UTS UNIVERSITY OF TECHNOLOGY SYDNEY

A Study on Machine Unlearning

by Weiqi Wang

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Professor Shui Yu

University of Technology Sydney
Faculty of Engineering and Information Technology

November 2024

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Weiqi Wang*, declare that this thesis is submitted in fulfilment of the requirements for the award of *Doctor of Philosophy*, in the *School of Computer Science*, *Faculty of Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

SIGNATURE: Production Note:
Signature removed prior to publication.

DATE: 25th November, 2024

PLACE: Sydney, Australia

Abstract

As the right to be forgotten has been legislated worldwide, numerous studies have sought to design unlearning methods to protect users' privacy when they wish to remove their data from machine learning (ML) service platforms and models. This has given rise to the concept of "machine unlearning"—a field focused on eliminating the influence of specific samples from trained ML models. There are many hot and under-explored topics in machine unlearning, including the development of more effective and efficient unlearning algorithms, auditing machine unlearning processes, evaluating machine unlearning effectiveness, and addressing the associated privacy and security challenges in machine unlearning et al.

This thesis aims to study four critical problems: (1) optimizing unlearning effectiveness and utility preservation in traditional unlearning scenarios, (2) improving unlearning effectiveness and efficiency in federated learning, (3) evaluating machine unlearning effectiveness, and (4) protecting privacy in machine unlearning.

Specifically, in centralized machine learning scenarios, approximate unlearning is a mainstream solution to implement unlearning. However, these approximate machine unlearning methods often cause significant model utility degradation after unlearning, known as "catastrophic unlearning." In our investigation, we first formulate the unlearning problem into a two-objective optimization problem, which contains forgetting and remembering goals. We propose a representation forgetting method with parameter self-sharing to solve the catastrophic unlearning problem. Our method maximizes the removal effect of erased samples (forgetting) while preserving the model's utility (remembering) during unlearning. A two-objective optimization scheme based on parameter self-sharing is designed to achieve optimal tradeoff of the two goals.

In federated learning scenarios, the servers are limited in accessing users' local data due to privacy concerns, which constrain the implementation of unlearning. Some studies proposed to unlearn the entire contribution of one user, which is impractical and degrades model utility dramatically. We introduce a federated unlearning (FedU) method, which achieves effective and efficient unlearning through user-side influence approximation forgetting. FedU supports unlearning several specified samples and significantly mitigates the model utility degradation caused by unlearning. Our FedU obeys the federated learning mechanism that prevents the server from accessing users' data and integrates naturally into the FL framework, as only unlearning requesting users execute unlearning operations while the other users and the server normally operate as before.

After unlearning, evaluating the effectiveness of the unlearning operation is critical and necessary. However, only a few works focused on investigating unlearning verification problems, and most of them rely on backdoor techniques. The backdoor-based methods can only build the connection between the models and backdoored samples rather than the genuine samples. Thus, the removal of backdoors can only verify that the backdoored samples are unlearned. Moreover, introducing backdoors in the machine learning models will degrade the model's utility. We propose an evaluating machine unlearning (EMU) method to address the unlearning evaluation problem, which is only based on the model difference before and after unlearning, avoiding the negative impact of models. In this study, we discover two factors (the similarity between erased and remaining samples and the type of learning task) significantly influencing unlearning and uncover how they impact unlearning and evaluating, which existing works have not explored.

Moreover, although machine unlearning is proposed to guarantee users' right to be forgotten to protect their privacy, recent studies pointed out that the changes caused by unlearning leak private information about the erased samples. We propose a compressive presentation forgetting method (CRFU) to protect against privacy leakage during machine unlearning. The idea is inspired by the information bottleneck (IB) theory, which learns a representation that maximizes the compression of input data while preserving sufficient information about the learning tasks. CRFU achieves unlearning by further minimizing the mutual information between learned representations and the erased data (both inputs and labels). Since the unlearning process is based on representations, which maximizes the distortion of both the original training data inputs and the erased data inputs, the model's output contains less information that adversaries can exploit. Consequently, CRFU can effectively defend against privacy leakage attacks during machine unlearning.

In summary, in this thesis, we investigate four fundamental problems in machine unlearning and propose corresponding solutions to tackle these challenges. Our research aims to fill existing gaps in machine unlearning research, providing robust approaches for implementing unlearning, evaluating unlearning, and protecting privacy throughout the unlearning process. We believe that the findings of this study contribute significantly to advancing the understanding and practical application of machine unlearning.

ACKNOWLEDGMENTS

I am profoundly grateful to Professor Shui Yu for his invaluable guidance, unwavering support, and profound insights throughout my Ph.D. journey. Being under the supervision of Professor Yu has been an incredibly fortunate experience. I sincerely thank him for his selflessness, sparing no effort to guide me to ensure that I am on the right track of research. My sincere thanks also go to my co-supervisor Professor Bo Liu, especially for his help in this thesis. I am also grateful to my master's supervisor, Professor An Liu, for his valuable guidance and assistance.

I wish to express my gratitude to the Graduate Research School of University of Technology Sydney and Australian Research Council (ARC DP200101374) for their financial support of my research. Their support has been instrumental in enabling the progression and completion of this work.

I also would like to extend my gratitude to all the panel members involved in the candidature assessment, Professor Tianqing Zhu, Professor Yi Zhang, Professor Shoujin Wang, and others. Their equitable evaluations and constructive feedback have been invaluable in shaping the course of my research.

I also could not have undertaken this journey without my colleagues and academic collaborators, including Dr. Chenhan Zhang, Dr. Zhiyi Tian, Dr. Shushu Liu, Mingjian Tang, and others. Learning and discussing with them has provided significant inspiration and encouragement throughout my academic journey. I cherish and appreciate this memory of working together.

I also want to express my thanks to my friends, Jin Xu and Zengzhuang Xu, who, despite infrequent in-person meetings, have consistently been there for me online, offering immense courage and comfort.

Lastly, my deepest appreciation goes to my parents, Xiang Ge and Songyun Wang, my grandparents, Liuxin Wang and Xiuqun Wang, and my girlfriend, Jing Ren. I am fortunate to have been surrounded by love every step of the way. While the Ph.D. journey is concluding, my love and gratitude for them will endure endlessly.

Weiqi Wang
25th November, 2024
Sydney, Australia

List of Publications

Journals :

1. **Weiqi Wang**, Chenhan Zhang, Zhiyi Tian, and Shui Yu. Machine unlearning via representation forgetting with parameter self-sharing. IEEE Transactions on Information Forensics and Security, vol. 19, pp. 1099-1111, 2024.

2. **Weiqi Wang**, Zhiyi Tian, Chenhan Zhang, and Shui Yu. SCU: An Efficient Machine Unlearning Scheme for Deep Learning Enabled Semantic Communications. IEEE Transactions on Information Forensics and Security (TIFS, Accepted).

3. **Weiqi Wang**, Chenhan Zhang, Zhiyi Tian, and Shui Yu. FedU: Federated Unlearning via User-Side Influence Approximation Forgetting. IEEE Transactions on Dependable and Secure Computing (TDSC, Accepted).

4. **Weiqi Wang**, Chenhan Zhang, Zhiyi Tian, Shushu Liu, and Shui Yu. CRFU: Compressive Representation Forgetting Against Privacy Leakage on Machine Unlearning. IEEE Transactions on Dependable and Secure Computing (TDSC, Accepted).

5. Chenhan Zhang, **Weiqi Wang**, Zhiyi Tian, and Shui Yu. Forgetting and remembering are both you need: Balanced graph structure unlearning. IEEE Transactions on Information Forensics and Security, vol. 19, pp. 6751-6763, 2024.

6. Zhiyi Tian, Chenhan Zhang, **Weiqi Wang**, Hanna Bogucka, and Shui Yu. ROSE: A Receiver-Oriented Semantic Communication Framework. IEEE Network, 2024, Accepted.

CONFERENCES :

1. **Weiqi Wang**, Zhiyi Tian, An Liu, and Shui Yu. TAPE: Tailored Posterior Difference for Auditing of Machine Unlearning. WWW25' (Accepted).

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

3. **Weiqi Wang**, Shushu Liu, Chenhan Zhang, Mingjian Tang, Feng Wu, and Shui Yu. CP-FL: Practical gradient leakage defense in federated learning with compressive privacy. In IEEE Global Communications Conference, 2023, pp. 5165–5170.

4. **Weiqi Wang**, Chenhan Zhang, Shushu Liu, Mingjian Tang, An Liu, and Shui Yu. FedMC: Federated learning with mode connectivity against distributed backdoor attacks. In IEEE International Conference on Communications, 2023.

5. **Weiqi Wang**, Shushu Liu, An Liu, Christy Jie Liang, and Shui Yu. Locally random sampling for practical privacy protection in federated learning. In IEEE Global Communications Conference, 2022, pp. 528–533.

TABLE OF CONTENTS

List of Figures

Figure **Page**

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Background

Over the past decade, enormously increased data and fast hardware improvement have driven machine learning developments quickly. Machine learning (ML) algorithms and artificial intelligence (AI) are embedded into day-to-day applications and wearable devices [63]. It continuously collects increasing amounts of user information, including private data such as driving trajectories, medical records, and online shopping histories [46, 74]. On the one hand, such an enormous amount of data helps to further advance ML and AI development. On the other hand, however, it poses a threat to users' privacy and creates a significant need for robust data management to ensure information security and privacy in ML [87].

Machine unlearning has drawn growing research attention as the recent legislation of the "Right to be Forgotten" in many countries. Notable instances include the GDPR in the European Union [65], the PIPEDA privacy legislation in Canada [69], and the California Consumer Privacy Act in the United States [19]. According to these laws, companies must take reasonable measures to guarantee that personal data is deleted upon request. It indicates that individual users have the right to request companies to remove their private data, which was previously collected for ML model training. The deletion is not only erasing their data from a database, but it also needs to delete the influence of the specified samples from trained models. The process of data removal from models was first conceptualized as machine unlearning [12]. We present the general

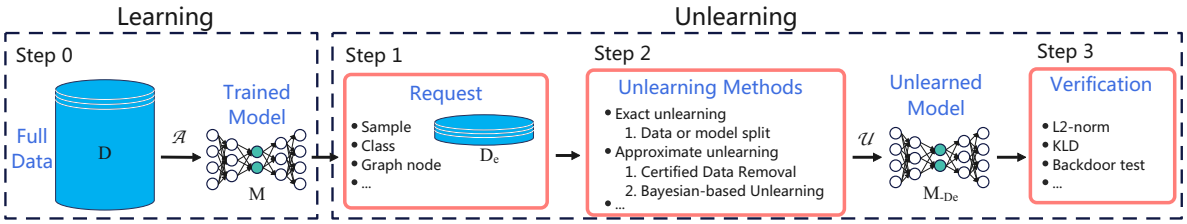


Figure 1.1: Machine unlearning process. 0) A ML model M was trained using the learning algorithm $\mathscr{A}$ based on dataset D. 1) A erased dataset D_e is requested for unlearning. 2) Unlearning methods $\mathscr{U}$ are condusted for data removal, outputing the unlearned model M_{-D_e}. 3) Verification methods are employed to evaluate unlearning effectiveness.

machine unlearning process in Figure 1.1. Specifically, suppose a user (Alice) wants to exercise her right [65] when quitting a ML application, then the trained model of such application must "unlearn" her data. Such a process of unlearning data from trained models includes three steps: first, a subset of the dataset previously used for ML model training is requested to be deleted; second, the ML model provider erases the contribution of these data from the trained models; third, the unlearning effectiveness is evaluated. A naive data-erasing method is retraining a new ML model from scratch [11]. However, the computation and storage costs of retraining are expensive, especially in complex learning tasks.

Many researchers have tried to find efficient and effective methods to implement unlearning rather than naive retraining, and their methods can be roughly divided into two kinds: exact unlearning and approximate unlearning. The exact retraining methods are extended from naive retraining and aim to reduce the computational cost of retraining a new model by redesigning the learning algorithms [11] and storing the intermediate parameters during the learning process [104]. Approximate unlearning seeks to modify the model using only the original model and the samples to be erased, approximating a model as if it were retrained on the remaining dataset [36, 67, 97, 98].

After conducting machine unlearning methods, researchers also need evaluation methods to audit the effectiveness of the unlearning methods. Distance metrics, such as L_2-Norm [101] and Kullback-Leibler divergence (KLD) [43, 67], are used to compare the difference between models before and after unlearning. Moreover, some studies also use the membership inference attacks to detect whether a sample was utilized for training the model is removed after unlearning [15, 38, 40]. Especially in [38, 40], they directly use the backdoored samples as the unlearned data to test the unlearning methods. Effective unlearning methods should significantly remove the influence of backdoors.

1.2 Research Problems

As previously discussed, although machine unlearning aims to support the "right to be forgotten" by removing the influence of specified samples from trained models, there are still many under-explored areas and challenges in machine unlearning. In this thesis, our focus is on addressing four critical challenges. Problem 1 and Problem 2 aim to solve the mainstream effective and efficient unlearning in centralized and distributed scenarios, respectively. After executing machine unlearning, auditing and evaluating unlearning effectiveness is essential. Hence, our Problem 3 focuses on how to audit the unlearning effectiveness. Additionally, existing studies find the privacy threats during machine unlearning operations. We discuss and solve the privacy leakage issue of machine unlearning in Problem 4. These four problems collectively cover a broad scope, from implementing unlearning to auditing unlearning outcomes and mitigating privacy risks. Their comprehensive approach provides a solid foundation for advancing research on machine unlearning. The details of the 4 problems are formulated as follows.

Research Problem 1: How can we achieve the optimal balance between the effectiveness and model utility preservation of machine unlearning?

Machine unlearning draws a lot 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 [11, 12, 14, 104] 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 [100, 101]. Additionally, cumulatively removing data from a model can lead to a significant decrease in accuracy, known as catastrophic unlearning [67]. 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.

In centralized machine unlearning, a few works proposed approximate unlearning methods to address the problem of removing data's impact from trained models. For example, Guo et al. [36] and Sekhari [78] introduced certified removal, a technique based on the Hessian matrix and the Newton update approach [103]. Their methods aim to make the unlearned model be ϵ -indistinguishable from the model that retrained based on the remaining dataset except for the erased samples. Fu et al. [28] and Nguyen et al. [67] introduced knowledge removal for unlearning Bayesian inference models [26]. Both the Hessian-based unlearning methods [36, 66, 78] and Bayesian inference un-

learning methods [28, 67, 68] 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.

Research Problem 2: How can we achieve effective and efficient unlearning in federated learning scenarios?

In federated unlearning, there are two main challenges: the inaccessibility of erased data for the server and the balance between unlearning and maintaining FL model utility. Firstly, in FL, users upload their model updates instead of their local datasets. Since the model update is a highly compressed representation of the local training dataset, it is challenging for the FL server to remove the information of a small set of training samples from the model without access to these data [55, 58]. A typical solution simplifies the problem of unlearning a small set of training samples by considering it as unlearning the entire local dataset of the requested users [39, 56]. However, completely removing the contributions of the unlearned user significantly compromises model utility. Secondly, existing federated unlearning related studies [58, 83, 93, 99] concentrate on optimizing the efficiency of the unlearning process. They usually only employ a fixed threshold to constrain the extent of updates and prevent significant losses in model utility due to unlearning. Nonetheless, utilizing a fixed threshold always falls short of striking an ideal balance between the effectiveness of unlearning and maintaining the utility of the original model [62].

Research Problem 3: How can we audit and evaluate the machine unlearning effectiveness after unlarning?

Among existing machine unlearning research, only a few works have attempted to verify unlearning using backdoor techniques [38, 40, 82]. However, the backdoor-based methods can solely establish the connection between models and backdoored data, failing to link the backdoored data and genuine data as these two datasets are distinct during learning and unlearning [30, 70, 91]. Specifically, in approximate unlearning [17, 28, 67], the model predicting accuracy on backdoored datasets declines much quicker than on genuine datasets [53, 70]. Thus, removing backdoors only verifies that backdoored samples are unlearned, not genuine ones. Moreover, backdoor-based methods need to embed backdoor triggers into the trained models, which negatively influences the model utility [30, 53]. *To date, the research question still remains: no work has provided a comprehensive evaluation method for machine unlearning effectiveness*

without compromising model utility.

Research Problem 4: How can we safeguard the privacy of unlearned data in machine unlearning?

Although machine unlearning aims to preserve users' privacy by removing their data from trained models, recent studies have revealed that the changes induced by unlearning can leak private information about the erased data. For instance, Chen et al. [15] proposed membership inference attacks to infer which data samples are present in the erased dataset. Additionally, Zhang et al. and Hu et al. [29, 41, 106] identified a further attack, known as privacy reconstruction, capable of recovering deleted data based on unlearning updates. Similarly, [76] introduced the privacy reconstruction attack aimed at recovering specific data points used in model updates. The privacy leakage caused by unlearning updates has become a critical issue in machine unlearning, and few works have effectively addressed this threat, particularly concerning reconstruction attacks.

Addressing potential privacy breaches in machine unlearning is critical and requires urgent attention. However, finding an effective solution remains a significant challenge. Firstly, existing unlearning methods, while effective, conflict with privacy leakage defenses because unlearning necessitates the complete removal of the specified samples' information from the trained model. This typically results in different outputs for the same query set before and after unlearning, which adversaries can exploit to infer private information about the erased data. Secondly, directly incorporating defense strategies such as differential privacy [24] into unlearning methods exacerbates model utility degradation, leading to what is termed "unlearning catastrophic" [22, 67], rendering the model unusable. The key challenge lies in designing an unlearning method that reduces the private information in the model output to prevent inference attacks while still achieving effective unlearning performance.

1.3 Thesis Contributions

In this thesis, we try to investigate and solve the above four problems in machine unlearning. Correspondingly, we propose four solutions and briefly introduce them as follows.

To solve the catastrophic accuracy degradation problem in centralized approximate unlearning, i.e., **Research Problem 1**, we systematically address both erasure effectiveness and model utility preservation in unlearning by formulating it as a two-

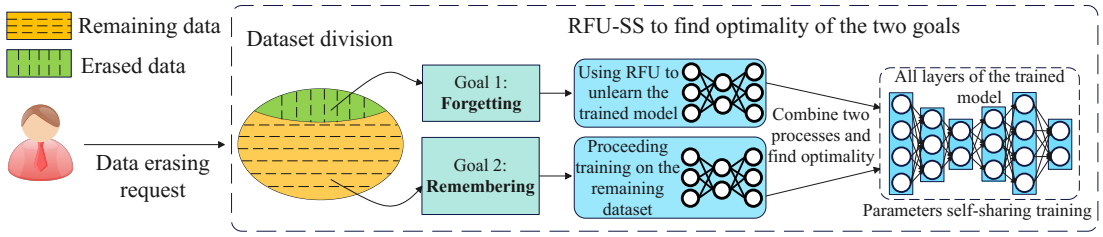


Figure 1.2: Process of a two-objective unlearning problem.

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 1.2. 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 [2, 88], and we introduce our RFU method following this IB structure for ease of understanding. Second, 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) [44, 79]. 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.

We propose a federated unlearning (FedU) scheme via user-side influence approximation forgetting to investigate the effective and efficient federated unlearning problem (**Research Problem 2**), as shown in Figure 1.3. In FedU, only the unlearning requester executes the influence approximation forgetting method; the server and other users perform as usual in FL. Specifically, the influence approximation forgetting method assesses the impact of specified samples on the trained FL model using only the user's local data. Subsequently, this estimated influence is subtracted from the model to facilitate unlearning. However, despite the retraining efforts by other users, directly eliminating the data influence still tends to reduce the model utility significantly [28, 67]. To compensate for the degradation of model utility resulting from unlearning, we design

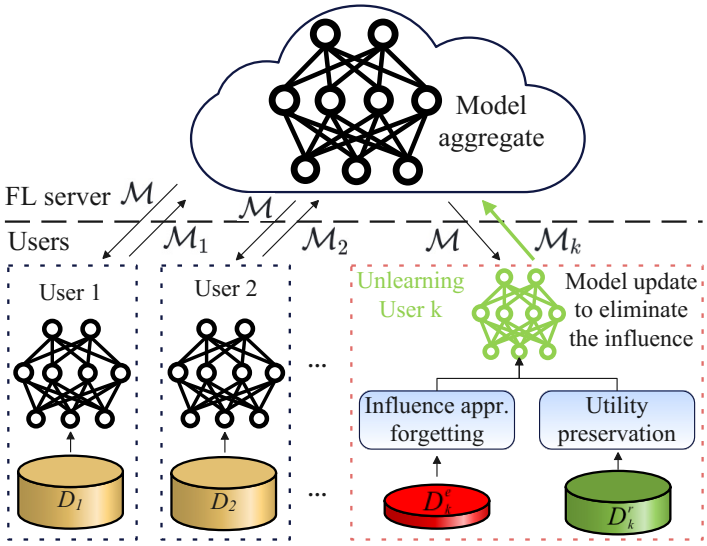


Figure 1.3: FedU is naturally integrated into FL frameworks.

a utility preservation method that simultaneously trains the unlearned model using a sub-dataset sampled from the unlearning user's remaining local dataset. The utility preservation method helps to rectify the model parameters that are excessively updated by unlearning, which is named FedU-U, i.e., federated unlearning using user-side influence approximation forgetting with utility preservation. Combining influence approximation forgetting and utility preservation methods, we design an adaptive optimizing process to train them simultaneously in the steepest descent direction and to obtain the optimal balance.

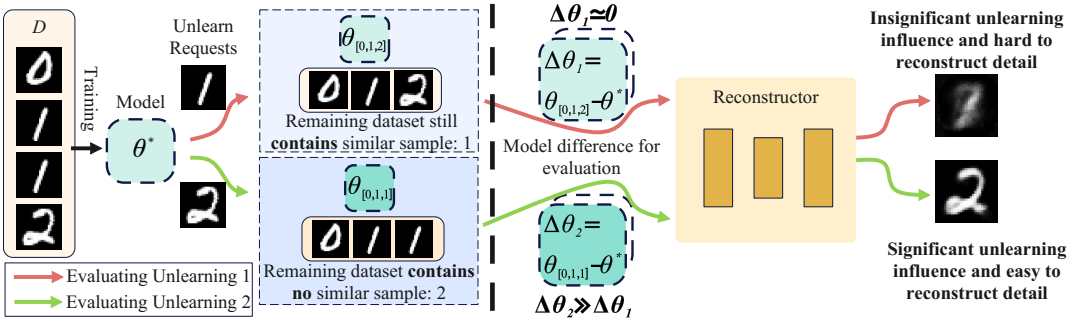


Figure 1.4: A contrastive example of auditing unlearning digital 1 (red arrows) and 2 (green arrows) from a model trained using samples $[0,1,1,2]$.

We propose a solution called EMU to assess the extent of unlearning effectiveness, which aims to solve the **Research Problem 3**. Machine unlearning operations unavoidably create two versions of the model: one before and one after unlearning [15]. These model differences usually contain the private information of the unlearned samples

[15, 41, 106]. EMU utilizes these model differences to assess the effectiveness of unlearning. Specifically, in EMU, we exploit the first-order influence function to efficiently simulate the model differences of unlearning for the unlearning evaluating analysis. Additionally, to improve the scalability of EMU for large models, we design a multi-task information bottleneck (MT-IB) structure for the learning service model and only calculate the difference of the representation layer to simplify the model difference. This structure also streamlines the evaluation process across different types of learning tasks. Based on the influence function theory and the MT-IB structure, we theoretically analyze how the similarity between erased and remaining samples, as well as task types, impacts unlearning effectiveness. We show an example of EMU when evaluating unlearning different samples in Figure 1.4.

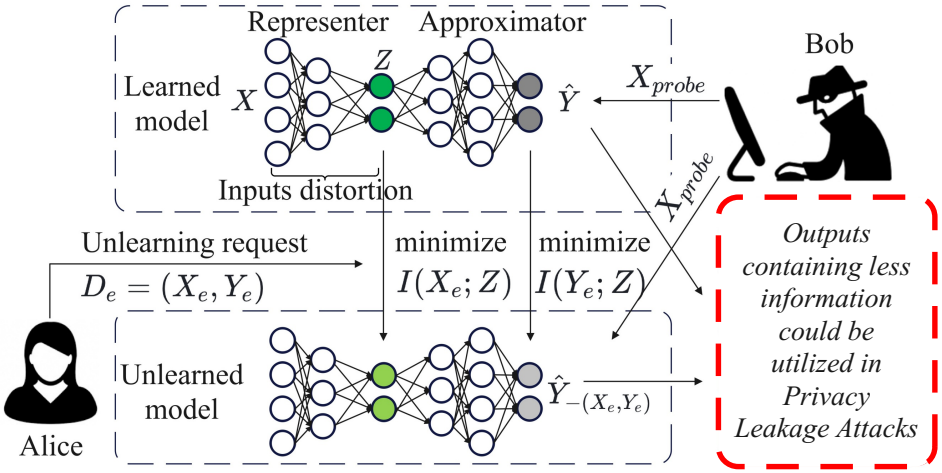


Figure 1.5: Compressive representation forgetting unlearning (CRFU) to defend against privacy leakage on unlearning.

