model accuracy drops to around 81%. In Figure 8, we show the accuracy changes during the training process, where "Origin" means the original model trained based on the dataset mixed backdoored samples, "Retraining" means the retrained model based on the remaining clean dataset without these backdoored samples. We can clearly see a gap between the accuracy of the model trained with backdoored samples and without these samples. Existing unlearning methods, such as HBU and VBU, find it difficult to mitigate the accuracy degradation to the same level as the original model, let alone recover the accuracy better than the original model. However, Figure 8 shows that RFU-SS can recover the model accuracy lost due to backdoored or noised samples in ML model training. When $EDR = 6\%$, RFU-SS increases the model accuracy from around 81.16% (Origin) to 83.36%. When $EDR = 10\%$, RFU-SS increases the model accuracy from around 80.55% (Origin) to 82.6%. Although it still has a little gap from the retrained model, RFU-SS is the first unlearning method that can recover the accuracy harmed by backdoored or noised samples.

F. Additional Evaluations on Normal Data Erasure

We additionally evaluate HBU, VBU and RFU-SS in unlearning the IID normal data, as shown in Figure 9. In this experiment, we hope to figure out the unlearning effectiveness of all methods when the erased dataset is normal IID. The results show that unlearning these regular IID data will inevitably harm the accuracy of the original model. Training with fewer data, even if the unlearning algorithm is retraining from scratch, will decrease model accuracy. When unlearning methods are VBU and HBU, these two methods unlearn the erased data, making the accuracy on the erased dataset drop, accompanied by the degradation of the accuracy on the remaining dataset. There is no clear gap between the model accuracy on the remaining dataset and the model accuracy on the erased dataset. It means that HBU and VBU cannot unlearn the erased data while not harming the knowledge learned on the remaining dataset. In contrast, RFU-SS can increase the accuracy gap between the model's accuracy on the remaining and the erased datasets. It means that RFU-SS makes the model forget the knowledge of the erased data even if the dataset is IID as the full training dataset. After 300 epochs of training, RFU-SS achieves an accuracy gap of around 30%. The model accuracy on the remaining dataset is approximately 93%, and the accuracy on the erased dataset

is around 63%. Neither HBU nor VBU can achieve such an unlearning effect in removing the influence of IID data.

VI. Summary and Future Work

In this paper, we address the problem of "catastrophic unlearning", which refers to dramatic model utility degradation during machine unlearning. We start with formulating machine unlearning as a two-objective optimization problem, including data erasure and accuracy preservation. Then, we propose a novel method named RFU-SS to solve this two-objective optimization unlearning problem. RFU-SS comprises two main steps. The first step of RFU-SS is the RFU method, which is an unlearning method tailored from models trained using the IB method. RFU unlearns the contribution of the erased dataset from the representation of a trained IB model. The second step involves optimizing RFU using parameter self-sharing, referred to as RFU-SS. Its goal is to identify a Pareto optimal solution for the two-objective unlearning problem, striking the ideal balance between removing the erased data's impact and preserving the model utility. Extensive experimental results demonstrate that RFU-SS significantly outperforms state-of-the-art methods in preventing accuracy degradation during unlearning. Moreover, RFU-SS can recover the model accuracy harmed by backdoored or noised samples, which cannot achieve by existing unlearning methods.

There are so many exciting future directions for this research. At a problem-definition level, we have further clarified that machine unlearning contains two purposes: user-specified data removal and model utility preservation. Based on this novel understanding and the idea of our RFU-SS solution, we can improve other existing machine unlearning methods to make them also mitigate "catastrophic unlearning". We also envision using the RFU-SS mechanism to support real-life applications such as medical diagnosis, point-of-interest recommendations, and graph analysis. Moreover, RFU-SS can be applied to numerous current life-long learning models, helping to forget noised or anomaly samples to ensure model performance.

References

[1] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 463–480.

[2] H. Yan, X. Li, Z. Guo, H. Li, F. Li, and X. Lin, "ARCANE: An efficient architecture for exact machine unlearning," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, Vienna, Austria, Jul. 2022, pp. 4006–4013, doi: 10.24963/ijcai.2022/556.

[3] L. Bourtoule et al., "Machine unlearning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 141–159.

[4] C. Chen, F. Sun, M. Zhang, and B. Ding, "Recommendation unlearning," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 2768–2777.

[5] G. Wu, M. Hashemi, and C. Srinivasa, "PUMA: Performance unchanged model augmentation for training data removal," in *Proc. 36th AAAI Conf. Artif. Intell., 34th Conf. Innov. Appl. Artif. Intell. (IAAI), 12th Symp. Educ. Adv. Artif. Intell. (EAAI)*, Feb./Mar. 2022, pp. 8675–8682. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20846

[6] Y. Wu, E. Dobriban, and S. Davidson, "DeltaGrad: Rapid retraining of machine learning models," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10355–10366.

[7] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Variational Bayesian unlearning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 16025–16036.

[8] C. Guo, T. Goldstein, A. Y. Hannun, and L. van der Maaten, "Certified data removal from machine learning models," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, in Proceedings of Machine Learning Research, vol. 119, Jul. 2020, pp. 3832–3842.

[9] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh, "Remember what you want to forget: Algorithms for machine unlearning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 18075–18086.

[10] P. Xu, F. Roosta, and M. W. Mahoney, "Second-order optimization for non-convex machine learning: An empirical study," in *Proc. SIAM Int. Conf. Data Mining*, 2020, pp. 199–207.

[11] S. Fu, F. He, and D. Tao, "Knowledge removal in sampling-based Bayesian inference," 2022, *arXiv:2203.12964*.

[12] C. W. Fox and S. J. Roberts, "A tutorial on variational Bayesian inference," *Artif. Intell. Rev.*, vol. 38, no. 2, pp. 85–95, Aug. 2012.

[13] R. Mehta, S. Pal, V. Singh, and S. N. Ravi, "Deep unlearning via randomized conditionally independent hessians," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10412–10421.

[14] Q. P. Nguyen, R. Oikawa, D. M. Divakaran, M. C. Chan, and B. K. H. Low, "Markov chain Monte Carlo-based machine unlearning: Unlearning what needs to be forgotten," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, Nagasaki, Japan, May/Jun. 2022, pp. 351–363.

[15] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," 2000, *arXiv:physics/0004057*.

[16] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," 2016, *arXiv:1612.00410*.

[17] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7482–7491.

[18] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, Montreal, QC, Canada, Dec. 2018, pp. 525–536.

[19] H. Hu, Z. Salcic, G. Dobbie, J. Chen, L. Sun, and X. Zhang, "Membership inference via backdooring," in *Proc. 31st Int. Joint Conf. Artif. Intell. (IJCAI)*, Vienna, Austria, L. D. Raedt, Ed., Jul. 2022, pp. 3832–3838, doi: 10.24963/ijcai.2022/532.

[20] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou, "Making AI forget you: Data deletion in machine learning," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, Vancouver, BC, Canada, Dec. 2019, pp. 3513–3526.

[21] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9301–9309.

[22] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 13, pp. 11516–11524.

[23] J. Brophy and D. Lowd, "Machine unlearning for random forests," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1092–1104.

[24] Y. Li, C. Wang, and G. Cheng, "Online forgetting process for linear regression models," in *Proc. 24th Int. Conf. Artif. Intell. Statist. (AISTATS)* in Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130, Apr. 2021, pp. 217–225. [Online]. Available: http://proceedings.mlr.press/v130/li21a.html

[25] A. Warnecke, L. Pirch, C. Wressnegger, and K. Rieck, "Machine unlearning of features and labels," 2021, *arXiv:2108.11577*.

[26] A. Golatkar, A. Achille, A. Ravichandran, M. Polito, and S. Soatto, "Mixed-privacy forgetting in deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 792–801.

[27] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," in *Proc. Algorithmic Learn. Theory*, 2021, pp. 931–962.

[28] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Apr. 2015, pp. 1–5.

[29] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 4, pp. 460–473, Jul. 1972.

[30] A. Achille and S. Soatto, "Information dropout: Learning optimal representations through noisy computation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2897–2905, Dec. 2018.

[31] S. Bang, P. Xie, H. Lee, W. Wu, and E. Xing, "Explaining a black-box by using a deep variational information bottleneck approach," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 13, pp. 11396–11404.

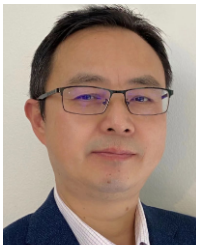
[32] J. M. Joyce, "Kullback–Leibler divergence," in *International Encyclopedia of Statistical Science*. Berlin, Germany: Springer, 2011, pp. 720–722.