To investigate **Research Problem 4**, we propose a Compressive Representation Forgetting Unlearning (CRFU) scheme to defend against privacy leakage on machine unlearning, as illustrated in Figure 1.5. The CRFU scheme is tailored for unlearning models trained using information bottleneck (IB) theory [88]. An IB-trained model comprises two components, a representer and an approximator, which aim to maximize the compression of input data while preserving sufficient information relevant to the labels in the learned representation. CRFU implements unlearning for both components. Specifically, CRFU further minimizes the mutual information between learned representations and the erased data (both inputs and labels) to remove the information of these samples in IB-trained models. Since the unlearning process is based on representations, which maximizes the distortion of both the original training data inputs and

the erased data inputs, the model's output contains less information that adversaries can exploit. Consequently, CRFU can effectively defend against privacy leakage attacks for a black-box ML model [71]. However, directly optimizing unlearning methods would lead to significant model utility degradation, known as "catastrophic unlearning." To counteract model utility decline, we introduce a remembering constraint and an unlearning rate in CRFU. Adjusting the unlearning rate helps balance the trade-off between completely removing the influence of specified samples and preserving previously acquired knowledge.

We summarize our key contribution of the thesis as follows:

- 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. Then, 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. Finally, 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.

- We explore the federated unlearning problem and propose the federated unlearning (FedU) framework via user-side influence approximation forgetting. Our method integrates naturally into the FL framework, as only unlearning requesters carry out the unlearning operation while the other users and the server typically operate as before. To compensate for utility reduction caused by unlearning, we further proposed a utility preservation method and developed an adaptive optimization scheme to attain the ideal equilibrium between forgetting effectiveness and utility preservation.

- Given the only update of the unlearning operation, the model difference, we propose the EMU method based on it to evaluate machine unlearning. EMU exploits the influence function to mimic the model difference efficiently and designs an MT-IB architecture to simplify model differences and improve scalability. The corresponding theoretical analysis is also provided. In our study, we discover two fac-

tors significantly influencing unlearning and uncover how they impact unlearning and evaluating, which existing works have not explored.

- We study the defense of privacy leakage on unlearning. Our approach, compressive representation forgetting unlearning (CRFU), can achieve a significant defense effect and is easy to extend to support most ML algorithms. We also introduce a remembering constraint and an unlearning rate in CRFU to achieve a balance between forgetting the erased samples and remembering previously learned knowledge. Adjusting the unlearning rate can control the unlearning extent and effectively mitigate the utility decline.

1.4 Thesis Organization

This thesis is presented as a *thesis by compilation*. Our focus in on enhancing the effectiveness and model utility preservation of unlearning, evaluation of machine unlearning, and safeguarding privacy in machine unlearning. The thesis organization is presented in the following Table 1.1.

Table 1.1: Thesis organization

Chapter	Corresponding Problem	Publication	Brief Introduction
Chapter 2	-	-	**Literature Review:** We thoroughly review the research progress of machine unlearning in recent years.
Chapter 3	**Resarch Problem 1:** How can we achieve the optimal balance between the effectiveness and model utility preservation of machine unlearning?	[97]	We present the results of addressing the catastrophic unlearning problem, where we introduce the representation forgetting via parameter self-sharing.
Chapter 4	**Research Problem 2:** How can we achieve effective and efficient unlearning in federated learning scenarios?	[96]	We introduce our work on how to address the federated unlearning problem using the user-side influence approximation forgetting method.
Chapter 5	**Research Problem 3:** How can we audit and evaluate the machine unlearning effectiveness after unlarning?	[95]	We present the study of how to evaluate both the data removal and unlearning effectiveness, exploring the influence factors of unlearning.
Chapter 6	**Research Problem 4:** How can we safeguard the privacy of unlearned data in machine unlearning?	[94]	We study the privacy leakage on machine unlearning and propose the compressive representation forgetting method (CRFU) to prevent the privacy leakage attack on the server side.
Chapter 7	-	-	**Summary:** We conclude our thesis and discuss future work.

CHAPTER 2

LITERATURE REVIEW

Machine unlearning has garnered significant research attention as a solution to the "right to be forgotten." In this chapter, we present an overview of key related work. We focus on centralized machine unlearning algorithms, specifically exact and approximate machine unlearning methods. Additionally, we examine distributed unlearning techniques, with an emphasis on federated unlearning. Furthermore, we discuss studies of privacy leakage attacks in machine unlearning, including membership inference attacks and privacy reconstruction attacks.

2.1 Machine Unlearning Algorithms

From the former introduction, we know that naive retraining is the most effective manner to realize machine unlearning. However, it is inefficient because it requires storing the entire original dataset and retraining the model from scratch, which consumes significant storage and computational resources, especially in deep learning scenarios. Therefore, researchers tried to design effective and efficient unlearning mechanisms. Exact unlearning was proposed to reduce the computation cost by splitting training sub-models based on pre-divided data sub-sets [11]. It also divides the stochasticity and the incrementality into sub-models. It can unlearn an exact model by ensembling the consisting submodels, but it still needs to store all the split data subsets for fast retraining. Another approach is approximate unlearning [78]. It can reduce both storage and computation costs because it unlearns only based on the erased data. However, it is diffi-

cult for approximate unlearning methods to control the accuracy degradation due to the challenges in estimating stochasticity and incrementality during the training process. Most of them bounded their estimation to avoid dramatic accuracy reduction.

2.1.1 Exact Machine Unlearning

Exact unlearning could also be called fast retraining, whose basic idea is derived from naive retraining from scratch. Following the background introduction of unlearning, we know the learning and unlearning algorithm, $\mathscr{A}(D)$ and $\mathscr{U}$, based on the trainset D and erased dataset $D_e \subseteq D$, respectively. If $\mathscr{U}(\cdot)$ is implemented as naive retraining, the equality between $\mathscr{A}(D \setminus D_e) \in \mathscr{H}$ and $\mathscr{U}(M, D, D_e) \in \mathscr{H}$ is absolutely guaranteed. However, naive retraining involves high computation and storage costs, especially for deep learning models and complex datasets [85]. Unlike naive retraining, which relies on the whole remaining dataset, exact unlearning tries to retrain a sub-model only using a subset of the remaining dataset to reduce calculation cost. A general operation of exact unlearning is that they first divide the dataset into several small sub-sets. Then, they transform the learning process by ensembling the sub-models trained with each sub-set as the final model [11, 12]. So that, when an unlearning request comes, they are just required to retrain the sub-model corresponding to the sub-set containing the erased data. They then ensemble the retrained sub-model and other sub-models as the unlearned model.

As we introduced above, exact unlearning aims to mitigate the computation cost when retraining a new model by transforming the original learning algorithms into an ensembling form. It divides the stochasticity and incrementality into several sub-models to reduce their influence. However, to some extent, they sacrificed the storage cost because they needed to store the whole training dataset in a divided form. In [12], Cao et al. transformed the traditional ML algorithms into a summation form. They are only required to update several summations when an unlearning requirement comes, ensuring the method runs faster than retraining from scratch. SISA [11] is a representative exact unlearning algorithm, which splits the full training dataset into shards and trained models separately in each shard. For unlearning, they simply need to retrain the shard that includes the erased data. Study [100] proposed a framework that precisely models the impact of individual training samples on the model concerning various performance criteria and removes the impact of samples that are required to be removed. Golatkar et al. [32] proposed an unlearning method on deep networks, splitting the trained model into two parts. The core part based on the data will not be deleted, and the unlearning

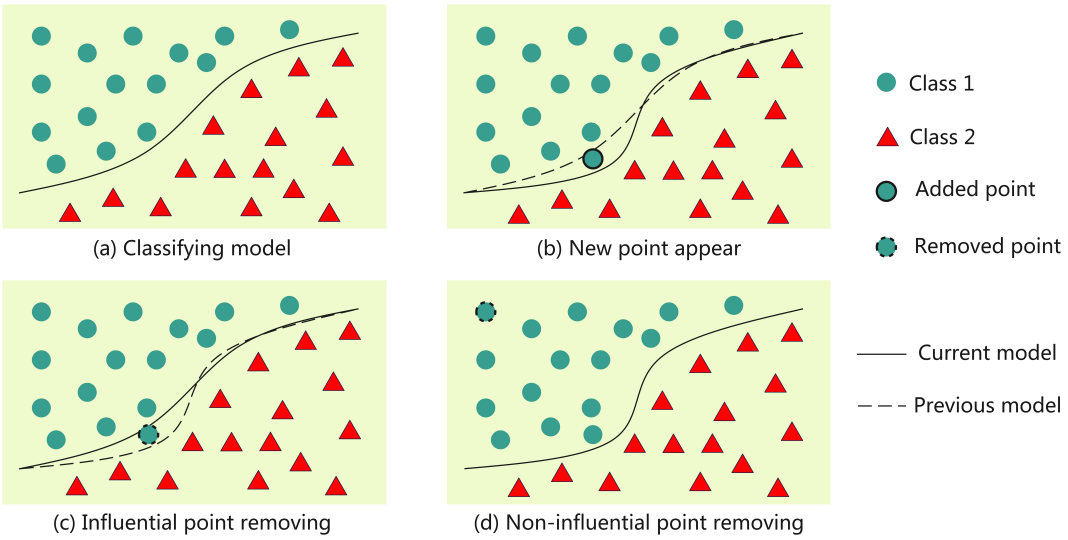


Figure 2.1: The model changes when adding a new point or removing a point.

part with the erased data will be unlearned with parameters bound. These methods are efficient in computation, but they sacrifice the storage space to store the intermediate training parameters of different slices and the related training sub-sets.

Besides the high storage cost, another major issue with exact unlearning is that it is only suitable for scenarios where the unlearning request involves removing a few samples with low frequency. Suppose an unlearning request needs to remove many data samples (usually, they are not in the same previous divided sub-set). In that case, exact unlearning must retrain all these related sub-models or even all the sub-models in the worst situation. At the moment, exact unlearning is no longer computation efficient, and the whole training dataset and intermediate parameters still need to be stored.

2.1.2 Approximate Machine Unlearning

Unlike exact unlearning, which only aims to reduce the retraining computation cost, approximate unlearning tries to directly unlearn based on the trained model and the erased data sample, which saves the computation and storage costs together. Approximate unlearning studies aim to unlearn a model that approaches the model trained on the remaining dataset, i.e., the unlearned model $\mathscr{U}(M, D, D_e)$ should match the retrained model $\mathscr{A}(D \setminus D_e)$. Since exact unlearning is implemented by retraining from the remaining dataset or sub-sets, they can almost guarantee equality before and after unlearning. However, since approximate unlearning tries to directly delete the influence of the unlearned samples from trained models, the core problem lies in precisely

estimating and removing this contribution, which includes both stochasticity and incrementality.

The text description of the changes between two different distribution spaces before and after removing the specific data is not intuitive. Fig. 2.1 shows illustrated changes when adding a new point or removing a point in a classification model. When an influential point appears, it usually pushes the line to move forward than the original classifying line to identify it, as shown in Fig. 2.1 (b). When this influential point is requested to be removed, the unlearning mechanism must recover the model to the original one that has not been trained by this specific point, as shown in Fig. 2.1 (c). However, when only unlearning a non-influential point, which may have almost non-influence on the model, the unlearned model may not change compared to the original trained model in this situation, as shown in Fig. 2.1 (d).

Many methods were proposed to implement approximate unlearning efficiently and effectively. The popular solutions are certified-removal [36] and Bayes-based mechanisms [67]. We briefly summarize current studies about approximate unlearning as follows. A certified-removal mechanism [36] was an approximate unlearning method similar to differential privacy. Guo et al.[36] defined the $\epsilon-$approximate unlearning, which makes sure that the model after and before unlearning must be ϵ-indistinguishable as the definition in DP. The difference between $\epsilon-$approximate unlearning and ϵ-DP is that the mechanism $\mathscr{A}$ on differential privacy is needed never to memorize the data in the first place, which is impossible in machine unlearning. The machine learning model does not learn anything from the training dataset if $\mathscr{A}$ is differentially private [11]. A similar solution to [36] is introduced in [78], which erases the influence of the specified samples from the gradients during unlearning. To mitigate accuracy degradation, model parameter updates are constrained by a differential unlearning definition.

Different from indistinguishable unlearning, other methods used the Bayes theorem to unlearn an approximate posterior. In [67], Nguyen et al. employed Bayesian inference to unlearn an approximate model using the erased data. Moreover, Fu et al. [28] developed a similar unlearning method, which uses Markov chain Monte Carlo (MCMC) methods. In [68], the authors also explained the effectiveness of the approximate unlearning method from the perspective of MCMC. Nevertheless, since those techniques are approximately unlearning the contribution of all the erasing data, including the inputs and labels, they inevitably decrease the model accuracy to some extent after unlearning.

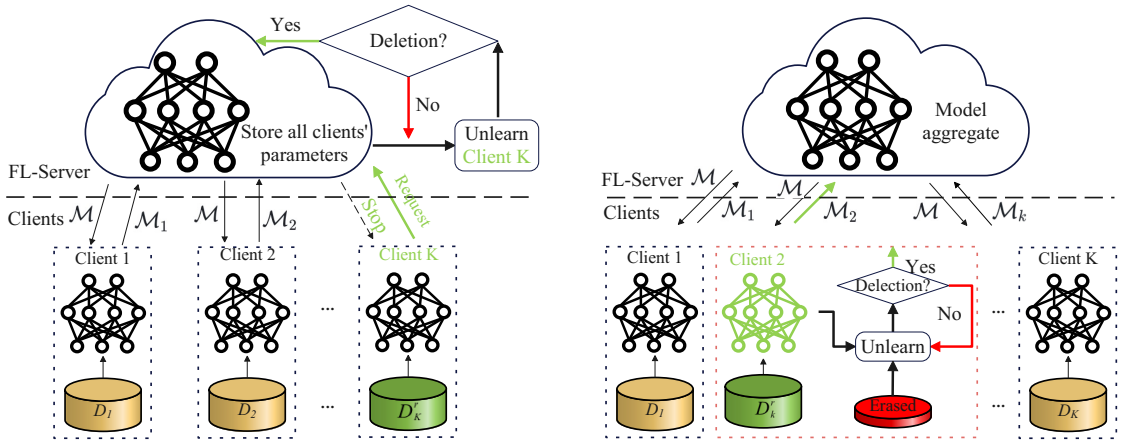
2.2 Federated Unlearning

We have introduced popular centralized machine unlearning in the former sub-chapter. Their main purpose is to reduce the calculation and storage costs and the accuracy degradation caused by unlearning. Recently, we noticed that some researchers paid attention to realizing unlearning in other scenarios, such as in federated learning (FL) scenarios. In this section, we will introduce the literature about federated unlearning.

FL was initially introduced to protect the privacy of participating clients during the machine learning training process in distributed settings. All participants will only upload their locally trained model parameters instead of their sensitive local data to the FL server during model training processes. Therefore, in a federated learning scenario, limited access to the dataset will become a unique challenge when implementing unlearning.

Since the local data cannot be uploaded to the federated learning (FL) server side, most federated unlearning methods try to erase a certain client's contribution from the trained model by storing and estimating the contribution of uploaded parameters. In this situation, they can implement federated unlearning without interacting with the client, shown as the server-side federated unlearning in Fig. 2.2 (a). The two representative methods are [56, 99]. Liu et al. [56] proposed "FedEraser" to sanitize the impact of a FL client on the global FL model. In particular, during the FL training process, the FL-Server maintains the updates of the clients at each routine iteration and the index of the related round to calibrate the retrained updates. Based on these operations, they reconstructed the unlearned FL model instead of retraining a new model from scratch. However, FedEraser can only unlearn one client's data, which means it must unlearn all the contributions of this specific client's data. It is unsuitable for a client who wants to unlearn a small piece of his data. Study [99] tried to erase a client's influence from the FL model by removing the historical updates from the global model. They implemented federated unlearning by using knowledge distillation to restore the contribution of clients' models, which does not need to rely on clients' participation and any data restriction.

Wang et al. [92] explored the problem of how to selectively unlearn a category from a trained CNN classification model in FL. The biggest challenge is that the local data utilized for the FL global model training cannot be accessed globally. Therefore, they explored the inner influence of each channel and observed that various channels have distinct impacts on different categories. They proposed a method that does not require



(a) Server-side federated unlearning (b) Client-side federated unlearning

Figure 2.2: Comparison between (a) server-side federated unlearning and (b) client-side federated unlearning

accessing the data used for training and retraining from scratch to cleanly scrub the information of particular categories from the global FL model. A method called TF-IDF was introduced to quantize the class discrimination of channels. Then, they pruned the channels with high TF-IDF scores on the erased classifications to unlearn them. Evaluations of CIFAR10 and CIFAR100 demonstrate the unlearning effect of their method.

Different from unlearning a whole client's influence and unlearning a class, Liu et al. [58, 93] investigated how to unlearn data samples in FL, shown as the client-side federated unlearning in Fig. 2.2 (b). In [58], they first defined a federated unlearning problem and proposed a fast retraining method to withdraw the influence of data from the FL model. Then, they proposed an efficient federated unlearning method following the Quasi-Newton methods and the first-order Taylor approximate method [58]. They utilized the practical Fisher Information Matrix to model the Hessian matrix at a low cost. When unlearning, they subtract the estimation matrix from gradients with a bound. In [93], they implement federated unlearning based on Bayesian inference, and they propose a parameters self-sharing method to reduce the model utility degradation.

2.3 Privacy Leakage in Machine Unlearning

Although unlearning methods were initially proposed to safeguard users' privacy, many researchers have noticed that unlearning brings new privacy threats simultaneously. Similar to machine learning scenarios, there are two main kinds of privacy threats in unlearning. They are membership inference attacks and private information re-

construction attacks. In this section, we detail the two prevalent unlearning privacy threats.

2.3.1 Membership Inference Attacks on Machine Unlearning

Chen et al. [15] first pointed out that when a model is unlearned, the discrepancy in the outputs from the model before and after unlearning leaks the privacy of erased data. Then, they proposed the corresponding membership inference attack pipeline in unlearning, which includes three phases: posterior generation, feature construction, and membership inference.

1. Posteriors Generation. Suppose that the attacker can access two versions of the trained model, the model $\theta_{\mathscr{A}}$ before unlearning and the model $\theta_{\mathscr{U}}$ after unlearning. Assume a target data point e, the attacker queries $\theta_{\mathscr{A}}$ and $\theta_{\mathscr{U}}$, and has the corresponding posteriors, $p(\theta_{\mathscr{A}})$ and $p(\theta_{\mathscr{U}})$, which also called as confidence values in [81].

2. Feature Construction. After achieving the two posteriors $p(\theta_{\mathscr{A}})$ and $p(\theta_{\mathscr{U}})$, the attacker sums them to make the inference feature vector F. Common method to construct the feature vector is shown in [15].

3. Inference. After the attacker finishes training the attack model based on the created features F, he inputs the collected feature to the inference model to predict if the specific sample e occurs in the erased dataset of unlearning models.

In [15], they assumed that the attacker has admission to two ML models before and after unlearning, but it is sometimes impractical, especially in black-box learning scenarios. Lu et al. [60] further proposed a label-only membership inference method to imply if a sample is unlearned, eliminating the dependence on accessing posteriors. Their basic idea is that the same noise injection on candidate data points will show different results for the sample in or not in the training dataset. Thus, they made the adversary continuously query the original and unlearned models and add noise to modify their outputs. Observing the disturbance amplitude lets them determine whether an item is deleted.

Golatkar et al. [33] derived an upper bound to confirm the maximizing knowledge that can be extracted from a black-box model. They queried the model with a picture and obtained the related output. They used the entropy of the result probabilities to

construct an effective black-box membership inference [80] attack in machine unlearning.

2.3.2 Privacy Reconstruction Attacks on Machine Unlearning

Privacy reconstruction is another popular attack in machine unlearning. In an unlearning scenario, Gao et al. [29] proposed the deleted reconstruction attacks to recover the removed data from the outputs of the original and unlearning models. In their work, they formalized erasure inference and erasure recovering attacks. The attacker seeks to infer which sample is removed or recovers the erased sample. In particular, for the deletion inference, they formulate the objective of an erasure inference to decide if a data instance e was in or not in the erased dataset, $e \in D_e$ or $e' \notin D_e$. For the deletion reconstruction, they focused on reconstructing the erased example e. In all their reconstruction attacks, the attacker does not have any particular samples, and the purpose is to extract the features knowledge of the erased example. Specifically, the deletion reconstructions include the deleted instance reconstruction and deleted label reconstruction. As named, the deleted instance reconstruction is to extract all of the information of the erased example, and the erased label reconstruction is to infer the label of the erased point in the classification problem.

Zanella et al. [105] indicated that the releasing snapshot of overlapped language models would leak the privacy of the training dataset. They verified that the model updates significantly threaten the private information added to or deleted from the training dataset by many experimental results. Zanella et al. found five phenomenons. First, an attacker can extract particular sentences used or not in the training dataset by comparing two models. Second, analyzing more model snapshots shows more information about the updated data than considering fewer model snapshots. Third, adding or deleting other non-private data during model updates can not mitigate privacy leakage. Fourth, differential privacy can reduce privacy leakage risks and decrease trained models' accuracy. Fifth, to mitigate the privacy leakage risks while keeping the model utility, the server can limit the model parameters access or only output a subset of the results.

Many studies further utilized these privacy threats to evaluate the unlearning effect. Huang et al. [42] proposed Ensembled Membership Auditing (EMA) for auditing data erasure. They use the membership inference to assess the removing effectiveness of unlearning. Graves et al. [34] indicated that if an attacker can infer the sensitive information that was wanted to be erased, then it means that the server has not guarded

the rights to be forgotten. Baumhauer et al. [7] developed linear filtration to sanitize classification models with logits prediction after class-wide deletion requests. They verified their methods by testing how well the method defends against privacy attacks.

Both [15] and [105] pointed out that differential privacy guarantees that a model does not reveal too much knowledge about any training sample. A DP-protected model can further guarantee the group's privacy by binding the impacts of a bunch of training samples. If using DP to protect the privacy of a bunch of $\|D \backslash D_e\|$training examples against snapshot attacks on $\theta_D, \theta_{D \backslash D_e}$, it means that $\theta_{D \backslash D_e}$ cannot be more useful than θ_D.

CHAPTER 3

MACHINE UNLEARNING VIA REPRESENTATION FORGETTING WITH PARAMETER SELF-SHARING

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. When executing unlearning, especially approximate unlearning, it is easy to over-unlearn the knowledge that was previously learned. 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 op-

timize 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`.

3.1 Preliminary

Before introducing the background knowledge, we first summarize the primary notations used in the chapter, as presented in Table 3.1. 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 $\mathscr{L}_{rep}$ for the representer θ^r, and the predicting approximation loss $\mathscr{L}_{app}$ 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 $\mathscr{L}^u$, which is combined by the representation unlearning loss $\mathscr{L}^u_{rep}$ to unlearn the representer and the approximation unlearning loss $\mathscr{L}^u_{app}$ 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.