[33] X. Lin, H.-L. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong, "Pareto multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, Vancouver, BC, Canada, Dec. 2019, pp. 12037–12047.
[34] P. Guo, C.-Y. Lee, and D. Ulbricht, "Learning to branch for multi-task learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3854–3863.
[35] J. Ma, Z. Zhao, J. Chen, A. Li, L. Hong, and E. H. Chi, "SNR: Sub-network routing for flexible parameter sharing in multi-task learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 216–223.
[36] X. Sun, R. Panda, R. Feris, and K. Saenko, "Adashare: Learning what to share for efficient deep multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, Nov. 2020, pp. 8728–8740.
[37] L. Duong, T. Cohn, S. Bird, and P. Cook, "Low resource dependency parsing: cross-lingual parameter sharing in a neural network parser," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics, 7th Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 845–850.
[38] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, "Latent multi-task architecture learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 4822–4829.
[39] J.-A. Désidéri, "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization," *Comp. Rendus Mathematique*, vol. 350, nos. 5–6, pp. 313–318, Mar. 2012.
[40] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Traces and Emergence of Nonlinear Programming*. Basel, Switzerland: Springer, 2013, pp. 247–258.
[41] J. Fliege and B. F. Svaiter, "Steepest descent methods for multicriteria optimization," *Math. Methods Oper. Res.*, vol. 51, no. 3, pp. 479–494, Aug. 2000.
[42] S. Schäffler, R. Schultz, and K. Weinzierl, "Stochastic method for the solution of unconstrained vector optimization problems," *J. Optim. Theory Appl.*, vol. 114, no. 1, pp. 209–222, Jul. 2002.
[43] M. I. Belghazi et al., "Mutual information neural estimation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 531–540.
[44] S. Peitz and M. Dellnitz, "Gradient-based multiobjective optimization with uncertainties," in *NEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop Held on September 20-24, 2016 in Tlalnepantla, Mexico*. Springer, 2018, pp. 159–182.
[45] F. Poirion, Q. Mercier, and J.-A. Désidéri, "Descent algorithm for non-smooth stochastic multiobjective optimization," *Comput. Optim. Appl.*, vol. 68, no. 2, pp. 317–331, Nov. 2017.

Weiqi Wang (Student Member, IEEE) received the M.Sc. degree in computer science from Soochow University, Suzhou, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Computer Science, University of Technology Sydney, under the supervision of Prof. Shui Yu. He previously worked as a Senior Algorithm Engineer with the Department of AI-Strategy, Local Consumer Services Segment, Alibaba Group. His research interests include machine unlearning, federated learning, and security and privacy in machine learning. He has been actively involved in the research community by serving as a reviewer for prestige journals, such as IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, and IEEE INTERNET OF THINGS JOURNAL, and international conferences, such as The ACM Web Conference (WWW), IEEE ICC, and IEEE GLOBECOM.

Chenhan Zhang (Student Member, IEEE) received the B.Eng. degrees from the University of Wollongong, Wollongong, Australia, and Zhengzhou University, Zhengzhou, China, in 2017 and 2018, respectively, and the M.S. degree from the City University of Hong Kong, in 2019. He is currently pursuing the Ph.D. degree with the School of Computer Science, University of Technology Sydney, Australia. His research interests include deep learning, security and privacy in graph neural networks, and intelligent transportation systems.

Zhiyi Tian (Student Member, IEEE) received the B.S. degree in information security and the M.S. degree in computer technology from Sichuan University, China, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree with the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. His research interests include security and privacy issues in deep learning and federated learning. He has been actively involved in the research community by serving as a reviewer for prestige journals, such as *ACM Computing Surveys* and IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and international conferences, such as IEEE ICC and IEEE GLOBECOM.



Shui Yu (Fellow, IEEE) received the Ph.D. degree from Deakin University, Australia, in 2004. He is currently a Professor with the School of Computer Science and the Deputy Chair of the University Research Committee and the University of Technology Sydney, Australia. He has published five monographs and edited two books, more than 500 technical papers at different venues, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, IEEE/ACM TRANSACTIONS ON NETWORKING, and INFOCOM. His current H-index is 73. He has been promoting the research field of networking for big data since 2013, and his research outputs have been widely adopted by industrial systems, such as Amazon Cloud Security. His research interests include cybersecurity, network science, big data, and mathematical modeling. He is an Elected Member of Board of Governors of IEEE VTS and ComSoc. He is a member of ACM and AAAS. He is serving the editorial boards for the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS (Area Editor) and the IEEE INTERNET OF THINGS JOURNAL (Editor). He served as a Distinguished Lecturer for the IEEE Communications Society, from 2018 to 2021. He is a Distinguished Visitor of the IEEE Computer Society.