3.1.1 Information Bottleneck

Our unlearning methods aim to remove the contribution of the erased samples from models trained using Information bottleneck (IB) [2, 88]. 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 [88, 89] approaches for optimizing the IB objective are based on the iterative Blahut Arimoto algorithm [10]. The

Table 3.1: Basic Notations of Chapter 3

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
$\mathscr{M}(\theta^r,\theta^a)$	the representer θ^r and the approximator θ^a of an IB model $\mathscr{M}$
θ^r_{fix}	the fixed representer of a trained IB model
θ^a_{fix}	the fixed approximator of a trained IB model
$\mathscr{L}_{rep}$	the representation loss to train the representer θ^r
$\mathscr{L}_{app}$	the approximation loss to train the approximator θ^a
$\mathscr{L}^u_{rep}$	the representation unlearning loss to unlearn the representer θ^r
$\mathscr{L}^u_{app}$	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

IB objective can be described as follows,

$$\mathscr{L}_{IB}=\beta I(Z;X)-I(Z;Y), \tag{3.1}$$

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 [2]. A prevalent solution involves utilizing variational inference to optimize the IB function, known as the Variational Information Bottleneck (VIB) [1, 2, 5]. 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 Equation (3.1). 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 3.2. The corresponding approximation loss function is $\mathscr{L}_{app}=-I(Z;Y)$ and representation loss function is $\mathscr{L}_{rep}=\beta I(Z;X)$. And the VIB model is denoted as $\mathscr{M}(\theta^r,\theta^a)$. The experimental optimizing loss function of a VIB

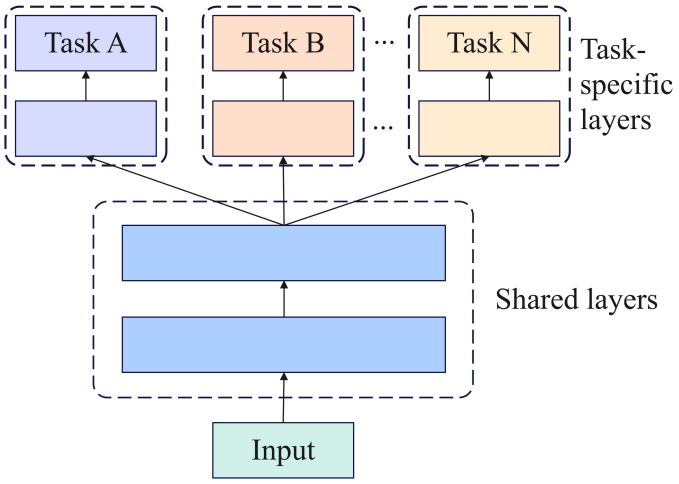


Figure 3.1: Hard parameter sharing for multi-task learning.

model is described as

$$\mathscr{L} = \mathscr{L}_{app} + \mathscr{L}_{rep} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{z\backsim p_{\theta^r}(z|x_i)}[-\log\ p_{\theta^a}(y_i|z)] + \beta D_{KL}[p_{\theta^r}(z|x_i)||\prod_{i}^{|z|} q_i(z_i)]. \tag{3.2}$$

The Equation (3.1) of IB objective can be optimized using Equation (3.2) in a variational form, where D_{KL} is the Kullback–Leibler divergence (KLD) [43]. In this chapter, we focus on finding the solution to unlearn these models trained based on the IB objective.

3.1.2 Hard Parameters Sharing

Our parameter self-sharing technique is inspired by the hard parameter sharing used in multi-task learning (MTL) [37, 54, 79]. There are two main solutions in the MTL domain: hard parameter sharing [61, 84] and soft parameter sharing [23, 75]. 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 3.1. We consider a multi-task learning problem defined over an input space $\mathscr{X}$ and a set of task spaces $\{\mathscr{Y}^t\}_{t\in[T]}$. MTL involves learning multiple tasks based on a large dataset consisting of N data points, such as $\{x_i, y_i^1, ..., y_i^T\}_{i\in[N]}$, 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 $f^t(x;\theta^{sh},\theta^t) : \mathscr{X} \rightarrow \mathscr{Y}^t$, where the parameters ($\theta^{sh}$) 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 min-

imization:

$$\min_{\theta^{sh},\theta^1,\ldots,\theta^T} \sum_{t=1}^{T} \alpha^t \mathscr{L}^t(\theta^{sh},\theta^t), \tag{3.3}$$

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

3.1.3 Multiple Gradient Descent Algorithm

It is possible to convert a MTL problem into a multi-objective optimization problem [79] 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) [21, 79]. It is based on the Karush-Kuhn-Tucker (KKT) [49] conditions, which are necessary for optimality [21, 25, 77]. 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 $\sum_{t=1}^{T} \alpha^t \nabla_{\theta^{sh}} \mathscr{L}^t(\theta^{sh},\theta^t) = 0$
- For all tasks t, $\nabla_{\theta^t} \mathscr{L}^t(\theta^{sh},\theta^t) = 0$

Any solution that meets these conditions is referred to as a Pareto stationary point [21, 79]. 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

$$\min_{\alpha^1,\ldots,\alpha^T} \quad \|\sum_{t=1}^{T} \alpha^t \nabla_{(\theta^{sh},\theta^t)} \mathscr{L}^t(\theta^{sh},\theta^t)\|_2^2, \qquad \text{s.t.} \quad \sum_{t=1}^{T} \alpha^t = 1, \alpha^t \geq 0 \quad \forall t. \tag{3.4}$$

Sener et al. [79] and Desideri et al.[21] 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.

3.2 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 Chapter 3.2.1. Second, we present an overview of our method,

RFU-SS, in Chapter 3.2.2. We then delve into RFU, addressing the forgetting objective for IB-trained models, in Chapter 3.2.3. Additionally, we introduce RFU-SS, which tackles the two-objective optimization unlearning problem, in Chapter 3.2.4.

3.2.1 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 $\mathscr{A}$ to learn the model $\mathscr{M}(\theta^r,\theta^a)$ 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 $\mathscr{U}$ to remove the trace of D_e while keeping the knowledge learned from the remaining dataset D_r. After unlearning, the unlearned model $\mathscr{M}_u(\theta^r,\theta^a)$'s distribution is expected to equal the distribution of the model that retrained solely based on the remaining dataset, i.e., $\mathscr{M}_u = \mathscr{U}(D,D_e,\mathscr{A}(D))$ is hoped to equal $\mathscr{M}_u = \mathscr{A}(D_r)$. 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

$$\begin{cases} p(Z|X_{-X_e}) \text{ unlearned by } \mathscr{M}_u = \mathscr{U}(D,D_e,\mathscr{A}(D)), \\ p(Z|X_r) \text{ retrained by } \mathscr{M}_u = \mathscr{A}(D_r), \\ p(Z|X_r) = p(Z|X_{-X_e}), \end{cases} \tag{3.5}$$

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 [28, 67] tried to solve the unlearning problem by minimizing the difference between the model unlearned based on the erased dataset $\mathscr{M}_u = \mathscr{U}(D,D_e,\mathscr{A}(D))$

and the model retrained without the erased dataset $\mathscr{M}_u = \mathscr{A}(D_r)$. 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 $\mathscr{M}_u = \mathscr{A}(D_r)$. Therefore, the certified-removal methods [36, 78] 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 [28, 67] 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 [67] 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

$$\textbf{Forgetting:} \qquad \min_{(\theta^r,\theta^a)} D_{KL}[p(Z|X_{-X_e})||p(Z|X_r)], \tag{3.6}$$

$$\textbf{Remembering:} \qquad \min_{(\theta^r,\theta^a)} \mathscr{L}(D_r;(\theta^r,\theta^a)), \tag{3.7}$$

The Equation (3.6) 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 Equation (3.7), the remembering objective, $\mathscr{L}(.)$ 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 $\mathscr{U}$ is retraining, the retraining process is to make the model approach to the retrained model, which is minimizing the first objective, Equation (3.6). 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.

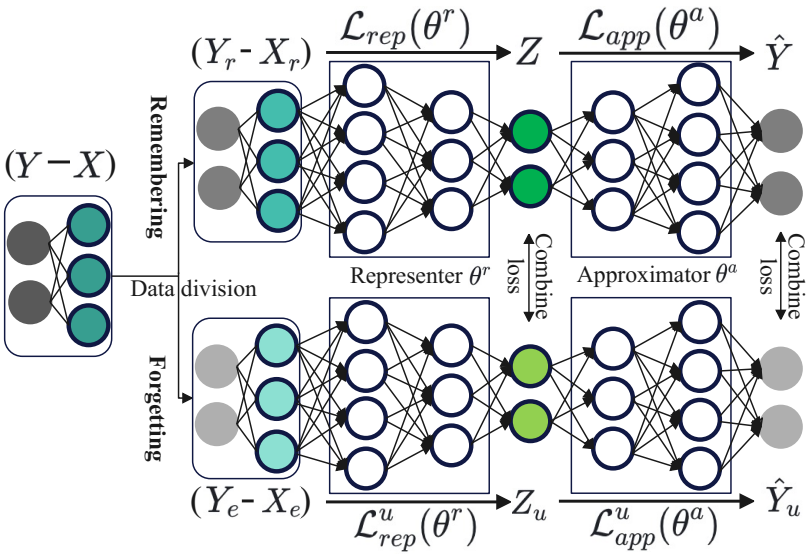


Figure 3.2: The RFU-SS Structure and Process.

3.2.2 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 3.2, 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 [52, 66, 78, 98] 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.

3.2.3 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 $\mathscr{M}(\theta^r,\theta^a)$ 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} [43] between the unlearned representation and originally trained representation before unlearning. Specifically, the unlearning loss for the representer can be described as

$$\mathscr{L}^u_{rep}(\beta_u)=\beta_u\cdot\underbrace{I_{\theta^r}(X_e;Z)}_{\text{Erasure term}}+\underbrace{D_{KL}[p_{\theta^r}(Z|X_{-X_e})||p_{\theta^r_{fix}}(Z|X_{-X_e})]}_{\text{Limitation term}}, \tag{3.8}$$

and the unlearning loss for approximator can be described as

$$\mathscr{L}^u_{app}(\beta_u)=\beta_u\cdot\underbrace{I_{\theta^a}(Y_e;Z)}_{\text{Erasure term}}+\underbrace{D_{KL}[p_{\theta^a}(Y_{-Y_e}|Z)||p_{\theta^a_{fix}}(Y_{-Y_e}|Z)]}_{\text{Limitation term}}, \tag{3.9}$$

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

$$\begin{aligned}\mathscr{L}^u &= \mathscr{L}^u_{rep}(\beta_u)+\mathscr{L}^u_{app}(\beta_u)\\ &= \beta_u\cdot I_{\theta^r}(X_e;Z)+D_{KL}[p_{\theta^r}(Z|X_{-X_e})||p_{\theta^r_{fix}}(Z|X_{-X_e})]\\ &+\beta_u\cdot I_{\theta^a}(Y_e;Z)+D_{KL}[p_{\theta^a}(Y_{-Y_e}|Z)||p_{\theta^a_{fix}}(Y_{-Y_e}|Z)].\end{aligned} \tag{3.10}$$

Minimizing the loss Equation (3.10) 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, $\mathscr{L}^r = I(X_r;Z) - I(Y_r;Z)$. Usually, the loss function can be optimized in a variational form like [1, 2, 5]. Alternatively, we can also use MINE [8] 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 Equation (3.2) in our Algorithm 1 later. Moreover, optimizing Equation (3.10) by grid search for a suitable unlearning rate β_u, RFU can already perform better than existing approximate unlearning methods.

3.2.4 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 Equation (3.1) on the remaining dataset (X_r, Y_r) to maintain the model accuracy. For clearness, we denote the corresponding loss in Equation (3.1) on the remaining dataset (X_r, Y_r) as $\mathscr{L}^r_{rep}$ and $\mathscr{L}^r_{app}$. In particular, we can optimize them as multi-task learning in Equation (3.3). The transformed minimization for the two-objective unlearning can be described as the following formulation:

$$\min_{(\theta^r,\theta^a)} \alpha^u \cdot (\mathscr{L}^u_{rep}(\beta_u) + \mathscr{L}^u_{app}(\beta_u)) + \alpha^r \cdot (\mathscr{L}^r_{rep} + \mathscr{L}^r_{app}), \tag{3.11}$$

where α^u and α^r are static or dynamically computed weights for forgetting and remembering tasks. We should notice that minimizing the unlearning loss to unlearn the information of $D_e = (X_e, Y_e)$ from the trained IB model can be proved equivalent to minimizing the retraining IB model's loss. Unlearning is to approximate a model approaching to retrained model, which is consistent with retraining. Hence, we can design multi-objective optimization to solve Equation (3.11).

It is difficult and time-consuming to achieve the optimal settings of Equation (3.11) if we use heuristic algorithms or grid search over various scalings [44, 79]. 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* $\mathscr{L}^u(\theta^r, \theta^a) \leq \mathscr{L}^u(\bar{\theta}^r, \bar{\theta}^a)$ *and* $\mathscr{L}^r(\theta^r, \theta^a) \leq \mathscr{L}^r(\bar{\theta}^r, \bar{\theta}^a)$ *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 ($\mathscr{P}_{(\theta^r, \theta^a)}$). Usually, we can achieve the local optimality of multi-objective optimization problems via gradient descent [21, 25, 72, 73, 77], 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*

$$\begin{cases} \alpha^u \nabla_{(\theta^r, \theta^a)}(\mathscr{L}^u(\theta^r, \theta^a)) + (1 - \alpha^u)\nabla_{(\theta^r, \theta^a)}(\mathscr{L}^r(\theta^r, \theta^a)) = 0, \\ \alpha^u \in [0, 1]. \end{cases} \tag{3.12}$$

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,

$$\min_{\alpha^u} ||\alpha^u \nabla_{(\theta^r, \theta^a)}(\mathscr{L}^u) + (1 - \alpha^u)\nabla_{(\theta^r, \theta^a)}(\mathscr{L}^r)||_2^2, \quad \text{s.t. } \alpha^u \in [0, 1]. \tag{3.13}$$

According to [21, 79], 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 Equation (3.13) 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:

$$\hat{\alpha^u} = \left[\frac{(\nabla_{(\theta^r, \theta^a)}(\mathscr{L}^r) - \nabla_{(\theta^r, \theta^a)}(\mathscr{L}^u))^\top (\nabla_{(\theta^r, \theta^a)}(\mathscr{L}^r))}{||\nabla_{(\theta^r, \theta^a)}(\mathscr{L}^u) - \nabla_{(\theta^r, \theta^a)}(\mathscr{L}^r)||_2^2} \right]_{+, \frac{1}{\top}}, \tag{3.14}$$

where $[\cdot]_{+,\frac{1}{\tau}}$ denotes clipping to $[0,1]$ as $[a]_{+,\frac{1}{\tau}} = \max(\min(a,1),0)$. The value of $\hat{\alpha^u}$ is highly related to the numerator, the dot product between $\nabla_{(\theta^r,\theta^a)}(\mathscr{L}^r) - \nabla_{(\theta^r,\theta^a)}(\mathscr{L}^u)$ and $\nabla_{(\theta^r,\theta^a)}(\mathscr{L}^r)$. If the dot product is a large positive value, it implies that the difference between the two gradient vectors aligned with the gradients of $\mathscr{L}^r$, the loss of remembering. It means that the changes in $\mathscr{L}^r$ and $\mathscr{L}^u$ are generally in the same direction, and thus $\mathscr{L}^u$ might serve as a good approximation for $\mathscr{L}^r$ 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 $\nabla_{(\theta^r,\theta^a)}(\mathscr{L}^r)$, meaning that changes in $\mathscr{L}^r$ and $\mathscr{L}^u$ 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 [21]. 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 $\mathscr{M}_{fix}(\theta^r_{fix}, \theta^a_{fix})$ using $\mathscr{M}(\theta^r,\theta^a)$ 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. 3.8 to 3.10, and the corresponding calculations are shown from lines 6 to 9. All these equations are extended and expressed in a variational empirical form like Equation (3.2). Finally, we update the model by combining the two optimization objectives together according to Equation (3.14), 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 $\mathcal{M}_u(\theta^r_u, \theta^a_u)$

Set the fixed temp model:
$\mathcal{M}_{fix}(\theta^r_{fix}, \theta^a_{fix}) \leftarrow \mathcal{M}(\theta^r, \theta^a)$
for *E epochs* **do**
  Sample minibatch of m data points $\{(x_i, y_i)\}_{i=1}^m$ from the erased dataset D_e;
  Sample minibatch of m data points $\{(x_j, y_j)\}_{j=1}^m$ from the remaining dataset D_r;
  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;
  Calculate the unlearned representation loss function Equation (3.8) in a per-sample form for each minibatch of size m as
  $\mathscr{L}^u_{rep}(\beta_u) = \beta_u \cdot \frac{1}{m}\sum_{i=1}^m D_{KL}[p_{\theta^r}(z|x_i)||\prod_i^{|z|} q_i(z_i)]$
  $+\frac{1}{m}\sum_{i=1}^m D_{KL}[p_{\theta^r_{fix}}(z|x_i)||p_{\theta^r}(z|x_i)]$
  Calculate the unlearned approximation loss function Equation (3.9) in a per-sample form for each minibatch of size m as
  $\mathscr{L}^u_{app}(\beta_u) = \beta_u \cdot \frac{1}{m}\sum_{i=1}^m \log\ p_{\theta^a}(y_i|z_i)$
  $+\frac{1}{m}\sum_{i=1}^m D_{KL}[p_{\theta^a_{fix}}(y_i|z_i)||p_{\theta^a}(y_i|z_i)]$
  Calculate the representation and approximation loss based on the remaining data as Equation (3.2) $\mathscr{L}^r_{app} + \mathscr{L}^r_{rep} = -\frac{1}{m}\sum_{j=1}^m \log\ p_{\theta^a}(y_j|z_j)$
  $+\frac{\beta}{m}\sum_{j=1}^m D_{KL}[p_{\theta^r}(z_j|x_j)||q_{\theta^r}(z_j)]$
  Calculate $\hat{\alpha^u}$ using Equation (3.14);
  We update the model (both the representer and approximator) according to the gradients of the combined loss functions Equation (3.13) as
  $(\theta^r, \theta^a) \leftarrow (\theta^r, \theta^a) - \eta\nabla_{(\theta^r,\theta^a)}(\hat{\alpha^u}\cdot(\mathscr{L}^u_{rep}(\beta_u) + \mathscr{L}^u_{app}(\beta_u)) + (1-\hat{\alpha^u})\cdot(\mathscr{L}^r_{rep} + \mathscr{L}^r_{app}))$
Return $\mathcal{M}_u(\theta^r_u, \theta^a_u) \leftarrow \mathcal{M}_u(\theta^r, \theta^a)$;

3.3 Performance Evaluation

In this section, we introduce the detailed evaluation of RFU-SS and comparisons between RFU-SS and the state-of-art methods [28, 36, 67, 78].

3.3.1 Experiment Setup

Our experiments are conducted on two representative datasets, MNIST and CIFAR10. These two datasets are commonly chosen in ML studies and cover varying levels of learning complexity. We evaluate RFU-SS and compare it with state-of-the-art unlearning methods in the traditional centralized setting. There are two representative approx-

Table 3.2: General Evaluation Results on MNIST and CIFAR10 of Chapter 3.

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%

imate unlearning methods, Hessian-matrix-based unlearning (HBU) [36, 78] and variational Bayesian unlearning (VBU) [28, 67]. To effectively evaluate unlearning methods, we refer to a common method [40] 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 [40] 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.

3.3.2 Overall Evaluation

We first demonstrate the overall evaluation results of different methods on MNIST and CIFAR10, shown in Table 3.2. 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.

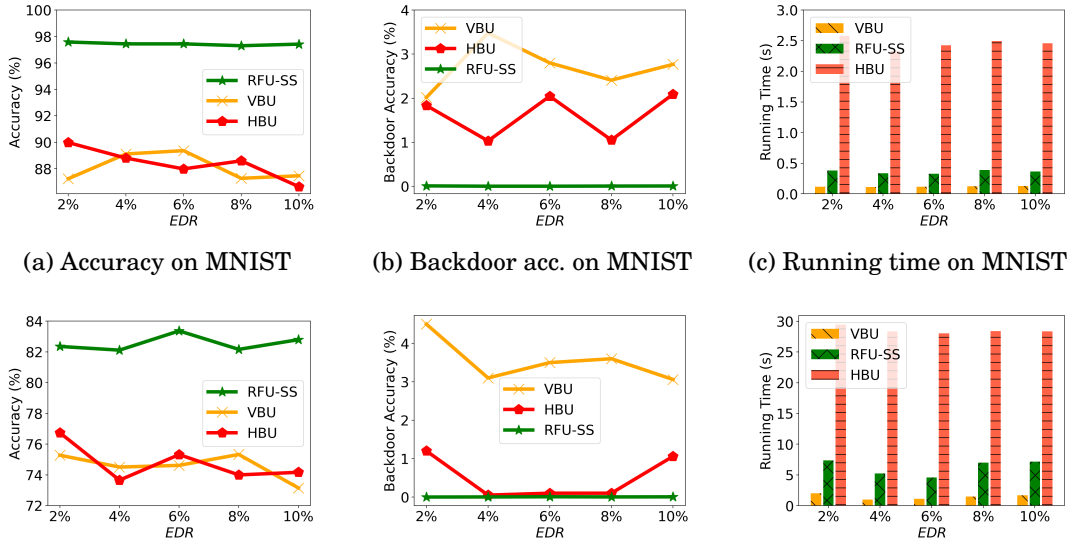
3.3.2.1 Evaluation of Efficiency

From the running time shown in Table 3.2, 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.

3.3.2.2 Evaluation of Effectiveness

In Table 3.2, 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.



(a) Accuracy on MNIST (b) Backdoor acc. on MNIST (c) Running time on MNIST

(d) Accuracy on CIFAR10 (e) Backdoor acc. on CIFAR10 (f) Running time on CIFAR10

Figure 3.3: The running time, acc. on the test dataset, and backdoor acc. on the erased dataset of various EDR

3.3.3 Evaluation of the Influence of Different Variables

3.3.3.1 Impact of the Erased Data Ratio

In this chapter, 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 3.3.

Our effectiveness evaluations include accuracy on the test dataset (Figures (a) and (d)) and backdoor accuracy on the erased dataset (Figures (b) and (e)). The accuracy of unlearned models on the test dataset and backdoored samples reflects the utility preservation and the forgetting effectiveness 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 (a) and (d), as EDR increases, the accuracy of all unlearned models decreases slightly. In this comparison, we focus

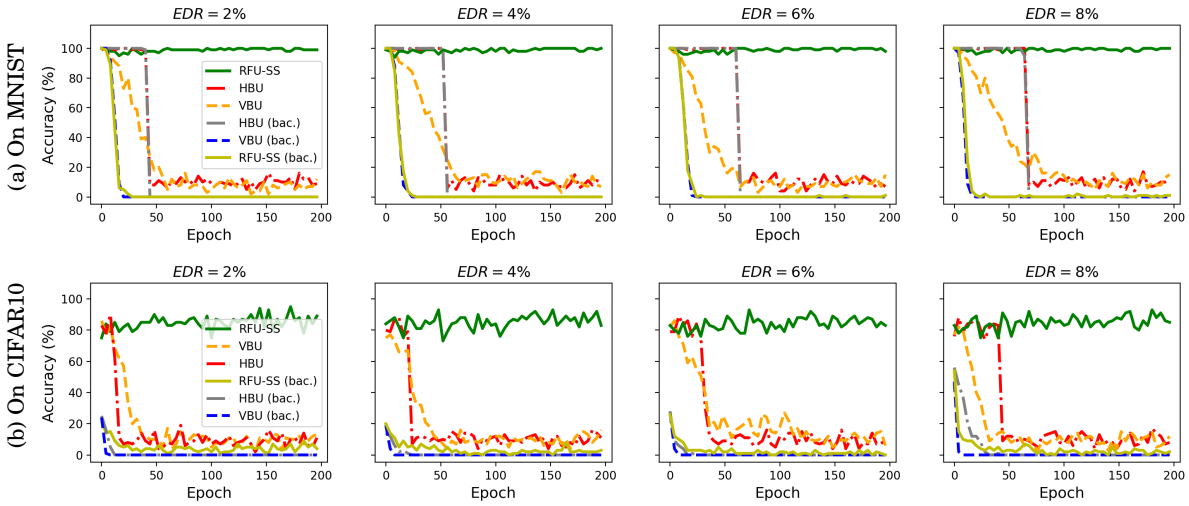


Figure 3.4: The changes of accuracy and backdoor accuracy on various EDR during training on MNIST and CIFAR10

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 (b) and (e), 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 (c) and (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 3.4 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,

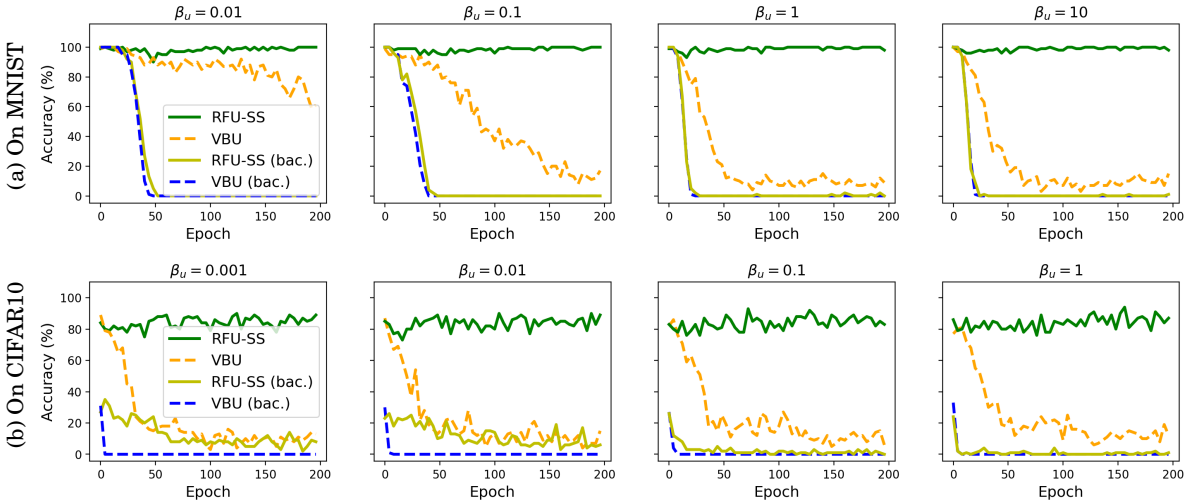


Figure 3.5: The changes of accuracy and backdoor accuracy on various β_u during training on MNIST and CIFAR10

and it takes more time for all unlearning methods to remove the influence of backdoors.

In Figure (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 (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.

3.3.3.2 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 3.5.

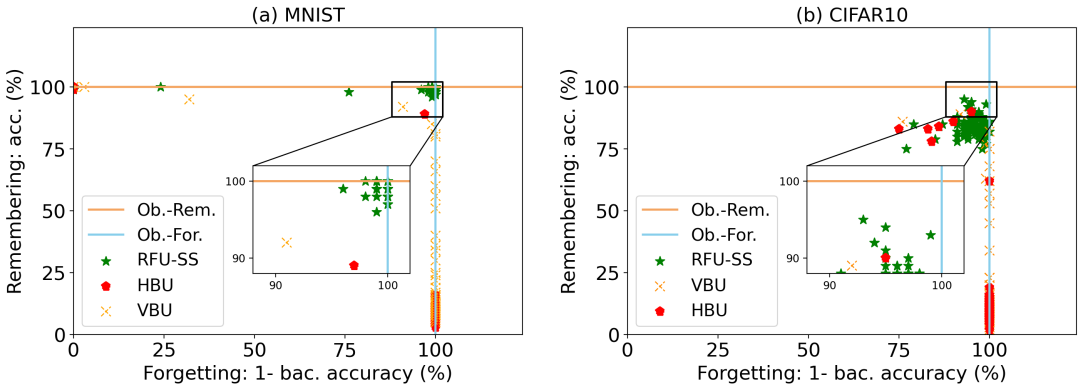


Figure 3.6: Results of two-objective optimization unlearning on MNIST and CIFAR10.

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 (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 (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 $\beta_u = 1$.

3.3.4 Optimization Results of Unlearning

Figure 3.6 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 3.6, RFU-SS can achieve the best remembering and 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 3.6(b) also shows similar optimization results on CIFAR10,

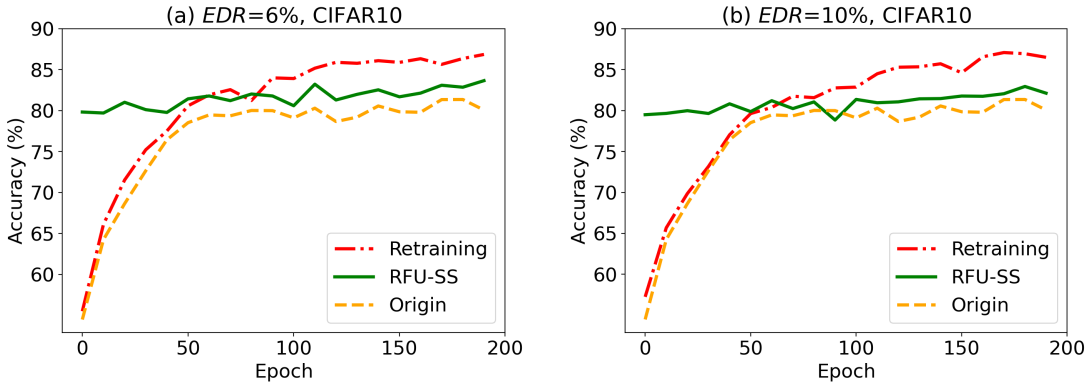


Figure 3.7: RFU-SS recovers the model knowledge harmed by backdoors on CIFAR10

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.

3.3.5 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%. Learning these backdoored samples is an extremely abnormal scenario in the real world. In Figure 3.7, 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 3.7 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

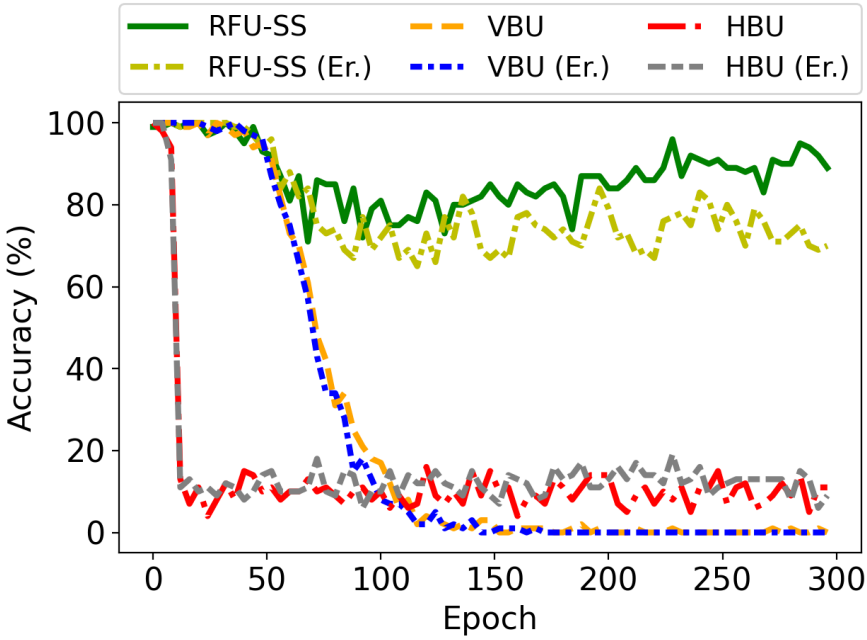


Figure 3.8: Training of unlearning independent and identical distribution (IID) normal data.

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.

3.3.6 Additional Evaluations on Normal Data Erasure

We additionally evaluate HBU, VBU and RFU-SS in unlearning the IID normal data, as shown in Figure 3.8. 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.

3.4 Summary

In this chapter, 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.

CHAPTER 4

FEDU: FEDERATED UNLEARNING VIA USER-SIDE INFLUENCE APPROXIMATION FORGETTING

Machine unlearning has become a significant research topic on a global scale due to the increasing importance of privacy protection, particularly in light of the right to be forgotten legislation. Although many solutions are proposed, the current mainstream centralized machine unlearning studies are not feasible in federated learning (FL), where the server has no access to any users' unlearning samples. In this chapter, we aim to tackle the *federated unlearning* problem by proposing a Federated Unlearning (FedU) scheme via a user-side influence approximation forgetting method, thereby eliminating the need to share raw data with the server. The key challenge is how can use machine unlearning to help protect the privacy of the unlearning users who hope to remove their personal data and meanwhile ensure the safety of the rest users who hope to retain their data contribution in the model. In FedU, only users who have unlearning needs execute the influence approximation forgetting, while other users and the server just conduct the same operations as they did in FL. The proposed influence approximation forgetting method achieves unlearning by estimating the influence of the erased samples relying on only the user's local data and eliminating this influence from the model. However, the model utility is still negatively influenced by directly removing the influence estimation. To mitigate the side effects of unlearning, we propose a utility preservation method that simultaneously trains the unlearned model based on the unlearning requesters' remaining local dataset. We design an adaptive

Table 4.1: Basic Notations of Chapter 4

Notations	Descriptions
D	the entire FL training dataset D encompasses all local dataset D_k, represented as $D = \{D_1, D_2, ..., D_K\}$
C_{k_c}	the normal user
C_{k_u}	the unlearned user
$D^e_{k_u}$	the local erased dataset of the unlearned user C_{k_u}
$D^r_{k_u}$	the local remaining dataset of the unlearned user C_{k_u}
$z : (x, y)$	the persample z with the input x and label y
$\mathscr{M}(\theta)$	the ML model $\mathscr{M}$ with the parameters θ
$\mathscr{M}_u$	the unlearned model
$\mathscr{L}_{IAF}$	the influence approximation forgetting loss
$\mathscr{L}_{UP}$	the utility preservation loss

optimization method to balance the forgetting and utility preservation effectiveness optimally during the unlearning process. Extensive evaluations on three representative public datasets demonstrate that our proposed method significantly outperforms state-of-the-art methods in both effectiveness and efficiency, avoiding more than 3% accuracy degradation when the number of unlearning requesters is large.

4.1 Preliminary and Problem Statement

4.1.1 Notations

Initially, we provide an overview of the key notations utilized in this chapter, as detailed in Table 4.1. In the FL scenarios, we use D to denote the full global training dataset that includes all the users' local datasets. We represent it as $D = \{D_1, D_2, ..., D_K\}$. In all K FL users, we denote the "normal user" as C_{k_c} and the "unlearned user" as C_{k_u}. For the unlearned user C_{k_u}, he/she can divide their local dataset D_{k_u} into the local erased dataset $D^e_{k_u}$ and the local remaining dataset $D^r_{k_u}$. During our experiments and training process, we denote the persample as $z : (x, y)$, which includes the feature vector x and its associated label y. The objective of the FL server is to train an ML model $\mathscr{M}$, parameterized by θ. We denote the unlearned model as $\mathscr{M}_u$. In our method, we have an influence approximation forgetting loss function $\mathscr{L}_{IAF}$ and the utility preservation loss function $\mathscr{L}_{UP}$.

4.1.2 Influence Function

Machine learning aims to identify a model $\mathscr{M}$, parameterized by θ, that establishes a mapping from an input feature space $\mathscr{X}$ to an output space $\mathscr{Y}$, denoted as $\mathscr{M}(\mathscr{X},\theta) \to \mathscr{Y}$. The training dataset is represented as $D = \{z_i : (x_i, y_i)\}_{i=1}^n$, where z denotes the training sample with input x and label y. The corresponding loss function is denoted as $\ell(z,\theta)$. The standard empirical risk minimization (ERM) of a machine learning model can be represented by the following optimization problem:

$$\theta^* = \arg\min_\theta \ \ell(D,\theta) = \arg\min_\theta \frac{1}{n} \sum_{z_i \in D} \ell(z_i,\theta). \tag{4.1}$$

If a training subset $D^e \subset D$ contains m samples and is up-weighted by an infinitesimal amount ϵ, it results in a removal of the subset of model parameters, which can be denoted as:

$$\theta^\epsilon_{D\setminus D^e} = \arg\min_\theta \frac{1}{n} \sum_{z_i \in D} \ell(z_i,\theta) - \epsilon \frac{1}{m} \sum_{z_j \in D^e} \ell(z_j,\theta). \tag{4.2}$$

The changes in the optimal model parameters can be expanded using the perturbation theory [3] as:

$$\Delta\theta = \theta^\epsilon_{D\setminus D^e} - \theta^* = \mathscr{O}(\epsilon)\theta^{(1)} + \mathscr{O}(\epsilon^2)\theta^{(2)} + \mathscr{O}(\epsilon^3)\theta^{(3)} + \cdots, \tag{4.3}$$

where each removing sample in D^e is up-weighted by a factor of ϵ. $\theta^{(1)}$ denotes the first-order (in ϵ) perturbation and $\theta^{(2)}$ is the second-order model perturbation.

According to existing studies [6, 47], the first-order influence approximation can be expressed as

$$\Delta\theta \simeq \mathscr{I}^{(1)}(D^e) = \frac{1}{n-m} H^{-1}_{\theta^*} \sum_{z_j \in D^e} \ell(z_j,\theta^*), \tag{4.4}$$

where $H^{-1}_{\theta^*}$ represents the reverse of the Hessian matrix with respect to the model parameters θ^*. Existing studies [6, 47] also discussed the second-order influence function. In this chapter, we focus on the first-order influence approximation, which is calculation-efficient and effective for unlearning.

4.1.3 Problem Definition of Federated Unlearning

We briefly introduce the problem definition of federated unlearning as follows. We assume there are K FL users, each with a local training dataset, denoted as $D_k = (X_k, Y_k)$,

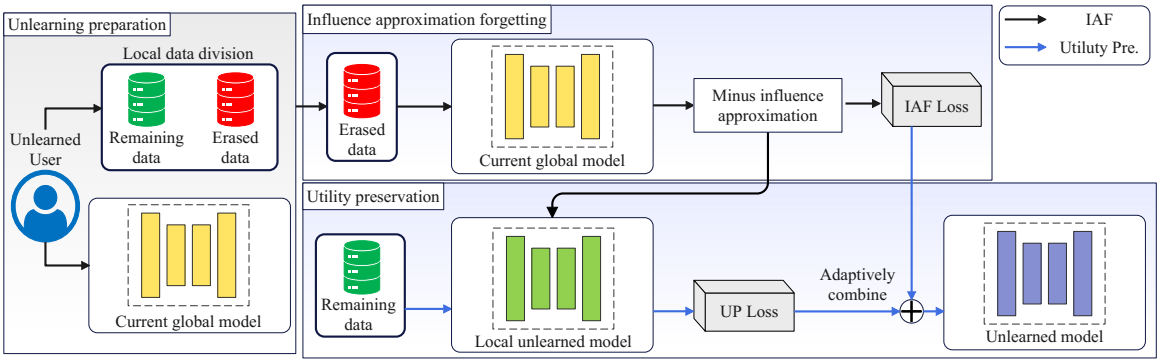


Figure 4.1: Overview of FedU via user-side influence approximation forgetting.

where X_k and Y_k are corresponding to the input and output. $D = \{D_1, D_2, ..., D_K\}$ is the whole training dataset from all users. The unlearned user C_{k_u} divides his local dataset to $D^e_{k_u} = (X^e_{k_u}, Y^e_{k_u})$ as the erased data and $D^r_{k_u} = (X^r_{k_u}, Y^r_{k_u})$ as the remaining local data. Removing the erased dataset $D^e_{k_u}$, the remaining global dataset distribution can be denoted as $D' = \{D_1, D_2, .., D^r_{k_u}, ..., D_K\}$.

We assume that the FL server already trained the global model $\mathscr{M}(\theta)$ leveraging the model contributions from all users $\mathscr{C}$. Users' local models are trained based on their local dataset D_k. Now, we need to design a specified unlearning algorithm $\mathscr{U}$ for the unlearned user $C_{k_u} \subseteq \mathscr{C}$ who wants to revoke the influence of his/her erased data $D^e_{k_u}$ from the global FL model. Ideally, the distribution of the unlearned model $\mathscr{M}_u(\theta_u)$ should match the distribution of the FL model that is retrained on the residual global dataset D' using the original FL algorithm. Therefore, the federated unlearning formulation can be described as

$$P(\mathscr{M}_u = \mathscr{F}(D') \in \mathscr{Q}) = P(\mathscr{M}_u = \mathscr{U}(D, D^e_{k_u}, \mathscr{F}(D)) \in \mathscr{Q}), \tag{4.5}$$

where $\mathscr{Q}$ represents the distribution space of hypothesis parameters for the model trained using the federated algorithm $\mathscr{F}$. Aligning the unlearned FL model with the retrained model involves two critical dimensions: data erasure and model utility preservation. First, data erasure must be implemented, ensuring that the information from the erased data is deleted from the model. The unlearned model should resemble the retrained model as if it had never observed the erased data. Second, the model's performance must be maintained to match that of the retrained model based on the remaining dataset. Current federated unlearning solutions [39, 58] have predominantly focused on the data erasure aspect, often neglecting the utility preservation aspect.

4.2 FedU via User-Side Influence Approximation Forgetting

4.2.1 Overview of FedU

We demonstrate the main procedure at the unlearned user of FedU in Figure 4.1. Any unlearned user can execute this method locally to remove his specified samples from the global model. One important component of FedU via user-side influence approximation forgetting is the influence approximation forgetting (IAF) method, which implements data erasure by subtracting the estimated influence of the specified-erased samples from the trained FL model. FedU with user-side IAF can already implement unlearning and achieve similar effects as current unlearning methods [58], but it still causes model utility degradation. To reduce the model utility reduction resulting from unlearning, we design a utility preservation method, which simultaneously trains the unlearned model using the remaining local dataset with the original FL algorithm. Moreover, we combine these two components and propose an adaptive optimization method to find the best balance between unlearning effectiveness and model utility preservation. We named the FedU with the utility preservation method as FedU-U.

4.2.2 Influence Approximation Forgetting

Our method is naturally integrated into common FL frameworks with the goal of minimizing the empirical risk of the FL model on the full training set $D = \{D_1, D_2, ..., D_K\}$. Assuming the unlearned user C_{k_u} and his/her local erased samples $D^e_{k_u} = (X_{k_u}, Y_{k_u})$, the first-order influence approximation is $\mathscr{I}^{(1)}(D^e_{k_u})$ in Equation (4.4). We just need to minus the influence estimation from the trained model $\mathscr{M}_{\theta^*}$ with parameters θ^* to implement unlearning, which is also applied in lots of existing Hessian matrix-based unlearning methods [36, 98, 100]. However, these methods rely on access to the entire remaining training dataset, which is unavailable to the unlearning user in the FL scenario.

Inspired by [6, 47], we propose an influence approximation forgetting method, which overleaps calculating estimation for the Hessian matrix $H^{-1}_{\theta^*}$ but directly estimates the entire forgetting influence $H^{-1}_{\theta^*} \sum_{z_j \in D^e_{k_u}} \ell(z_j, \theta^*)$ based on Hessian-vector products (HVPs) employing several sampled local remaining samples. It requires only the assistance of the erased samples and a few local training samples of the unlearning user, ensuring that the user can calculate it efficiently. We use $\nabla^2_\theta \ell(z_i, \theta^*)$ of any $z_i \in D^r_{k_u}$ as an unbiased estimator of the Hessian matrix H. We can uniformly sample

t points $z_1, z_2, ..., z_t$ from the local remaining dataset $D^r_{k_u}$; define the gradient vector $g_u = \nabla\ell(z_u, \theta^*)$ and the initial Hessian-vector products as $\tilde{H}_0^{-1} g_u = g_u$. Then, we can recursively compute $\tilde{H}_i^{-1} g_u = g_u + (I - \nabla^2_\theta \ell(z_i, \theta^*))\tilde{H}_{i-1}^{-1} g_u$. $\tilde{H}_t^{-1} g_u$ is the final unbiased estimate of the influence $H^{-1} g_u$ for forgetting. For stability, we can pick t to be large enough and we can repeat above process r times and average results to reduce variance. All these computations are easy to implement in auto-grad methods. And the approximate optimization can be represented as

$$\bar{\theta}_{D \setminus D^e_{k_u}} = \theta^* + \frac{1}{n-m} \tilde{H}^{-1}_{(\theta^*, t, D^r_{k_u})} \sum_{z_j \in D^e_{k_u}} \nabla_\theta \ell(z_j, \theta^*), \tag{4.6}$$

where $\tilde{H}^{-1}_{(\theta^*, t, D^r_{k_u})}$ is the inverse of the estimated Hessian matrix calculated using the estimator based on t sampled data points of $D^r_{k_u}$, and the $\nabla_\theta \ell(z_j, \theta^*)$ term can be calculated directly based on the erased samples. The loss function for influence approximation forgetting still keeps the ERM function as the main part, which can be represented as follows:

$$\text{IAF Loss:} \quad \mathscr{L}_{\text{IAF}} = \arg\min_{\theta^*} \frac{1}{n-m} \sum_{z \in D \setminus D^e_{k_u}} \ell(z, \theta). \tag{4.7}$$

When calculating the loss using Equation (4.7) and update the unlearned model according to Equation (4.6), in a strongly convex setting, we can achieve the unlearning update bound $\|\theta_u - \bar{\theta}_{D \setminus D^e_{k_u}}\| \leq \frac{2ML^2m^2}{\lambda^3 n^2}$. We present the analysis of this bound in Chapter 4.3.1. From the deviation between our influence approximate forgetting $\bar{\theta}_{D \setminus D^e_{k_u}}$ and the empirical unlearned minimizer θ_u, this bound indicates that ensuring the size m of the erased data $D^e_{k_u}$ is relatively small compared to the size n of D in Equation (6.3) will help in maintaining a small bound for model utility preservation.

4.2.3 Utility Preservation and an Adaptive Optimization Method

Although the influence approximation forgetting based on HVPs speeds up the unlearning update, the estimated influence approximation makes it hard to capture all the information, which leads to utility degradation. Therefore, we designed a utility preservation method to compensate for the utility degradation caused by the IAF method. We show this component in Figure 4.1 as the blue arrows. In this component, we simultaneously train the unlearned local model with the remaining local dataset $D^r_{k_u} = (X^r_{k_u}, Y^r_{k_u})$ using the original FL optimization loss according to Equation (4.1). We denote the utility preservation (UP) loss as $\mathscr{L}_{\text{UP}} = \ell(D^r_{k_u}, \theta)$. Since the trained FL model has already

learned the knowledge of the remaining data from the previous FL training, we only need to sample several data points for utility preservation optimization. Inspired by multi-task learning [54, 79], we combine the two losses and formalize them as the following two tasks optimization problem from the unlearned user C_{k_u}'s perspective

$$\min_{\theta_{k_u}} (\mathscr{L}_{\text{IAF}}(\theta_{k_u}), \mathscr{L}_{\text{UP}}(\theta_{k_u}))^T. \tag{4.8}$$

We aim to optimize the IAF loss and UP loss simultaneously based on the same model structure but different sub-datasets, $D^e_{k_u}$ and $D^r_{k_u}$. We can solve this problem according to [54, 79] to find the Pareto optimality of the two losses, but with much additional computation. Here, similar to the method in [25], we introduce a simple solution to find the steepest descent algorithm of the two-task optimization problem. The update rule is $\theta_{k_u,t+1} = \theta_{k_u,t} + \eta d_t$, where η is the learning rate and the updating direction $d_{k_u,t}$ is determined by:

$$(d_{k_u,t}, \alpha_{k_u,t}) = \arg \min_{d \in R^n, \alpha \in R} \alpha + \frac{1}{2}\|d\|^2, \quad \text{s.t.} \quad \nabla \mathscr{L}_{\text{IAF}}(\theta_{k_u,t})^T d \leq \alpha, \nabla \mathscr{L}_{\text{UP}}(\theta_{k_u,t})^T d \leq \alpha. \tag{4.9}$$

We show the proof that the solutions of the Equation (4.9) can find optimality or will update the model along the steepest direction in Chapter 4.3. However, directly solving the optimization problem of Equation (4.9) would be significantly slow, particularly in models of high dimensionality exceeding millions of parameters. We can reformulate the Equation (4.9) using the Karush-Kuhn-Tucker (KKT) conditions [31] to address the constrained two-task optimization challenge as

$$d_{k_u,t} = -(\lambda \nabla \mathscr{L}_{\text{IAF}}(\theta_{k_u,t}) + \beta \nabla \mathscr{L}_{\text{UP}}(\theta_{k_u,t})), \quad \text{where} \quad \lambda + \beta = 1, \tag{4.10}$$

$\lambda \geq 0$ and $\beta \geq 0$ serve as the Lagrange multipliers associated with the linear inequality constraints. The unlearned user C_{k_u} can locally execute federated unlearning based on his erased data $D^e_{k_u}$ and remaining local data $D^r_{k_u}$ using Equation (4.10) to optimized both objectives of Equation (4.8). Then, C_{k_u} uploads the unlearned local model $\mathscr{M}_{k_u}(\theta_{k_u})$ to FL server for updating the global unlearning model. The FL server collects all users' models, including the normal and unlearned models, to update the global model similar to the traditional FL aggregation as

$$\mathscr{M}_u(\theta) = \frac{1}{K} \sum_{k=1}^{K} \mathscr{M}_k(\theta_k), \tag{4.11}$$

where k is the index of unlearned users, k_u, or the index of normal users, k_c, and K is the number of total users.

Algorithm 2: FedU via User-Side Influence Approximation Forgetting

Input: K is the number of users; E is the number of local epochs, η is the learning rate.

1 **FL server executes:**
2 Reinitialize the global model $\mathscr{M}_u^{t=0}(\theta) \leftarrow \mathscr{M}(\theta)$, and send it to all users;
3 **User executes:**
4 **for** *all users in parallel* **do**
5 **for** $t = 0, 1, ..., T$ **do**
6 **if** *is unlearned user* **then**
7 IAF-U($k_u, \mathscr{M}_u^t(\theta), E, D_{k_u}^e, D_{k_u}^r$);
8 Uploads $\mathscr{M}_{k_u}^t$ to server and recieves $\mathscr{M}_u^{t+1}$;
9 **if** Verification($\mathscr{M}_u^{t+1}, D_{k_u}^e$)==Yes **then** change to normal user;
10 **else**
11 NormalTraining($k_c, \mathscr{M}_u^t(\theta), E, D_{k_c}$);
12 Uploads $\mathscr{M}_{k_c}^t$ to server and recieves $\mathscr{M}_u^{t+1}$;
13 **IAF-U**($k_u, \mathscr{M}_u^t(\theta), D_{k_u}^e, D_{k_u}^r$): ▷ Run on the unlearned user C_{k_u}
14 **for** E *epochs* **do**
15 Smaple a minibatch, $\{z_i : (x_j, y_j)\}_{j=1}^m$, from erased dataset $D_{k_u}^e$;
16 Sample a minibatch, $\{z_j : (x_i, y_i)\}_{i=1}^m$, from remaining dataset $D_{k_u}^r$;
17 Calculate the influence approximation forgetting loss $\mathscr{L}_{\text{IAF}}$ according to Equation (4.7);
18 Calculate the utility preservation loss $\mathscr{L}_{\text{UP}}$ according to Equation (4.1);
19 Calculate d using Equation (4.10) based on $\mathscr{L}_{\text{IAF}}$ and $\mathscr{L}_{\text{UP}}$, and update the model as: $\theta \leftarrow \theta + \eta d$
20 **NormalTraining**($k_c, \mathscr{M}_u^t(\theta), D_{k_c}$): ▷ Run on normal FL user C_{k_c}
21 **for** E *epochs* **do**
22 Sample minibatch $\{z_i : (x_i, y_i)\}_{i=1}^m$ from local D_{k_c};
23 Calculate the training loss according to Equation (4.1);
24 Update: $\theta \leftarrow \theta - \eta \nabla_\theta \ell(\cdot, \theta)$
25 **FL server executes:**
26 Update unlearned global model, aggregating local models from all users as Equation (4.11): $\mathscr{M}_u^{t+1}(\theta) \leftarrow \frac{1}{K} \sum_{k=1}^K \mathscr{M}_k^t(\theta_k)$;
27 Return $\mathscr{M}_u(\theta)$;

4.2.4 Pseudocode of the Full Version FedU Algorithm

Assume the server has trained a FL model $\mathscr{M}(\theta)$, which is then used as the initial model for unlearning, denoted $\mathscr{M}_u^0(\theta)$. FL server sends the model to users to continue the FL training (including unlearning and learning), shown in Line 2 of Algorithm 2. Users who want to unlearn their local samples execute influence approximation forgetting with utility preservation method (IAF-U) (Lines 13 to 19) based on the erased dataset

$D^e_{k_u}$, and the sampled same size sub-dataset from $D^r_{k_u}$. For ease of simultaneously optimizing the data forgetting and model utility, we randomly select the subset from $D^r_{k_u}$ of the same size as $D^e_{k_u}$. During the unlearning process, these unlearned users use a Verification function to evaluate if the unlearned model forgets the erased data. We use a simple backdoor inference method as the Verification function, which tests if the model identifies the backdoored-erased samples with lower accuracy than randomly selecting. Suppose the unlearned model passes the verification testing. In that case, these unlearning users will stop unlearning and change their operation to normal FL users, training models only based on the remaining local dataset. For those normal users, they train the model using NormalTraining (Lines 20 to 24) as a regular FL process. When receiving all users' models of one iteration, the FL server updates the unlearned global model of this iteration using FL aggregation Equation (4.11) as Line 26 and then sends the updated model for the next round of training.

4.3 Theoretical Analysis

The Theoretical analysis includes three main goals. First, we analyze the bound for the unlearning update using our IAF method. Second, we aim to demonstrate that the solutions obtained from Equation (4.9) for the two objectives identify the steepest gradient descent direction, thereby achieving optimality. Third, we will analyze the complexity of our FedU.

4.3.1 Bound of our IAF Update

Our FedU achieves unlearning based on the IAF method, which is updated using Equation (4.6). In a strongly convex setting, we can achieve the unlearning update bound as follows:

Theorem 1. *For any $z \in \mathcal{Z}$, the function $\ell(z,\theta)$ is λ-strong convex, L-Lipschitz and M-Hessian Lipschitz with respect to θ. Let $D \sim \mathcal{D}^n$ be a set of n samples, and $D^e_{k_u} \subseteq D$ denote the set of m delete requests for user k_u. Define the model point θ_u as the empirical minimizer over $D \backslash D^e_{k_u}$, i.e., $\theta_u \in \arg\min_\theta \sum_{z \in D \backslash D^e_{k_u}} \frac{1}{n-m} \ell(z,\theta)$. Then,*

$$\|\theta_u - \bar{\theta}_{D \backslash D^e_{k_u}}\| \leq \frac{2ML^2m^2}{\lambda^3 n^2}, \tag{4.12}$$

where the point $\bar{\theta}_{D \backslash D^e_{k_u}}$ is achieved using Equation (4.6) *with current optimal θ^*.*

We prove Theorem 1 based on the following Lemma 1.

Lemma 1. *The original optimized model θ^* trained on D, and θ_u defined in Theorem 1, satisfy the guarantee $||\theta^* - \theta_u|| \leq \frac{2mL}{\lambda n}$.*

Lemma 1 is easy to derived based on the L-Lipschitz definitions in Theorem 1. Then, based on Lemma 1, we can prove Theorem 1 as follows.

Proof. Define the functions $\hat{\mathscr{L}}_1$ and $\hat{\mathscr{L}}_2$ as

$$\hat{\mathscr{L}}_1(\theta) := \frac{1}{n}\sum_{z\in D}\ell(z,\theta), \hat{\mathscr{L}}_2(\theta) := \frac{1}{n-m}\sum_{z\in D\setminus D^e}\ell(z,\theta). \tag{4.13}$$

Using the Taylor's expansion for $\nabla\hat{\mathscr{L}}_2(\theta_u)$ around the point θ^*, we get that

$$\|\nabla\hat{\mathscr{L}}_2(\theta_u) - \nabla\hat{\mathscr{L}}_2(\theta^*) - \nabla^2\hat{\mathscr{L}}_2(\theta^*)[\theta_u - \theta^*]\| \leq \frac{M}{2}\|\theta^* - \theta_u\|^2, \tag{4.14}$$

where M denotes the Hessian-Lipschitz constant for the function $\ell(z,\cdot)$, i.e., $||\nabla^3\hat{\mathscr{L}}(\theta^*)|| \leq M$. Since θ_u is a minimizer of $\hat{\mathscr{L}}_2$, and $\hat{\mathscr{L}}_1$ is smooth, we have that $\nabla\hat{\mathscr{L}}_2(\theta_u) = 0$. Plugging this in the above bound, we get

$$\|\nabla\hat{\mathscr{L}}_2(\theta^*) + \nabla^2\hat{\mathscr{L}}_2(\theta^*)[\theta_u - \theta^*]\| \leq \frac{M}{2}\|\theta^* - \theta_u\|^2. \tag{4.15}$$

Note that

$$\begin{aligned}
\nabla\hat{\mathscr{L}}_2(\theta^*) &= \frac{1}{n-m}\sum_{z\in D^r}\nabla\ell(z,\theta^*) \\
&= \frac{1}{n-m}\sum_{z\in D}\nabla\ell(z,\theta^*) - \frac{1}{n-m}\sum_{z\in D^e}\nabla\ell(z,\theta^*) \\
&= \frac{n}{n-m}\nabla\hat{\mathscr{L}}_1(\theta^*) - \frac{1}{n-m}\sum_{z\in D^e}\nabla\ell(z,\theta^*) \\
&= -\frac{1}{n-m}\sum_{z\in D^e}\nabla\ell(z,\theta^*),
\end{aligned} \tag{4.16}$$

where the equality in the second line holds because $D^r = D\setminus D^e$, the third and last line follows by using the definition of the $\hat{\mathscr{L}}_1(\theta^*)$ and θ^* is the minimizer for the function $\hat{\mathscr{L}}_1(\theta^*)$ and hence $\nabla\hat{\mathscr{L}}_1(\theta^*) = 0$. Plugging the above in Equation (4.15), we get that

$$\|-\frac{1}{n-m}\sum_{z\in D^e}\nabla\ell(z,\theta^*) + \nabla^2\hat{\mathscr{L}}_2[\theta_u - \theta^*]\| \leq \frac{M}{2}\|\theta^* - \theta_u\|^2. \tag{4.17}$$

Now, let us define a vector v such that

$$\theta_u = \theta^* + \frac{1}{n-m}(\nabla^2\hat{\mathscr{L}}_2(\theta^*))^{-1}\sum_{z\in D^e}\nabla\ell(z,\theta^*) + v. \tag{4.18}$$

Plugging it to Equation (4.17), we can achieve that

$$\|\nabla^2 \hat{\mathscr{L}}_2(\theta^*)v\| \leq \frac{M}{2}\|\theta^* - \theta_u\|^2. \tag{4.19}$$

Since the function $\hat{\mathscr{L}}_2$ is λ-strongly convex, we have that $\|\nabla^2 \hat{\mathscr{L}}_2(\theta^*)v\| \geq \lambda\|v\|$ for any vector v. Then, we can achieve that

$$\|v\| \leq \frac{M}{2\lambda}\|\theta^* - \theta_u\|^2. \tag{4.20}$$

Finally, with the Lemma 1 implies that $\|\theta^* - \theta_u\| \leq \frac{2mL}{\lambda n}$ and the above bound, we can achieve that

$$\|v\| \leq \frac{2Mm^2L^2}{\lambda^3 n^2}. \tag{4.21}$$

Pluggin in the vector v in Equation (4.18), we achieve that

$$\|\theta_u - \theta^* - \frac{1}{n-m}(\nabla^2 \hat{\mathscr{L}}_2(\theta^*))^{-1} \sum_{z \in D^e} \nabla \ell(z, \theta^*)\| \leq \frac{2Mm^2L^2}{\lambda^3 n^2}. \tag{4.22}$$

■

Theorem 1 bounds the deviation between $\bar{\theta}_{D \setminus D^e_{k_u}}$ achieved by our influence approximation forgetting method and the empirical unlearned minimizer θ_u. It suggests that maintaining a relatively small size m of the erased data $D^e_{k_u}$ compared to the size n of D in Equation (6.3) will help preserve model utility within a tighter bound.

4.3.2 Optimization Analysis

Here, we mainly aim to prove that the solutions of Equation (4.9) can find the steepest gradient descent direction to achieve unlearning and utility preservation optimally. The solution of Equation (4.9) will satisfy:

Lemma 2. *Let* (d, α) *serve as the resolution to Equation* (4.9)*.*

1. *If* θ_t *is Pareto critical, then* $d_t = 0 \in \mathscr{R}^n$ *and* $\alpha_t = 0$*.*

2. *If* θ_t *is not Pareto critical, then*

$$\alpha_t \leq -\frac{1}{2}\|d_t\|^2 < 0, \nabla \mathscr{L}_{IAF}(\theta_t)^T d \leq \alpha, \nabla \mathscr{L}_{UP}(\theta_t)^T d \leq \alpha. \tag{4.23}$$

A FL model θ is deemed Pareto critical if, within its vicinity, no other solution can simultaneously offer better outcomes for both the influence approximation forgetting and utility preservation objectives. Put another way, if $d_t = 0$, it means no direction capable of enhancing performance for both influence approximation forgetting and utility preservation simultaneously. In this situation, improving performance for one task will inevitably compromise the performance of the other, designating the solution as a Pareto critical point. Conversely, when $d_t \neq 0$, the conditions $\nabla \mathscr{L}_{\mathrm{IAF}}(\theta_t)^T d < 0$ and $\nabla \mathscr{L}_{\mathrm{UP}}(\theta_t)^T d < 0$ signify that d_t represents a viable descent direction for both influence approximation forgetting and utility preservation. Consequently, updates to the solution ought to proceed using $\theta = \theta + \eta d_t$.

4.3.3 Complexity Analysis

This section provides an in-depth analysis of the time and space complexities involved in our FedU-U (with utility preservation) method. Here, p represents the total parameter number of the FL model. Additionally, n signifies the sample number, and d stands for the dimensionality of the feature vector.

Time Complexity: Let $f(p)$ represent the time complexity associated with forward propagation. As highlighted in [35], the time complexity for a single step of backpropagation can reach up to $5f(p)$. Consequently, the complexity of calculating the derivative for each training sample amounts to $6f(p)$. Hence, the overall time complexity for a single training cycle involving all batches of the removed dataset, denoted as B_u, is $6f(p)B_u$. Assuming the unlearning procedure involves T_u iterations, with each iteration requiring t_u time to complete, the cumulative runtime for the local model, focusing solely on the removed dataset, is calculated as $6f(p)B_uT_ut_u$.

Our FedU-U approach incorporates a utility preservation training that utilizes a selected subset from the remaining dataset. In this chapter, we set the size of this selected subset to be equivalent to that of the erased dataset, B_u. Given that the time complexity for a single instance of contrastive forward propagation is $f_c(p)$, the total runtime of the FedU-U method is expressed as $(6f(p)+6f_c(p))B_uT_ut_u$.

In the context of Hessian matrix-based federated unlearning (HFU) methods as discussed by [58], there's a requirement to compute the Hessian matrix locally, utilizing the local remaining dataset, indicated as B_r. We use $h(p)$ to represent the time complexity for computing the Hessian matrix of the model. The total running time can be formulated as $(6f(p)B_u + h(p)B_r)T_ht_h$. To quantify the efficiency improvement, let v

represent the acceleration factor, as determined by the equation

$$v = (\frac{T_u}{T_h})^{-1} = [\frac{(6f(p) + h(p)\frac{B_r}{B_u})T_h t_h}{(6f(p) + 6f_c(p))T_u t_u}]^{-1}, \tag{4.24}$$

where t_u and t_h represent the duration for a single iteration of local training for Algorithm 2 and the HFU method [58], respectively.

Space Complexity: In our Fed-U approach, while we retrain the model for utility preservation, this process relies on a subset of data that is equivalent in size to the dataset marked for erasure. As a result, the space complexity of our method exhibits linear complexity, specifically $\mathcal{O}(2d|D^e| + p)$. This is significantly more efficient compared to existing Hessian matrix-based methods, which require quadratic complexity to compute the Hessian matrix using the remaining dataset, the size of which is $n - |D^e|$. Thus, the space complexity for Hessian-matrix-based methods is $\mathcal{O}(d^2(n - |D^e|) + p)$.

4.4 Performance Evaluation

In this section, we thoroughly assess our federated unlearning methods through extensive experiments. The majority of evaluations in this chapter focus on the unlearning of backdoored samples to clearly demonstrate the effect of the unlearning process. We first illustrate overall effect evaluations of our method and the state-of-the-art federated unlearning methods in Chapter 4.4.2. Second, an evaluation of the model from the perspective of the FL server is detailedly discussed in Chapter 4.4.3. This evaluation encompasses three key aspects: the impact of the number of unlearned users, the erased data ratio, and the number of total FL users. Third, we present a detailed evaluation of local unlearning training in Chapter 4.4.4, which offers a comparison from the user's perspective. Finally, we also conduct an experiment to unlearn normal data in FL to evaluate different federated unlearning methods in Chapter 4.4.5.

4.4.1 Experiment Setting

Datasets and Compared Benchmark. We carry out comprehensive experiments using three popular public datasets, MNIST [20], CIFAR10 [48], and STL-10 [18]. The three datasets are benchmark datasets for image classification tasks, which cover a wide range of object categories with different learning complexities. We compare the performance of our proposed methods with the state-of-the-art federated unlearning method [58]. Since the state-of-the-art method is based on the Hessian matrix, we call

it Hessian-based federated unlearning (HFU). All the comparisons are conducted in the federated learning setting.

Models. In our experiments, we utilize two model architectures of different sizes: a 5-layer multi-layer perceptron (MLP) with ReLU activations and ResNet-18. Specifically, we employ the 5-layer MLP model trained on MNIST, and the ResNet-18 model trained on CIFAR10 and STL-10.

During the implementation of FL model training, the minibatch size is fixed at 100, and the local datasets of all users are independent and identically distributed (IID) and balanced. The learning rate is set to $\eta = 0.001$ for training models on MNIST, and $\eta = 0.0005$ for training models on CIFAR10 and STL-10. Each user's local epochs are $E = 600$ for MNIST and $E = 100$ for CIFAR10 and STL-10. The FL model typically requires 20 global iterations to converge on MNIST and 40 global iterations on CIFAR10 and STL-10, respectively. All models were developed using PyTorch and tested on a cluster equipped with four NVIDIA 1080Ti GPUs.

Evaluation Metrics. For a robust evaluation of unlearning methods, our experiments follow a widely recognized unlearning verification protocol [40]. This involves initially mixing backdoored samples into training datasets for the FL model training. Subsequently, we deploy federated unlearning methods to remove these backdoored samples. After unlearning, we verify whether the backdoor can still attack the model to verify the unlearning effectiveness. Therefore, we have two metrics to assess unlearning effectiveness, i.e., the accuracy on the test dataset and the backdoor accuracy on the erased dataset [40]. To evaluate the efficiency, we consider the running time, determined by timing the training duration for one batch and then multiplying this by the number of training epochs.

Table 4.2: General Evaluation Results on MNIST, CIFAR10, and STL-10

FL server	MNIST, $K=10, K_u=3, EDR=10\%$					CIFAR10, $K=10, K_u=3, EDR=10\%$					STL-10, $K=10, K_u=3, EDR=10\%$				
	Origin	HFU	FedU	FedU-U	Retrain	Origin	HFU	FedU	FedU-U	Retrain	Origin	HFU	FedU	FedU-U	Retrain
Running T.	44	**8.8**	13.2	**8.8**	44	220.8	**16.56**	**16.56**	**16.56**	220.8	154	**6.1**	8.13	**6.1**	154
Accuracy	97.7%	96.62%	96.76%	**97.7%**	97.6%	80.7%	79.3%	78.95%	**80.16%**	80.4%	52.93%	50.83%	50.98%	**51.23%**	52.68%
Bac. Acc.	100%	5.43%	4.86%	**3.96%**	0.08%	89.4%	7.01%	6.04%	**3.73%**	0.3%	15.22%	1.71%	1.33%	**1.0%**	0.5%
User k_u	MNIST, $K=10, K_u=3, EDR=10\%$					CIFAR10, $K=10, K_u=3, EDR=10\%$					STL-10, $K=10, K_u=3, EDR=10\%$				
	Origin	HFU	FedU	FedU-U	Retrain	Origin	HFU	FedU	FedU-U	Retrain	Origin	HFU	FedU	FedU-U	Retrain
Running T.	2.2	0.44	**0.22**	0.44	1.98	5.52	3.3	**0.55**	1.10	4.968	3.85	1.77	**0.38**	1.15	3.85
Accuracy	100%	87.00%	83.83	**99.97%**	100%	100%	84.99 %	82.19%	**96.39%**	100%	100%	79.99%	74.00%	**88.99%**	100%
Bac. Acc.	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	85.99%	2.13%	5.99%	**1.99%**	0%

4.4.2 Overall Evaluation

We present a general and comprehensive evaluation of various federated unlearning methods applied to MNIST, CIFAR10, and STL-10, analyzed from the viewpoints of both the FL server and the unlearning users, as depicted in Table 4.2. We mainly compared three unlearning methods, HFU, FedU and FedU-U, where HFU [58] is the state-of-the-art federated unlearning method, FedU is only executing the influence approximation forgetting method in the user-side, and FedU-U adds the utility preservation method in FedU. We also conduct experiments on retraining from scratch, which, while being the gold-standard method for unlearning, comes with substantial computational costs. We configure the total number of FL users (K) to be 10, with the number of unlearning users (K_u) set to 3, and the erased data ratio (EDR) of each user's local dataset at 10%.

From the perspective of the FL server, compared with HFU and FedU, FedU-U achieves the best performance in all metrics. FedU-U achieves similar effectiveness as retraining but a speedup of at least more than 5× than retraining from scratch. For example, on MNIST, it maintains the model accuracy as the original model, without any model utility degradation, meanwhile reducing the backdoor accuracy to 3.96%. It reduces the FL model's training time from 44 to 8.8 seconds. On STL-10, the running time of FedU-U speedup is more than 20× than the retraining method, with only round 1% model accuracy degradation.

From the perspective of the unlearned user, FedU-U consistently exhibits superior performance in effectiveness, i.e., the highest accuracy within the residual local dataset and the most effective removal of backdoors. However, FedU-U consumes more running time than FedU because FedU only executes the component of influence approximation forgetting. Additionally, HFU always consumes the most local running time due to the necessity of calculating the Hessian matrix using the remaining local dataset.

4.4.3 Evaluation of Global Model

This chapter demonstrates two critical variables, K_u and EDR, which significantly impact unlearning performance. To assess the scalability of various federated unlearning methods, we further investigate the impact of the total number of FL users, represented as K. When evaluating the impact of K_u or EDR, we fix another variable and set $K = 10$. When we evaluate the impact of K, we keep the ratio of unlearned users $\frac{K_u}{K} = 30\%$ and $EDR = 10\%$.

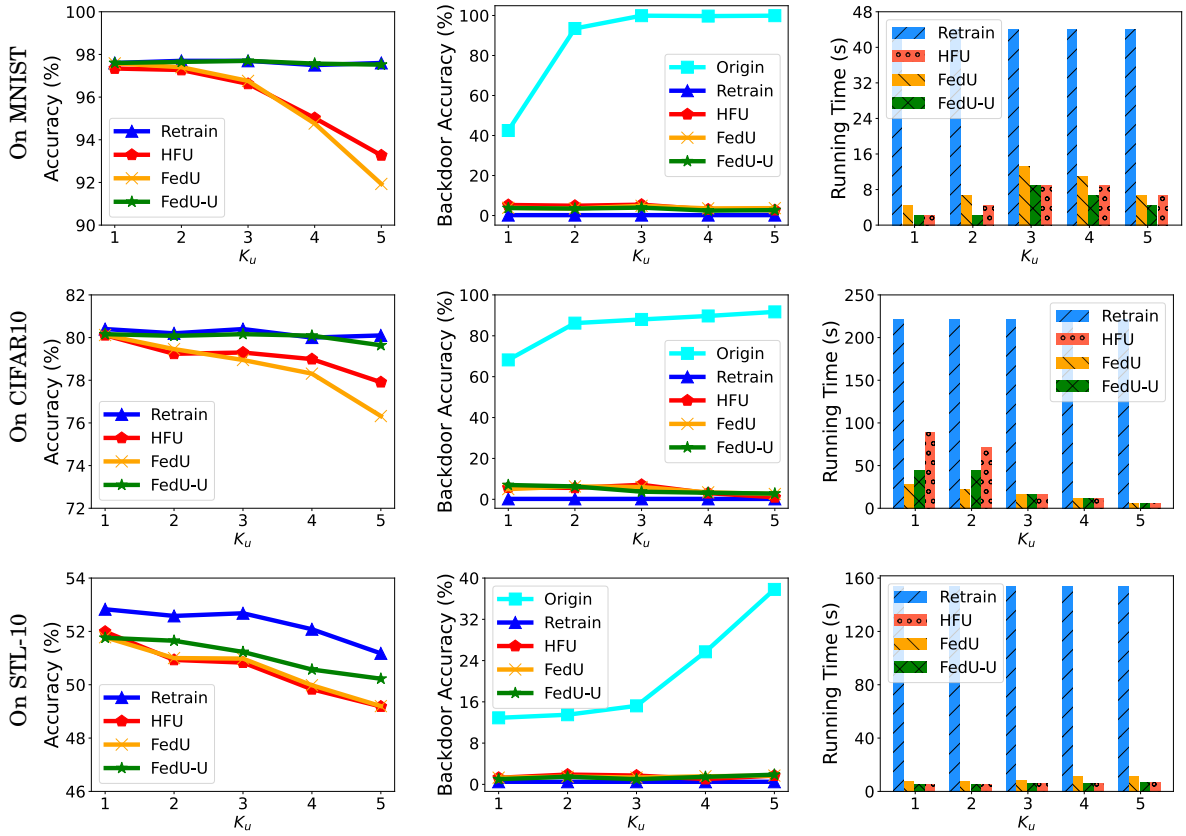
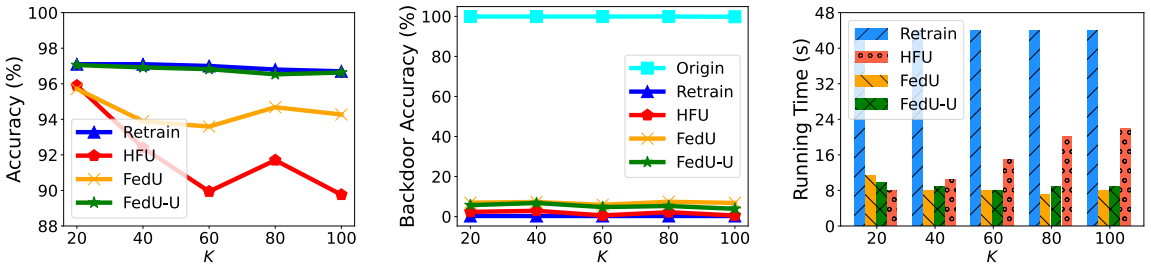


Figure 4.2: The model accuracy, backdoor accuracy and running time of various K_u on MNIST, CIFAR10, and STL-10.

4.4.3.1 Impact of the Number of Unlearned Users

On all datasets, when evaluating the model's performances on different K_u, we keep $EDR = 10\%$ and $K = 10$. Figure 4.2 displays the results of different federated unlearning methods of various K_u, on MNIST, CIFAR10, and STL-10 datasets. It is observed that the backdoor accuracy of the original FL model escalates with an increase in K_u, as depicted under "Origin" in the middle column in Figure 4.2. All methods successfully reduce the backdoor accuracy lower than 10% and have a better removal effect as K_u increases.

In the models' accuracy aspect, shown in the first column in Figure 4.2, FedU-U achieves the least degradation and is the only one similar to the retrained model's accuracy. HFU and FedU perform similarly on all three datasets when K_u is small. However, when K_u increases, the accuracy of FedU drops worse than that of HFU on MNIST and CIFAR10. The running time is different on MNIST, CIFAR10 and STL-10 when K_u



(a) Accuracy of various K (b) Backdoor acc. of various K (c) Running time of various K

Figure 4.3: The model accuracy, backdoor accuracy and running time of various K on MNIST.

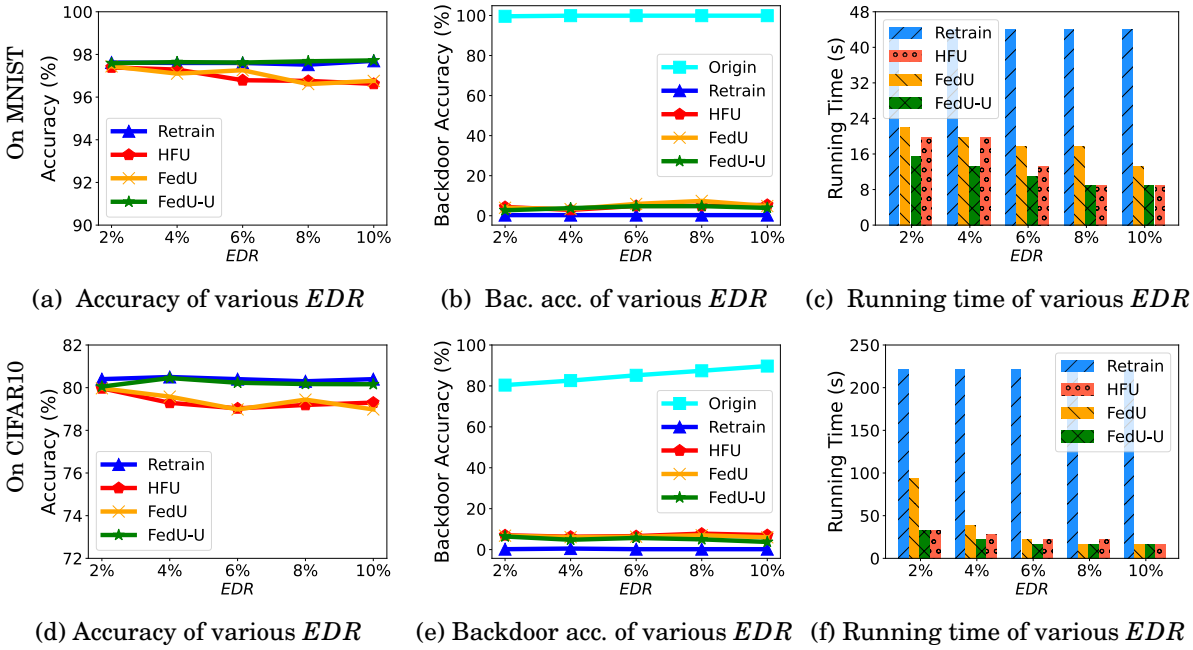
increases presented in Figures 4.2c and 4.2f. We analyze it depending on the distinct models, datasets and backdoor extents. On MNIST, when $K \leq 2$, since the model is shallowly backdoored, the model is easy to unlearn and hence consumes less time. However, on CIFAR10, the Resnet18 model is harder for a few users to modify than an MLP model (on MNIST), hence taking more time when K is small. Notably, both FedU and FedU-U achieve a 5-fold speed-up in comparison to retraining from scratch. By contrast, on STL-10, the backdoor method almost failed to backdoor the global FL model, less than 50% when $K_u = 5$. Since the global model is not well-backdoored on STL-10, removing the influence of backdoored samples takes less time.

4.4.3.2 Impact of the Number of Total FL Users

We conduct additional experiments varying the total number of FL users, denoted as K, to showcase the scalability of different methods. We keep the ratio of unlearned users $\frac{K_u}{K} = 30\%$ and $EDR = 10\%$. The performances of the FL global models on MNIST are shown in Figure 4.3. FedU-U still works effectively in all evaluation metrics as the total number of FL users increases, while HFU has a huge accuracy degradation and running time increase. We parallelly train the same epochs with the same minibatch size of the local users' models, so the total retraining time will not increase as K increases. These results demonstrate the superior scalability of FedU-U and FedU.

4.4.3.3 Impact of the Erased Data Ratio

Figure 4.4 shows the models' performances of different local erased data ratios EDR on MNIST and CIFAR10. In this situation, $K_u = 3$ and $K = 10$, the unlearned users with all different EDR from 2% to 10% can successfully backdoor the global model, as



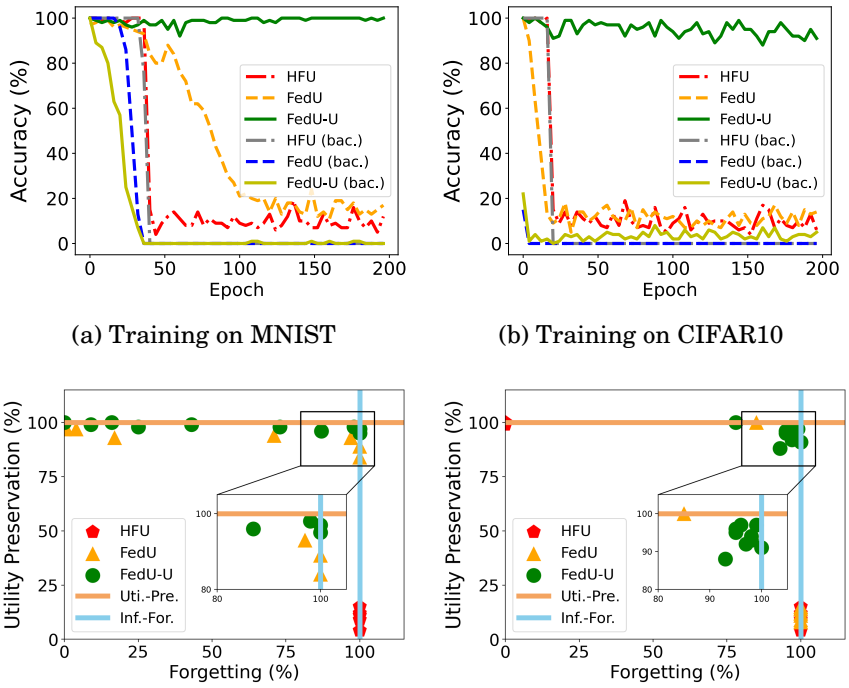
(a) Accuracy of various EDR (b) Bac. acc. of various EDR (c) Running time of various EDR

(d) Accuracy of various EDR (e) Backdoor acc. of various EDR (f) Running time of various EDR

Figure 4.4: The model accuracy, backdoor accuracy and running time of various EDR on MNIST and CIFAR10.

shown in Figures 4.4b and 4.4e. All unlearning methods also have completely removed the influence of backdoored erased data from the global model, making the backdoor accuracy lower than random selecting 10%. These methods consume less running time as EDR increases, shown in Figures 4.4c and 4.4f. They save more than half the running time than the retraining method in all EDR. When EDR = 10%, FedU-U requires less than 1/5 of the running time compared to retraining.

On CIFAR10, all methods can achieve a speedup of more than 10× when EDR is large. However, when EDR is small, the complex nature of learning tasks on CIFAR10 leads to that if the global model has been compromised with a backdoor, it becomes more challenging to effectively unlearn the backdoor with just a small number of erased samples. From the utility degradation aspect, as shown in Figures 4.4a and 4.4d, FedU-U performs best, keeping similar accuracy as the retraining method without degradation. FedU and HFU have a larger accuracy degradation than FedU-U, around 1% accuracy reduction, and FedU has a similar accuracy performance as HFU.



(a) Training on MNIST

(b) Training on CIFAR10

(c) Opt. on MNIST

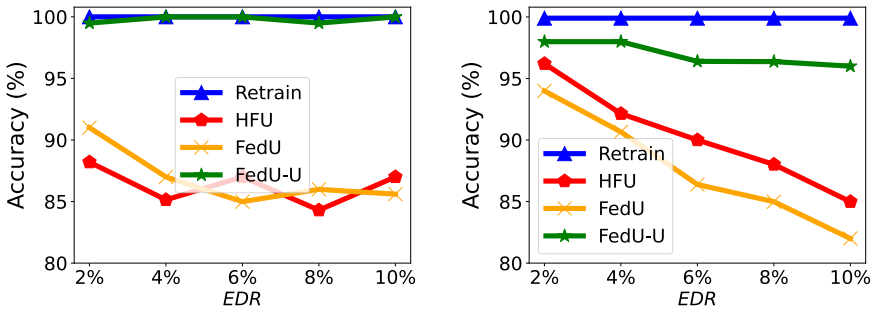
(d) Opt. on CIFAR10

Figure 4.5: Accuracy and backdoor accuracy changes (a and b), and optimization (abbreviated as Opt.) results (c and d) in local unlearning training.

4.4.4 Evaluation of Local Unlearning Training

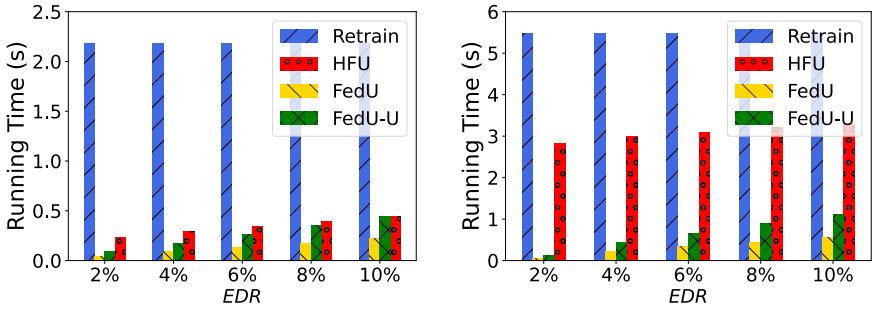
Overall Local Unlearning Results: Since the evaluations on the unlearned user's side have a minor relationship to different K_u, we only show the comparison of different EDR. The outcomes related to accuracy and running time on the remaining data for MNIST and CIFAR10 are presented in Figure 4.6. Specifically, Figure 4.6a illustrates the model accuracy on the remaining local dataset for MNIST, where FedU-U performs similarly to the retraining method, which keeps almost 100% accuracy. FedU and HFU have an obvious accuracy degradation, which moreover decreases as EDR increases. The model accuracy results on CIFAR10 in Figure 4.6b show a similar conclusion. The running time results for MNIST and CIFAR10, as illustrated in Figures 4.6c and 4.6d demonstrate that FedU achieves the lowest running time cost, making it the most efficient, while FedU-U ranks as the second most efficient method.

Detailed Unlearning Optimization Results: Figures 4.5a and 4.5b illustrate the detailed fluctuations in accuracy on the unlearned user's local remaining dataset and the backdoor (abbreviated as bac.) accuracy on the local specified-erased data across



(a) Accuracy on MNIST

(b) Accuracy on CIFAR10



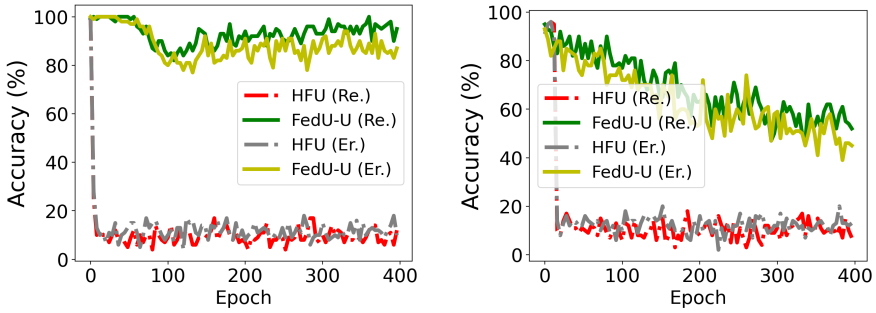
(c) Running time on MNIST

(d) Running time on CIFAR10

Figure 4.6: Local Performances on MNIST and CIFAR10

each local training epoch. These details highlight the optimization process for achieving the objectives of forgetting and utility preservation. FedU-U uniquely maintains the accuracy on the residual dataset throughout the unlearning process, and its backdoor accuracy decreases more rapidly than that of FedU and HFU. Moreover, FedU performs better than HFU, with faster backdoor accuracy dropping and slower accuracy degradation on MNIST. FedU can successfully eliminate the influence of the specified dataset, but its accuracy on the remaining dataset degrades as the unlearning training continues. HFU can effectively forget the erased data. However, this process simultaneously compromises the local model utility on the remaining dataset.

Figures 4.5c and 4.5d display the local optimization results of all unlearning methods, detailing the progress towards the goals of influence approximation forgetting and utility preservation in federated unlearning on MNIST and CIFAR10. In these two figures, we illustrate the effectiveness of utility preservation through the model accuracy on the remaining data and gauge the effect of forgetting as `Forgetting` = 1 − `backdoor accuracy`. FedU-U excels in both utility preservation and forgetting effectiveness, attaining a perfect score of 100% for each. While some outcomes from FedU



(a) Training process on MNIST (b) Training process on CIFAR10

Figure 4.7: Local unlearning normal data training.

manage to strike a balance between forgetting and utility preservation, they do not reach the optimal level demonstrated as FedU-U. HFU struggles to optimally balance the two goals, as its optimization results primarily favor either maximal utility preservation with minimal forgetting effect or vice versa, making it challenging to achieve a satisfactory compromise between the two.

4.4.5 Evaluation of Unlearning Normal Data

We also assess HFU and FedU-U in the context of unlearning normal data on MNIST and CIFAR10, as detailed in Figure 4.7. In this experiment, our aim is to discern the differences between unlearning backdoored samples and normal IID data. Here, we set the $EDR = 6\%$. The experimental findings indicate that the removal of normal data inevitably diminishes the utility of the original model more than unlearning backdoored samples that we showed before.

The accuracy of HFU declines rapidly with ongoing model training on both MNIST and CIFAR10. There is no distinct gap between model accuracy on the remaining dataset and the erased dataset, which indicates that HFU is unable to unlearn the specified data without detrimentally affecting the knowledge learned from the local remaining dataset. Conversely, FedU-U widens the accuracy gap. This indicates that FedU-U enables the model to eliminate the contribution associated with the specified data, even when the data is normal IID, while still preserving the original model. Throughout the training process, FedU-U achieves an accuracy gap exceeding 15% on MNIST. HFU fails to replicate this unlearning effect, struggling to erase the contribution of IID data without compromising the original utility of the model.

4.5 Further Discussion

While checking if a backdoor was unlearned can be seen as a necessary condition to check whether the data was unlearned, unlearning a specific pattern as the backdoor triggers might be different than unlearning the benign or normal data. We have conducted additional experiments on normal data in Chapter 4.4.5, which demonstrates that unlearning normal data will harm the utility of the original model more than unlearning the backdoored samples because the erased normal data is similar to the remaining data. Even though, FedU-U can significantly mitigate the utility degradation than the state-of-the-art methods [58].

Another concern is that for just a few unlearning users, all other users should needlessly run retraining, and it is not practical. Necessitating non-unlearn users to conduct additional training steps impacts efficiency, unlearning effectiveness, and model utility. Involving more additional nodes to participate in unlearning training will increase the computation consumption. At the same time, it also enhances model utility, as the inclusion of more nodes equates to a greater diversity of remaining datasets. However, as the number of non-unlearn nodes increases, the unlearning effect may be diminished, with the influence of the unlearned client's data being progressively reduced. Exploring how many non-unlearn users participate in the retraining to balance the efficiency, unlearning effect, and model utility preservation will be our future work. Alternatively, using mode connectivity [107] to learn the knowledge based on the former model updates of normal users and several benign samples will be a possible solution that avoids the retraining of normal users.

4.6 Summary

In the chapter, we propose FedU and the full version FedU-U to solve the federated unlearning problem. First, FedU-U forgets the information of erased samples by subtracting the estimated data influence from the trained models. Second, it mitigates the accuracy degradation caused by unlearning using a utility preservation method and an adaptive optimization scheme. We conduct comprehensive experiments to assess the effect and efficiency of the proposed approaches, which show superiority over the state-of-the-art federated unlearning methods.

CHAPTER 5

EVALUATING MACHINE UNLEARNING BASED ON MODEL DIFFERENCE

Increasing attention is being paid to machine unlearning, which supports individuals' "right to be forgotten." While most studies focus on the efficiency and effectiveness of unlearning algorithms, the evaluation of machine unlearning effectiveness remains underexplored. Offering robust evaluation services for unlearning is critical, not only to uphold privacy legislation but also to assess and improve existing unlearning methods. Lots of existing methods employ backdoor methods to evaluate unlearning effectiveness, which can only verify the unlearning effect of backdoored samples and negatively impact the model utility as they need to embed backdoors into the model first.

In this chapter, we propose the evaluating machine unlearning (EMU) method, which aims to evaluate the effectiveness of unlearning and verify data removal. Machine unlearning inherently creates a difference on the model before and after unlearning. The model difference contains information about the unlearned samples. Even if the erroneous update of over-unlearning on the retained parameters, the model difference would contain the over-unlearning information and increase at the same time. We utilize the influence function theory to simulate changes of the unlearning model to generate the evaluating inputs efficiently. Additionally, we design a multi-task information bottleneck structure to enhance the scalability of EMU and simplify the analysis of different learning tasks. We provide a theoretical analysis of how the similarity between erased

and remaining samples, as well as task types, affects the extent of unlearning—factors that have been largely overlooked. Extensive experiments on various model architectures and representative datasets confirm our analysis, demonstrating the effective evaluation for unlearning genuine samples and the model utility preservation of our method.

5.1 Problem Statement and Scheme Definition

In this section, we start by introducing the problem statement, then the requirements and metrics of our scheme.

5.1.1 Machine Unlearning Evaluation Problem

To enhance comprehension of the unlearning evaluation challenge, we begin by outlining the core steps involved in the unlearning process.

Machine Unlearning. The common machine unlearning process includes three phases. (1) The ML server trained a model with parameters θ^* based on dataset D. Then, (2) users upload the requested unlearning dataset D_u to the server for unlearning operations. (3) After receiving the unlearning requests, the server employs an unlearning method $\mathscr{U}$ to unlearn D_u's influence from θ^*, outputting an unlearned model $\theta_{u,D\setminus D_u}$.

Given that users may have limited means to ascertain the effectiveness of unlearning, it is crucial for the server—possessing greater computational resources and data access—to assist in evaluating the unlearning execution. We subsequently introduce the problem of machine unlearning evaluation from the ML server' perspective.

Problem Statement (Evaluating Unlearning Effectiveness)**.** *Given the above machine unlearning services, the ML server is tasked with assessing the effectiveness of the unlearning algorithm $\mathscr{U}$ in removing the influence of the unlearning dataset D_u from the original model θ^*. This involves evaluating to what extent the updated model parameters $\theta_{u,D\setminus D_u}$ forgetting the information of D_u.*

The unlearning effectiveness evaluation is essential for ensuring that the unlearning process has been properly executed. By accurately assessing the extent of unlearning, the server helps maintain user trust, uphold data privacy standards, and provide feedback on the unlearning algorithm design.

5.1.2 Unlearning Evaluation Requirements and Metrics

We now define the requirements of an effective solution to the unlearning evaluating problem. In terms of verification capacity and model utility preservation, the scheme must be able to verify the data removal, assess unlearning effectiveness, and preserve model functionality. Since we evaluate the unlearning effectiveness by assessing the unlearned information of the model difference, we first define the model difference.

Model Difference. We can subtract the original-optimized model parameters value using the unlearned model parameters value as the unlearning model difference,

$$\Delta\theta = \theta_u - \theta^*, \tag{5.1}$$

where θ_u is the parameters of the unlearned model, θ^* is the parameters of the original trained model, and $\Delta\theta$ is the model difference, also called unlearning update.

Data Removal Verifiability. Existing methods treat the unlearning model difference as a privacy breach and train membership inference models to infer whether the unlearned samples in the training dataset in a black-box setting [15, 50]. Similar to these works, we train a verifying model to infer whether the unlearned samples are still in the remaining dataset to determine if the unlearning is executed. With the white-box access, the server can easily know the model difference, hence providing a high-accurate data removal verification. We use the Verifiability metric to assess the performance of the verifying model, which computes the correct predicting ratio as

$$\text{Verifiability} := \frac{1}{m} \sum_{x_u \in D_u} \mathbb{I}(\text{Verifier}(\theta^*, \theta_u, x_u) = 1), \tag{5.2}$$

where m is the size of D_u and $\mathbb{I}$ is the indicator function that outputs 1 when the statement is true (i.e., $\text{Verifier}(\theta^*, \theta_u, x_u) = 1$) and 0 otherwise.

Assess How Much Information is Unlearned through Unlearned Information Reconstruction. We train a reconstructing model to distill the unlearned knowledge based on the model difference. To assess the unlearning effectiveness, we use the metric of cosine similarity between the erased and reconstructed samples,

$$\text{Rec. Similarity: } \text{sim}(\hat{X}_u, X_u) = \frac{\hat{X}_u \cdot X_u}{||\hat{X}_u|| \cdot ||X_u||}. \tag{5.3}$$

Here, X_u is the original unlearned data vectors, and $\hat{X}_u$ is the reconstructed vectors, and $\hat{X}_u \cdot X_u$ is the corresponding dot product. $||\hat{X}_u||$ and $||X_u||$ are the Euclidean norms of the two vectors. For the same reconstruction model, a larger reconstruction similarity

reflects more information related to the unlearned data contained in the unlearning model difference.

Functionality Preservation. The unlearning evaluating scheme should not come at the cost of significant model utility degradation. In other words, the performance of the model after processing evaluating should be only slightly worse than, if not equivalent to, the originally trained model. The formal definition is

$$\Pr_{(x,y)\in D}[\theta(x)=y] \approx \Pr_{(x,y)\in D}[\theta_{\mathrm{E}}(x)=y]. \tag{5.4}$$

It checks that the performance of the model with an evaluating module θ_{E} does not deviate too much from those of original trained θ.

5.2 EMU base on Model Difference

5.2.1 Overview of the EMU

We solve the unlearning effectiveness evaluating problem by analyzing the difference between the model before and after unlearning, which is called EMU. Specifically, EMU includes three main components: unlearning model difference generation, assessment model training, and random division to make EMU suitable for multi-sample unlearning evaluation.

5.2.2 Unlearning Model Difference Generation

When preparing the unlearning model difference, we should notice that only waiting for the unlearning requests for model difference generation is impractical and slow. To speed up the process, we sample an auxiliary dataset D_a from the full training dataset D. It is easy to implement as the server has access to all the training data. The model difference based on the current trained model θ^* and D_a can be described as

$$\Delta\theta_{x_a} = \theta_{D\setminus(x_a\in D_a)} - \theta^*. \tag{5.5}$$

This is an example to generate model difference $\Delta\theta_{x_a}$ for a single sample $x_a \in D_a$. In multi-sample scenarios, we just need to sample a subset $\{x_1, x_2, ..., x_a\} \subseteq D_a$ as the simulated unlearning dataset D_u. One drawback is that the training computation consumption is expensive if we prepare the unlearning model difference by conducting a gold-standard unlearning method, retraining from scratch. To simulate the model difference efficiently, we utilize an influence approximation method to achieve approximate $\Delta\theta$.

Approximate Model Difference Generation. The process is similar to approximate unlearning methods but is based on the influence function theory in ML [4, 6, 47]. Indeed, we do not need to truly unlearn the model. We just calculate the approximate unlearning model difference at current θ^*, which can be approximated as follows.

Theorem 2 (First-order model difference approximation)**.** *When computing the influence of a few samples, such as m samples, the scaling of the first-order $\theta^{(1)}$ is $\frac{m}{n}$, while the scaling of the second-order coefficient is $\frac{m^2}{n^2}$, which is very small when n is large. Thus, the second-order term can be ignored, and the model difference can be approximated using the first-order influence estimation as*

$$\Delta\theta = \theta_{D\setminus D_u} - \theta^* \simeq \frac{1}{n-m} H_{\theta^*}^{-1} \sum_{x_u \in D_u} \nabla L(x_u;\theta^*), \tag{5.6}$$

where $\theta_{D\setminus D_u}$ is the unlearning model, and$H_{\theta^*}^{-1}$ denotes the inverse of the Hessian matrix evaluated at θ^*.

We omit the proof of Theorem 2, which is similar to the proof in [47]. Calculating the model difference using the above approximation method has been demonstrated to be much more efficient than the naive retraining method. During the practical training process, we can utilize the stochastic estimation method and Hessian-vector products (HVPs) to further speed up the Hessian matrix calculation following [47].

Based on the influence function theorem of approximate model differences, we can further analyze how the similarity between the removed samples D_u and the remaining samples $D\setminus D_u$ affects the model differences. For easy understanding, we simplify the two similarity scenarios: removing a unique sample (a sample that appears only once in the dataset, lowest similarity) and removing a duplicate sample (a sample that appears multiple times, highest similarity). Then, we have the following theorem.

Theorem 3 (Model difference influenced by erased samples)**.** *Let D be a training dataset of size n, possibly containing duplicates. Let $x_{uniq.}$ be a unique sample in D (appears only once), and let x_d be a duplicate smaple in D (appears k times, $k \geq 2$). Given a convex loss function $L(x;\theta)$, the unlearning change in model parameters $\Delta\theta_j$ when removing a sample x_j is proportional to the influence of the sample on the model. Specifically, for a unique sample $x_{uniq.}$ and a duplicate sample x_d appearing k times, the model diiferences satisfy:*

$$\|\Delta\theta_{uniq.}\| > \|\Delta\theta_d\|, \tag{5.7}$$

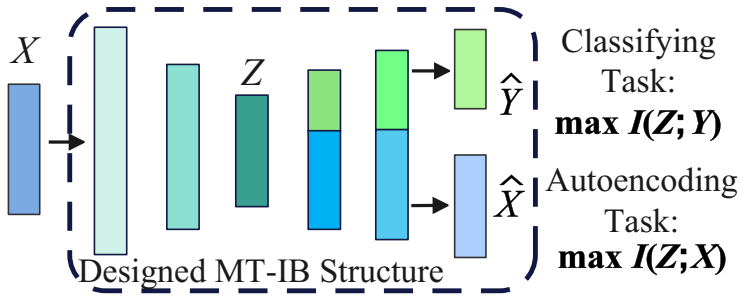


Figure 5.1: The MT-IB Structure.

when $k \geq 2$*, assuming the gradients* $\nabla_\theta L(x_{uniq.};\theta^*)$ *and* $\nabla_\theta L(x_d;\theta^*)$ *are of comparable magnitude.*

We present the proof of Theorem 3 in Section 5.3.1.

Simplify Unlearning Model Difference with Specific Layer Difference. We can achieve the best evaluating effect if we use the flattened parameters of the service model difference. However, using all the parameters as input for evaluating would be computationally expensive especially the models are large. We design the multi-task information bottleneck (MT-IB) structure and only choose the difference of the informative representation layer to replace the whole model difference to improve the scalability.

Different from the original information bottleneck (IB) architecture [2, 88], we design the MT-IB structure as shown in Figure 5.1. MT-IB includes two different output layers to assist in analyzing how the different ML tasks will influence the unlearning effect and extent. Layers from the input layer to the representation layer are the same as in the original IB, minimizing mutual information between the representation Z and the original sample i. We divide the layers following the representation layer into two task objectives: the classifying task and the autoencoding task. The classifying task maximizes the mutual information between Z and the task target Y, $I(Z;Y)$. The autoencoding task aims to learn the most information about the original samples. Hence, it maximizes the mutual information $I(Z;X)$. We rewrite the IB objective for the MT-IB as below,

$$\mathscr{L}_{\text{MT-IB}} = I(Z;i) - (1-\alpha)I(Z;Y) - \alpha I(Z;X), \tag{5.8}$$

where $\alpha \in [0,1]$ is a proposed task weight that controls the model attention between the classifying and the autoencoding tasks. When $\alpha = 0$, the MT-IB is a classification model, and when $\alpha = 1$, the MT-IB is an autoencoding model. Changing α from 0 to 1 indicates the learned information extent about the original data X and assists in analyzing how the ML task T influence unlearning effectiveness, and we have

Theorem 4 (Upper bound of representation layer information for task type T)**.** *In the IB framework, for a given learning task T and input data X, the information that the representation layer Z learned is upper-bounded by the mutual information between X and the task T:*

$$I(X;T) \geq I(Z;T). \tag{5.9}$$

It implies that the learning task highly impact the information that the representation learned, hence influence what the representation unlearned during unlearning. We present the analysis of Theorem 4 in Section 5.3.2.

5.2.3 Machine Unlearning Evaluation Models Training

Inspired by [76], we train the Reconstructor using an autoencoder, which includes an encoder and a decoder. The target of the Reconstructor is to learn an encoding for the model differences $\Delta_z\theta_a$. Here, Δ_z means the difference of representation layer Z. In the autoencoder model, the encoder takes the model difference as the input and encodes it into a latent representation μ, and the decoder needs to decode the latent representation to recover the erased samples. We use mean squared error (MSE) as the loss function, which can be described as

$$\mathscr{L}_{Rec} = ||\hat{X_u} - X_u||_2^2, \tag{5.10}$$

where $\hat{X_u}$ is the reconstructed sample for X_u. As introduced in Equation (5.3), we can use the reconstruction similarity to assess the unlearning extent of specified data on the model.

5.2.4 Random Division Strategy for Multi-Sample Unlearning Evaluation

If we follow the above procedure to train the assessment model, it will be effective for single-sample unlearning evaluation. However, unlearning evaluation for multi-sample scenarios is challenging because all samples' information in this unlearning setting is interwoven into one model update. We propose the following strategy to enable the assessment model to reconstruct for multiple samples.

Influence-based Division. The influence-based division strategy continues utilizing the convenient properties of the first-order model difference approximation method. After achieving the overall model difference $\Delta_z\theta_{D_u}$ for multiple samples, we directly simulate the model influence approximation for each sample. Then, we divide the received

total model difference by the calculated weight of each sample. Suppose each divided model difference obeys a Gaussian random distribution; then we have:

$$\begin{aligned} \Delta_z\theta_{x_u} \sim \mathcal{N}\left(\frac{\Delta_z\theta_{D_u}}{H_{\theta^*}^{-1}\sum_{x_u\in D_u}\nabla L(x_u;\theta^*)}\cdot\nabla L(x_u;\theta^*),\sigma^2\right), \\ \text{s.t. } \Delta_z\theta_{D_u} = \sum_{x_u\in D_u}\Delta_z\theta_{x_u}. \end{aligned} \tag{5.11}$$

In Equation (5.11), we treat the assigned model difference as the mean and a deviation is added to it. The requirement is that the sum of the divided model difference slice $\sum_{x_u\in D_u}\Delta_z\theta_{x_u}$ should equal to the received entire model difference $\Delta_z\theta_{D_u}$. Thus, we ensure every inference and reconstruction has a special separated model difference, and no additional noise will be introduced to the original $\Delta_z\theta_{D_u}$.

5.3 Theoretical Analysis

5.3.1 Proof of Theorem 3

Proof. According to empirical risk minimization (ERM), the model parameters θ^* minimize the empirical risk over training dataset D:

$$\theta^* = \arg\min_{\theta}\frac{1}{n}\sum_{i=1}^{n}L(x_i;\theta). \tag{5.12}$$

The influence of a sample x_j on the parameters θ^* can be approximated using influence functions according to Theorem 2:

$$\Delta\theta_j \approx -\frac{1}{n-1}H^{-1}\nabla_\theta L(x_j;\theta^*), \tag{5.13}$$

where H is the Hessian of the total loss at θ^*. The model difference when removing a unique sample $x_{uniq.}$ is

$$\Delta\theta_{uniq.} \approx -\frac{1}{n-1}H^{-1}\nabla_\theta L(x_{uniq.};\theta^*). \tag{5.14}$$

Since $x_{uniq.}$ appears only once, its influence is proportional to $\frac{1}{n-1}$. After removing one instance of x_d, it appears $k-1$ times in $D_{\setminus d}$. The model difference is

$$\Delta\theta_d \approx -\frac{1}{n-1}H^{-1}(\nabla_\theta L(x_d;\theta^*)\times\frac{1}{k}). \tag{5.15}$$

Assuming gradients are comparable, i.e., $\|\nabla_\theta L(x_{uniq.};\theta^*)\| \approx \|\nabla_\theta L(x_d;\theta^*)\|$. The ratio of model difference is

$$\frac{\|\Delta\theta_{uniq.}\|}{\|\Delta\theta_d\|} \approx \frac{1}{1/k} = k. \tag{5.16}$$

Since $k \geq 2$, we have

$$\|\Delta\theta_{uniq.}\| > \|\Delta\theta_d\|. \tag{5.17}$$

■

Here, we also provide an example to explain how the similarity of the erased and remaining samples influences the unlearning model difference and unlearning evaluation.

Let us consider a linear model with parameters θ and two input vectors $\mathrm{v}_1 = [v_{1,1}, v_{1,2}, v_{1,3}]$ and $\mathrm{v}_2 = [v_{1,1}+\epsilon_1, v_{1,2}+\epsilon_2, v_{1,3}+\epsilon_3]$. If ϵ_i (for $i \in [1,2,3]$) represents dissimilar difference (could be noise or other dissimilarities), then v_2 can be represented by v_1 as $\mathrm{v}_2 = \mathrm{v}_1 + \epsilon$. If $\epsilon \simeq 0$, then $\mathrm{v}_2 \simeq \mathrm{v}_1$. Assume we have a model θ_{v_1} trained on v_1, and then we train it on v_2 and achieve

$$\begin{aligned} \theta_{[\mathrm{v}_1,\mathrm{v}_2]} &= \theta_{\mathrm{v}_1} + \Delta\theta_{\mathrm{v}_1}(\mathrm{v}_2) \\ &= \theta_{\mathrm{v}_1} + \Delta\theta_{\mathrm{v}_1}(\mathrm{v}_1 + \epsilon) \\ &= \theta_{\mathrm{v}_1} + \Delta\theta_{\mathrm{v}_1}(\mathrm{v}_1) + \Delta\theta_{\mathrm{v}_1}(\epsilon). \end{aligned} \tag{5.18}$$

Since the linear model θ_{v_1} is optimized using v_1, the $\Delta\theta_{\mathrm{v}_1}(\mathrm{v}_1)$ will be minimized at the same time, i.e., $\Delta\theta_{\mathrm{v}_1}(\mathrm{v}_1) \simeq 0$. Now the $\Delta\theta_{\mathrm{v}_1}(\mathrm{v}_2) \simeq \Delta\theta_{\mathrm{v}_1}(\epsilon)$. If the dissimilarity $\epsilon = 0$, then the model difference $\Delta\theta_{\mathrm{v}_1}(\mathrm{v}_2) = \theta_{[\mathrm{v}_1,\mathrm{v}_2]} - \theta_{\mathrm{v}_1}$ for unlearning v_2 from $\theta_{[\mathrm{v}_1,\mathrm{v}_2]}$ will be approximate to 0, which is hard for reconstructing unique information of v_2. If $\epsilon > 0$, then we can reconstruct the unique information of v_2 that different from v_1 using $\Delta\theta_{\mathrm{v}_1}(\epsilon)$.

5.3.2 Impact of ML Task Type

During the training structure of MT-IB in Figure 5.1, the Markov chain of classifying task is established as $i : (X,Y) \to Z \to \hat{Y}$. The Markov chain of autoencoding task is established as $i : (X,Y) \to Z \to \hat{X}$. Due to the Markov chain principle, once information is lost in one layer, it cannot be regained in subsequent layers. This process can be described using mutual information as

$$\begin{aligned} &\textbf{Classifying task:} \quad I(X;Y) \geq I(Z;Y) \geq I(\hat{Y},Y) \\ &\textbf{Autoencoding task:} \quad I(X;X) \geq I(Z;X) \geq I(\hat{X},X). \end{aligned} \tag{5.19}$$

Assume we have a task T, following the Markov chain in the MT-IB model structure, we have $i : (X,T) \to Z \to \hat{T}$. For an effective IB model for task T, it needs to learn

a representation Z that maximizes the mutual information $I(Z;T)$. According to the Markov chain and the mutual information relationship in Equation (5.19), we have $I(X;T) \geq I(Z;T) \geq I(\hat{T},T)$. Now, we have the reconstruction upper bound from the mutual information perspective is $I(X;T)$.

When the task is an autoencoding task, the T is to recover X, and the model will learn most information about data X, hence, achieving the largest mutual information $I(X;X) = H(X)$. When the task T is not related to the data X, and the model will learn nothing about T from X, hence, achieving the lowest mutual information $I(X;T) = 0$.

5.4 Performance Evaluation

5.4.1 Experimental Settings

In this section, we first introduce the datasets, models, and evaluation metrics used in the experiments. Then, we introduce the compared unlearning evaluation and verification methods, and the machine unlearning benchmarks.

Datasets. We have conducted experiments on three representative datasets: MNIST [20], CIFAR10 [48], and CelebA [59]. Both MNIST and CIFAR10 are used to train a 10-class classification model. Our experiment on the CelebA dataset is a binary classification task that aims to recognize the gender features based on the CelebA face images.

Models. We employ three model architectures of varying sizes for our experiments: a 7-layer convolutional neural network (CNN), a 5-layer multi-layer perceptron (MLP), and ResNet18. For the MNIST dataset, we employ two MLPs to form the reconstruction model. For models that on CIFAR10 and CelebA, we employ two CNNs to establish the Reconstructor. The ML service model for all datasets follows the MT-IB structure, with ResNet-18 used before the representation layer to learn effective features. After the representation layer, a 5-layer MLP is used for classification, and a 7-layer CNN for autoencoding.

Additionally, we implement a verifying model built as a 5-layer MLP to directly evaluate the verifiability, ensuring our method is comparable with these backdoor-based verification techniques. We train this verifying model post-reconstruction to determine whether recovered samples are unlearned to assess the data removal. The process starts with constructing a dataset for verification. In this dataset, we set the label to 1 for the samples reconstructed from the model difference of the unlearned samples. Conversely, for data that hasn't undergone the unlearning process, a negative label is assigned to

both the instance and its corresponding reconstructed pair. We then train a Verifier model using the constructed data, and the fully trained model is returned as the final verification model.

For the MNIST dataset, we use a learning rate of $\eta = 0.001$, and for CIFAR10 and CelebA, the learning rate is set to $\eta = 0.0005$. The minibatch size is 16 for MNIST and CIFAR10, and 160 for CelebA. All experiments are implemented in PyTorch 3.8 and run on NVIDIA Quadro RTX 6000 GPUs.

Metric. We use the model accuracy to evaluate the model utility preservation, the reconstruction similarity between the reconstructed sample $\hat{X}_u$ and erased sample X_u to assess the unlearned extent, and verifiability to recognize whether the data is unlearned. Additionally, we employ the running time to evaluate the efficiency. We summarize the metrics as follows:

- **Model Accuracy.** It is used to evaluate functionality preservation and to show if the evaluating methods will influence the original ML service utility.
- **Reconstruction Similarity.** It assesses the unlearning effectiveness by reconstructed cosine similarity between the reconstructed sample $\hat{X}_u$ and erased sample X_u, as shown in Equation (5.3).
- **Verifiability.** It is utilized to assess the data removal, which calculates the classifying accuracy of the Verifier as Equation (5.2).
- **Running Time.** It assesses the efficiency of methods, calculated by recording the time used in each training batch and multiplying it with the training epochs.

Unlearning Evaluation and Verification Methods. There are primarily three unlearning verification methods [38, 40, 82], all using the backdooring techniques. Guo et al. [38] put much attention into designing invisible backdoor triggers, hence, to some extent, diminishing the verification ability. MIB [40] has the best verification effect among these three method. Therefore, we directly compare our EMU with the MIB method. Note that these backdoor-based solutions can merely verify the unlearning of backdoored data. Therefore, our experiments for MIB are mostly set for unlearning backdoored samples. By contrast, we evaluate our method for unlearning genuine data.

Machine Unlearning Methods. We conduct the machine unlearning evaluation experiments on four mainstream unlearning methods: SISA [11], HBU [36], VBU [67] and RFU [97]. We briefly summarize these unlearning benchmarks below,

- **SISA [11].** SISA is an exact unlearning method. The main process of SISA divides the dataset D into several shards $D^1, D^2, ..., D^k$ and trains sub-models with parameters $\theta^1, \theta^2, ..., \theta^k$ for each shard. When the server receives a request for unlearning sample x_u, it just needs to retrain the sub-model θ^i of shard D^i that contains x_u. We set $k = 5$ disjoint shards and corresponding sub-models. We put the unlearned samples only on one shard, which is the ideal scenario of SISA.

- **VBU [67].** VBU is an approximate unlearning method based on variational Bayesian inference. For the convenience of experiments, we set a middle layer of original neural networks as the Bayesian layer and calculate the unlearning loss according to [67] based on the Bayesian layer and erased samples for unlearning.

- **HBU [78].** HUB is an approximate unlearning method, which needs to calculate the inverse Hessian matrix of the remaining dataset as the weight for an unlearning update. We implement HBU follow the unlearning process as introduced in [78].

- **RFU [97].** RFU is an approximate unlearning method. It tries to unlearn a bottleneck representation by minimizing the mutual information between the representation and the erased samples. We set a middle layer of the original model as the representation layer and implement unlearning according to [97].

Table 5.1: Overall Evaluation of Different Methods.

Single-Sample Unlearning Evaluation	MNIST, *USS* = 1			CIFAR10, *USS* = 1			CelebA, *USS* = 1		
	MIB	EMU(CT)	EMU(AT)	MIB	EMU(CT)	EMU(AT)	MIB	EMU(CT)	EMU(AT)
Model Accuracy	98.31%	99.23%	**99.31%**	79.45%	81.34%	**81.44%**	96.05%	97.08	**97.38%**
Rec. Sim.	-	0.441	**0.967**	-	0.895	**0.975**	-	0.839	**0.978**
Verifiability	0.00%	87.93%	**99.33%**	0.00%	0.00%	**95.64%**	0.00%	0.00%	**93.14%**
Running time (s)	639	**135**	145	672	**136**	137	1622	33.74	**32.32**
Multi-Sample Unlearning Evaluation	MNIST, *USS* = 20			CIFAR10, *USS* = 20			CelebA, *USS* = 20		
	MIB	EMU(CT)	EMU(AT)	MIB	EMU(CT)	EMU(AT)	MIB	EMU(CT)	EMU(AT)
Model Accuracy	98.73%	99.15%	**99.32%**	79.13%	81.23%	**81.34%**	96.88%	97.25%	**97.31%**
Rec. Sim.	-	0.436	**0.936**	-	0.898	**0.971**	-	0.839	**0.972**
Verifiability	0.00%	92.50%	**95.83%**	0.00%	0.00%	**94.68%**	0.00%	0.00%	**92.62%**
Running time (s)	638	124	**115**	673	**101**	110	1663	21.93	**21.77**

USS: Unlearned Sample Size; CT: Classifying Task, $\alpha = 0$; AT: Autoencoding Task, $\alpha = 0.9$.

5.4.2 Overview Effectiveness of EMU

We present the general evaluation results on different datasets in Table 5.1.

Setup. We assess the evaluation methods from the proposed five metrics, and we mainly focus on two scenarios: the single-sample unlearning evaluation (the upper half of Table 5.1) and the multi-sample unlearning evaluation (the lower half). For single-sample unlearning verification, we set the Unlearning Sample Size (*USS*) equal to 1. For the multi-sample unlearning scenario, we set the $USS = 20$. The evaluation of EMU includes the classifying task (CT) and autoencoding task (AT), $\alpha = 0$ and $\alpha = 0.9$, respectively. Here, we choose $\alpha = 0.9$ rather than $\alpha = 1$ (the model only has the autoencoder task) because we still hope to use the model accuracy to evaluate the model utility. Although MIB is only suitable for the classifying task, we use the same model structure as EMU and set $\alpha = 0.9$ for the implementation of MIB. The unlearning benchmark used here is the representative unlearning method SISA [11]. We use the bolded values to illustrate the best performance and the red-colored values to show that the results are opposite from expectations.

Evaluation of Functionality Preservation. We measure the functionality preservation using the accuracy of the model. Either in single-sample or multi-sample scenarios, EMU always has more than 1% higher model accuracy than MIB. The MIB, the backdoor-based method, adds backdoored data into the training dataset, which modifies the labels according to the backdooring triggers, negatively influencing model utility. By contrast, the EMU method does not participate in the model learning and unlearning processes. Therefore, EMU has no impact on the model utility for the ML service, demonstrating the best functionality preservation.

Evaluation of Unlearning Effectiveness Assessment. Reconstruction similarity and verifiability are utilized to measure the assessment of unlearning effectiveness. The MIB method is unable to solve the Section 5.1.1 to provide the unlearning effectiveness assessment. Therefore, we set a dash for MIB in the reconstruction similarity metric. The reconstruction similarity for the autoencoding task consistently performs much better than for the classifying task, demonstrated in both single-sample and multi-sample unlearning scenarios. It confirms the previous analysis that tasks requiring more information related to the data make models learn more information, in which the unlearning model difference will provide more information, showing a higher unlearning influence of the data on the model.

In the verifiability metric, the MIB cannot successfully verify any genuine samples unlearning in Table 5.1 because the aim is to evaluate the unlearning for genuine samples. EMU provides effective data removal verification (accuracy more than 90%) for the autoencoding task on all datasets. Although EMU also fails to evaluate the data re-

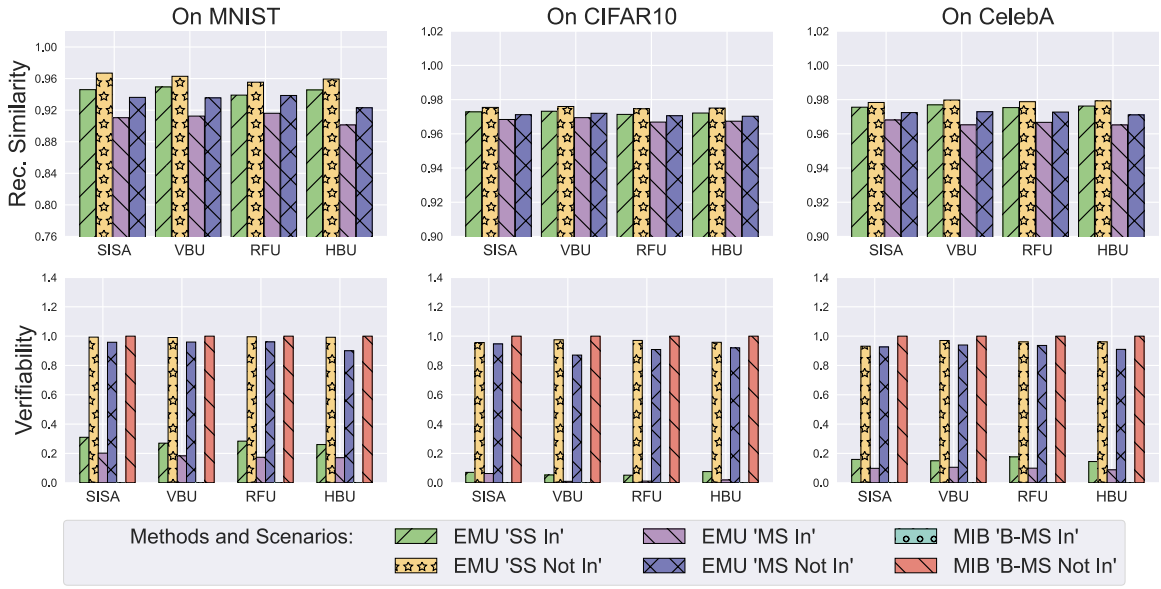


Figure 5.2: Evaluating unlearning effect about different unlearning methods. Exact unlearning methods always achieve better unlearning effects than approximate unlearning methods.

moval for the classifying task on CIFAR10 and CelebA, it provides a high verifiability on MNIST. We infer it is because the MNIST sample does not contain redundant information besides the labels, and it is easily learned by models. The classifying task already enables the model to learn most of the information of the MNIST samples. Hence, EMU can achieve high verifiability for the classifying task as for the autoencoding task on MNIST.

Evaluation of Efficiency. When demonstrating the running time, EMU did not participate in the model training process for evaluating preparation. Therefore, EMU is much more efficient than the MIB method, which needs to be involved in the model learning process for backdooring. Specifically, EMU obtains at least 5× speedup for evaluation on all datasets.

5.4.3 Evaluations for Various Unlearning Benchmarks

Setup. In this experiment, we employ four representative unlearning methods to test the evaluation effect of EMU and MIB. The experimental results are presented in Figure 5.2. This experiment also includes two scenarios: the single-sample (SS) and the multi-sample (MS), where USS=1 and USS=20, respectively. From previous experimental results, we know that MIB is infeasible in verifying the genuine samples during unlearning. Here, we evaluate MIB based on backdoored multi-samples (B-MS), denoted

as $D_{u,b} \leftarrow (X_u + noise, Y_{backdoor})$. We embed a white block noise patch at the bottom right of selected data and modify the labels for the triggers. To make the evaluation of EMU align with MIB, we also add the noise patch to the erased samples. However, we do not change the labels, which ensures the genuine utility of these data. The modified data for EMU evaluation can be denoted as $D_{u,noi} \leftarrow (X_u + noise, Y_u)$. To straightforwardly display the compared verification results of having or not having unlearned specified samples, we correspondingly set two situations, i.e., the erased samples $D_{u,b}$ and $D_{u,nois}$ not in or still in the remaining dataset. Here, the task weight α is set to 0.9.

Evaluations of Unlearning Effectiveness Assessment. The first row, the reconstruction similarity, illustrates the evaluation results of the unlearning effectiveness assessment. Since MIB is unable to measure how much information is removed from the trained model, we omit evaluating MIB in this row. On all three datasets, it is obvious that unlearning the specified samples (not in the remaining dataset) offers a more pronounced model difference in extracting the erased information than not unlearning the specified samples (in the remaining dataset). Compared with approximate unlearning methods (VBU, RFU, and HBU), the exact unlearning method (SISA) has a more obvious reconstruction similarity gap for "unlearned" and "not unlearned". This is because the approximate unlearning methods usually limit the unlearning update extent to prevent catastrophic unlearning. Therefore, it is harder to assess how much information is removed for approximate unlearning methods.

The second row demonstrates the verifiability evaluation of MIB and EMU. There are clear gaps for all unlearning verification methods when the specified samples are unlearned or not unlearned on all datasets. The gap implies that all unlearning benchmarks have effectively unlearned samples, and the evaluation methods effectively verify the unlearning. Nevertheless, at the same time, we should notice that MIB can solely verify the backdoored samples $D_{u,b}$, while EMU has the ability to verify the genuine samples $D_{u,noi}$.

5.4.4 Impact of the Similarity between the Erased and Remaining Data

Here, we evaluate the similarity between the erased and remaining data, and we find that it has a significant influence on unlearning effectiveness, which confirms our previous analysis.

Setup. To quantify the similarity and make the experiments easy to understand, we

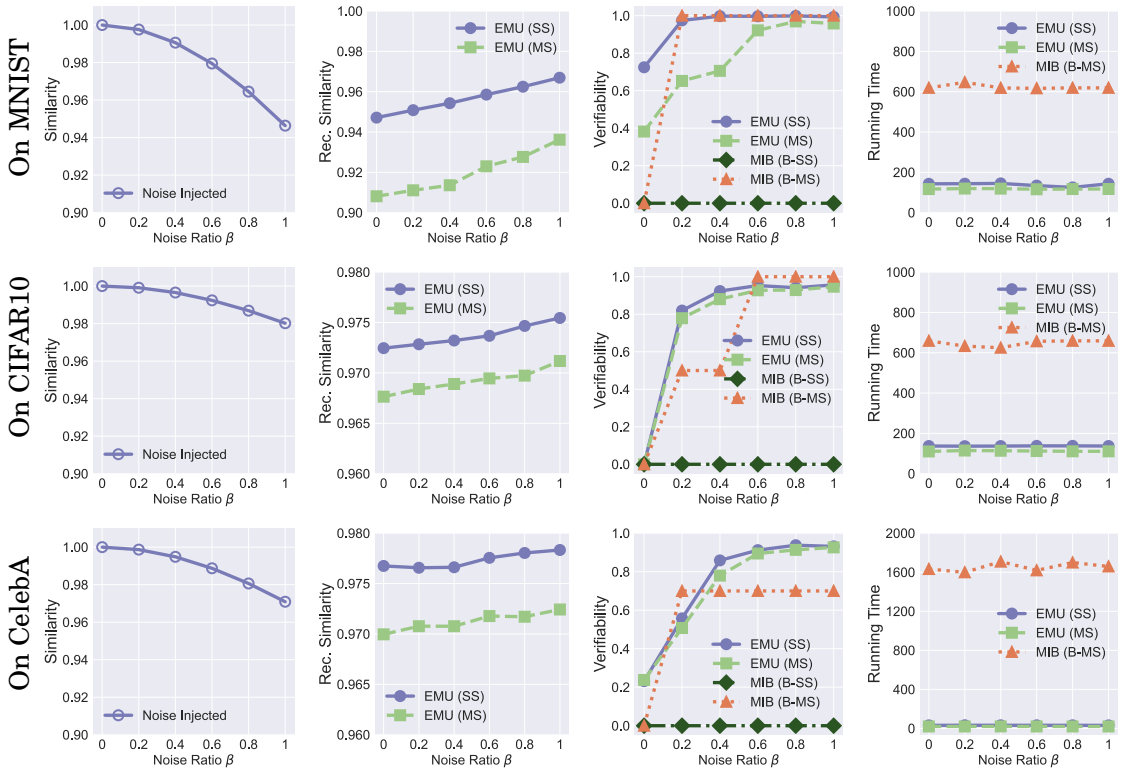


Figure 5.3: Evaluations of the impact of the similarity between the erased and remaining data. "SS" means single-sample unlearning scenario, and "MS" stands for multi-sample unlearning scenario. "B-SS" denotes backdoored single-sample scenario.

keep the previous unlearning and evaluating process and just control the noise amount injected into the unlearned samples. We add a noise rate β to control the noise injection, and the final unlearned data is $D_{u,\beta} \leftarrow (X_u + \beta \times noise, Y_u)$. We mixed both $D_{u,\beta}$ and original D_u into the training dataset. When unlearning, we only remove the contribution of noise-injected samples ($D_{u,\beta}$) from the model while keeping the original samples (D_u) in the remaining dataset. This way, we can intuitively quantify the similarity between the unlearned samples and the samples in the remaining dataset. The task weight α is 0.9, and the unlearning method that we tested here is SISA. We demonstrate the corresponding experimental results on different datasets in Figure 5.3.

Relationship between Data Similarity and Noise Injection Ratio. The first column in Figure 5.3 shows the relationship between the noise injection ratio β and similarity, where the similarity is between the original image data D_u and the noise-injected data $D_{u,\beta}$. On all the datasets, adding more noise will decrease the similarity.

Impact on Unlearning Effectiveness Assessment. If the unlearned samples are more dissimilar to the remaining samples, the model difference will be more remark-

able for reconstruction, with a higher reconstruction similarity, as shown in the second column in Figure 5.3, which confirms the analysis in Theorem 3. The trend in both SS and MS scenarios confirms this claim. Higher reconstruction similarity demonstrates that the unlearning model difference contains more information about the unlearned samples, indicating more information is removed from the model, showing better unlearning effectiveness.

The third column of Figure 5.3 demonstrates the verifiability of different methods. It is obvious that a higher noise ratio causes better verifiability. When the noise ratio is lower than 20%, it is challenging to determine if the samples $D_{u,0.2}$ are unlearned because the remaining dataset has a comparable sample $D_{u,0}$. The verifiability of both EMU and MIB has a huge improvement when the noise ratio grows more than 20%. Here, we should still notice that the verifiability of MIB is effective only for backdoored multiple samples (B-MS). MIB is infeasible in verifying the unlearning of a single backdoored sample because it is impossible to effectively backdoor the ML model with just one sample.

Impact on Efficiency. Our EMU significantly improves the unlearning evaluation efficiency as our method is not involved in the ML service model training and unlearning process. The corresponding results are shown in the fourth column in Figure 5.3. EMU consumes less than 1/3 of the computation cost compared to the MIB, which needs to participate in the original model training process for backdooring.

5.4.5 Impact of the Task Weight

We here conduct experiments to evaluate the impact of different ML task types for unlearning. And the theoretical analysis of this hypothesis from the mutual information perspective is presented in Appendix 5.3.2.

Setup. Based on the MT-IB structure, we can adjust the task weight α to control the model task preference between the autoencoding and classifying tasks, where the autoencoding task requires the most information about the dataset and the classifying task needs much less information according to the analysis in Appendix 5.3.2. The evaluation includes both the unlearned ("Not In" the remaining dataset) and not unlearned ("In" the remaining dataset) scenarios. The benchmark unlearning method tested here is SISA. We mainly present the results for single-sample unlearning in Figure 5.4.

Impact on Unlearning Effectiveness Assessment. In Figure 5.4, there are gaps in the metrics for evaluating whether unlearned samples are in or not in the remaining dataset. And the gaps expand as the task weight α increases. Higher weight task value

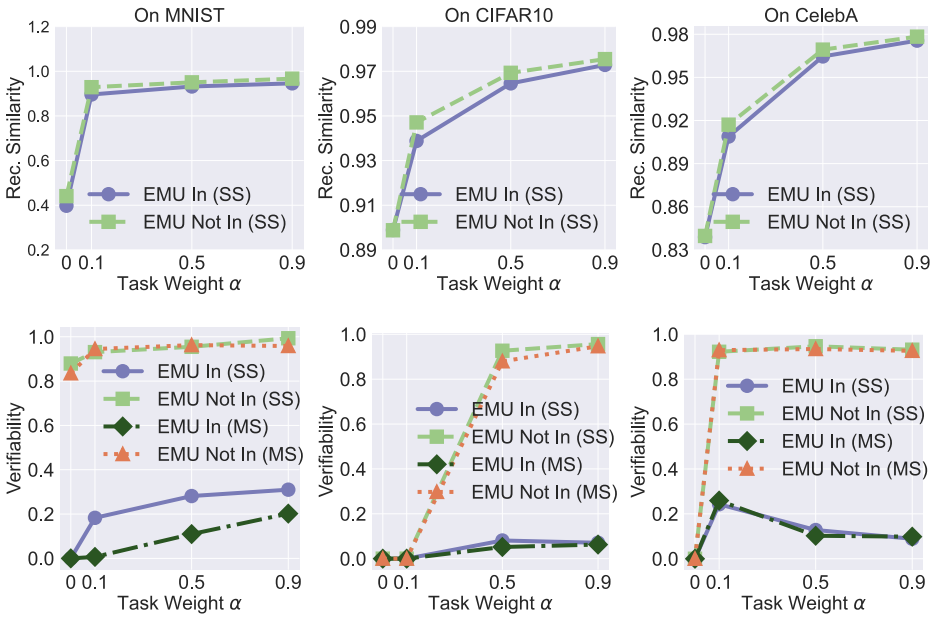


Figure 5.4: Impact about different task weight α.

provides more learned information on the model for unlearning evaluation. When $\alpha = 0$, the ML service model only contains the classifying task, and only the evaluation on MNIST is effective. Both value and gap reach max when $\alpha = 0.9$, the highest weight to the autoencoding task. It means that in this situation, the model learns the most information about the unlearned samples, and hence, the unlearning operation removes the most information from the model, which indicates the best unlearning effectiveness for the model. All the results confirm the analysis in Appendix 5.3.2: the unlearning effectiveness assessment will be better when the model task relates more information about the training dataset.

5.4.6 Impact of Unlearned Samples Size (*USS*)

We conduct experiments for evaluating the impact of *USS* in machine unlearning and present the results in Figure 5.5. We set the largest *USS* 100 in our experiments, roughly 0.2% of training datasets in MNIST and CIFAR10. In the real world, we believe 0.2% data is extensive for unlearning. The results in [9] present that 3.2 million requests for deleting URLs have been issued to Google from 2014 to 2019, which certainly constitutes less than 0.2% of the total URLs Google indexes in the 5 years.

Setup. Similar to the setting in Section 5.4.3, we evaluate the impact of different *USS* based on two scenarios, where the erased data is "Not In" the remaining dataset to

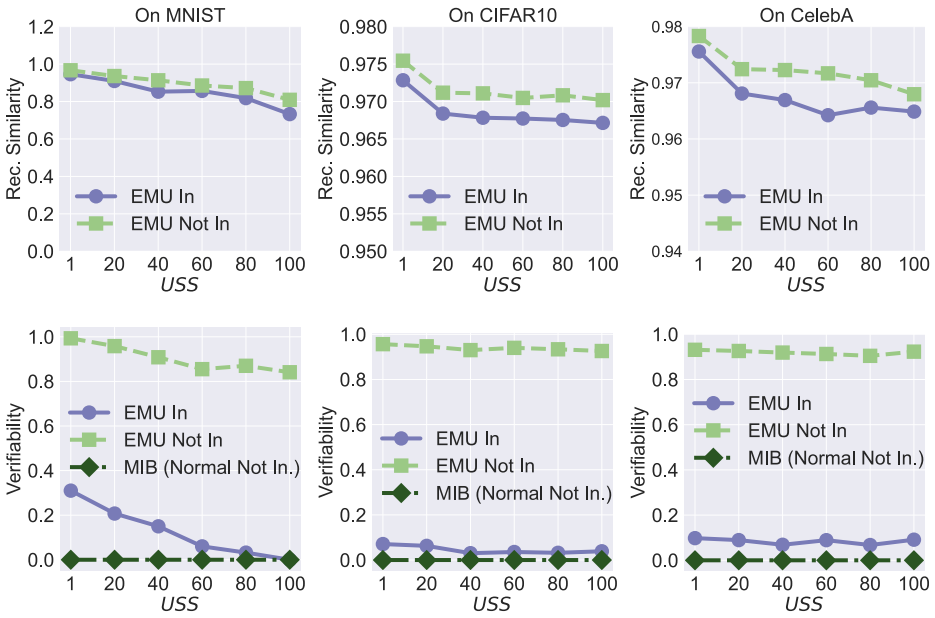


Figure 5.5: Evaluations of impact of *USS*. "In" scenario stands for keeping the erased dataset $D_{u,noi}$ in the remaining dataset for retraining. And "Not In" means removing the $D_{u,noi}$ from the remaining dataset for retraining.

represent unlearned, and the erased data is still "In" the remaining dataset to represent not unlearned. For the MIB method, we also try to verify the unlearning of genuine noised samples $D_{u,noi}$ to keep consistency with EMU. The task weight α is 0.9, and the unlearning method tested here is SISA. The results about average UE, reconstruction similarity and verifiability on three datasets are shown in Figure 5.5.

Impact on Unlearning Evaluation. In Figure 5.5, we demonstrate the evaluation of unlearning effectiveness assessment in the first row and the verifiability in the second row. There are always clear gaps between the results of unlearned ("Not In") and not unlearned ("In") situations, which demonstrates the effectiveness of EMU in evaluating machine unlearning. When $USS = 1$, EMU achieves the best performance on all datasets. All three evaluation metrics appear a huge drop when *USS* increases from 1 to $USS > 1$. Then, when *USS* continuously increases, the trend of all unlearning evaluation metrics still decreases but is not as dramatic as when going from a single sample to multiple samples. Lastly, the MIB method is infeasible in supporting the verification of unlearning genuine samples, which is presented in the black dashed line in the last row of Figure 5.5.

5.5 Summary

We investigate the unlearning evaluation problem and propose an EMU approach to evaluate how much information is unlearned based on the model difference before and after unlearning. In EMU, we first propose a model difference simulation scheme based on influence function theory to generate the unlearning model difference for evaluating efficiently. Then, we design a multi-task information bottleneck structure to enhance the scalability. Based on the influence function and information bottleneck, we provide a theoretical analysis of how the similarity between erased and remaining samples, as well as task types, impact unlearning effectiveness. The extensive experimental results also confirm the analysis.

CHAPTER 6

CRFU: Compressive Representation Forgetting Against Privacy Leakage on Machine Unlearning

Machine unlearning allows data owners to erase the impact of their specified data from trained models. Unfortunately, recent studies have shown that adversaries can recover the erased data, posing serious threats to user privacy. An effective unlearning method removes the information of the specified data from the trained model, resulting in different outputs for the same input before and after unlearning. Adversaries can exploit these output differences to conduct privacy leakage attacks, such as reconstruction and membership inference attacks. However, directly applying traditional defenses to unlearning leads to significant model utility degradation. In this chapter, we introduce a Compressive Representation Forgetting Unlearning scheme (CRFU), designed to safeguard against privacy leakage on unlearning. CRFU achieves data erasure by minimizing the mutual information between the trained compressive representation (learned through information bottleneck theory) and the erased data, thereby maximizing the distortion of data. This ensures that the model's output contains less information that adversaries can exploit. Furthermore, we introduce a remembering constraint and an unlearning rate to balance the forgetting of erased data with the preservation of previously learned knowledge, thereby reducing accuracy degradation. Theoretical analysis demonstrates that CRFU can effectively defend against privacy leakage attacks. Our experimental results show that CRFU signifi-

Table 6.1: Basic Notations of Chapter 6.

Notations	Descriptions
$D = (X, Y)$	Full training data D, including inputs X and labels Y
$D_e = (X_e, Y_e)$	The erased data D_e, including inputs X_e and labels Y_e
$D_r = (X_r, Y_r)$	The remaining data D_r, including inputs X_r and labels Y_r
Z	The compressive representation
$p(Z\|X)$	The representation posterior learned based on X
$p(Z\|X_{-X_e})$	The unlearned representation posterior unlearned based on X_e
$\hat{Y}$	The predicted approximation of the model
$\hat{Y}_{-(X_e,Y_e)}$	The unlearned approximation, it can also be written as $\hat{Y}_u$
x, z, y	The persample from X, Z, Y
θ^r	The representer of an IB model
θ^a	The approximator of an IB model
$\mathscr{L}_{rep}$	The representation loss
$\mathscr{L}_{app}$	The approximation loss of predicting
$\mathscr{L}^u_{rep}$	The representation unlearning loss
$\mathscr{L}^u_{app}$	The approximation unlearning loss
β	The Lagrange multiplier in IB
β_u	The unlearning rate of CRFU

cantly increases the reconstruction mean square error (MSE), achieving a defense effect improvement of approximately 200% against privacy reconstruction attacks with only 1.5% accuracy degradation on MNIST.

6.1 Preliminary and Threat Model

In Table 6.1, we present a concise summary of the key notations employed throughout this chapter. The full training dataset is represented as D, consisting of the input data X and labels Y. The notation Z is used to denote the compressive representation learned through the IB model, which is optimized to maximize the distortion of input data while concurrently retaining maximal relevant information about the labels. The representation posterior, formulated from X, is expressed as $p(Z|X)$. Additionally, the unlearned representation posterior, unlearning based on the specified data X_e, is denoted as $p(Z|X_{-X_e})$. The notation $\hat{Y}$ is used to utilized to represent the predictive approximation output of an IB model. After unlearning, the predictive approximation

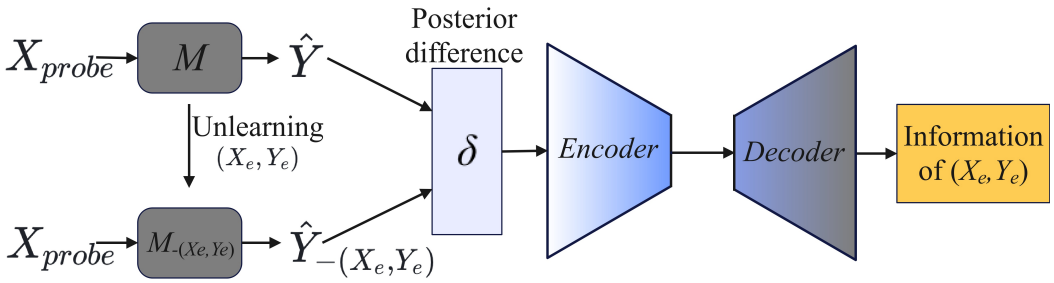


Figure 6.1: Privacy leakage attack based on the black-box model's outputs before and after unlearning.

is denoted either as $\hat{Y}_{-(X_e,Y_e)}$ or $\hat{Y}_u$. In experimental setups, we use sample variables x,z,y drawn from populations X,Z,Y. In IB algorithms, there are two types of losses: representation loss ($\mathscr{L}_{rep}$) and prediction approximation loss ($\mathscr{L}_{app}$). Here, β acts as a Lagrange multiplier in IB, balancing distortion and utility. In the CRFU process, we encounter the final unlearning loss, a combination of representation unlearning loss ($\mathscr{L}^u_{rep}$) and approximation unlearning loss ($\mathscr{L}^u_{app}$). The parameter β_u is used to manage the trade-off between completely unlearning the influence of specified samples and not entirely forgetting the previously learned representation.

6.1.1 Threat Model of Privacy Leakage Attacks on Unlearning

One effective privacy leakage attack is called model inversion, initially introduced by Fredrikson et al. [27]. It aims to deduce missing attributes of input features by interacting with a trained ML model. An example of this is the reconstruction of facial images from deep learning models, as studied in [64]. In the context of machine unlearning, researchers like Hu et al. [15, 29, 41, 106] have identified that the discrepancies in a model's outputs before and after unlearning could potentially reveal information about the deleted data. These attacks in the realm of unlearning primarily focus on exploiting these differences to compromise the erased data privacy. The main process and adversary capabilities of privacy leakage attacks on unlearning are summarized as follows.

6.1.1.1 Attacking Data Preparation

Assuming D represents the original training dataset and e is the sample a user requests to unlearn, we denote $D_r = D \setminus \{e\}$ as the remaining dataset. If there are several erased samples, we refer to the erased dataset as D_e. Machine unlearning aims to erase the influence of the specified dataset D_e from a trained ML model $\mathscr{M}(D)$, where the model

is trained based on the full training dataset D. The unlearning update is performed by executing an unlearning algorithm $\mathscr{U}$ on the current trained model using the erased dataset. More formally, given a erased dataset D_e, a trained ML model $\mathscr{M}$, and the unlearning algorithm $\mathscr{U}$, the unlearning process can be defined as $\mathscr{U} : D_e, \mathscr{M}(D) \rightarrow \mathscr{M}_u(D \backslash D_e)$, where $\mathscr{M}_u$ is the unlearned version model $\mathscr{M}$.

6.1.1.2 Adversary Goal and Capabilities

The goal of privacy leakage attacks on machine unlearning is to infer the privacy of the erased samples D_e of an unlearning update. We assume the adversary can only have black-box access to the target model. This limitation means that the adversary can interact with the model solely through queries using a specific set of data samples, known as the probing set, and subsequently receive the corresponding outputs. Furthermore, it is presumed that the adversary's local probing dataset is sourced from the same distribution as that of the target model's training dataset. Moreover, we consider that the adversary has knowledge of both the unlearning and original learning algorithms and has the capability to establish the same learning and unlearning training as the target model. This can be achieved by performing model hyperparameter stealing attacks [90]. Finally, and most importantly, we assume that the target model effectively removes the information of the erased data, and the erased dataset and the remaining dataset are disjoint.

6.1.1.3 Privacy Inference Attack Process on Unlearning

In privacy inference attacks, the adversary's initial step involves collecting varied outputs from their probe data X_{probe}. This includes obtaining the original model outputs $\hat{Y}$ before unlearning, as well as the outputs $\hat{Y}_{-(X_e,Y_e)}$ after unlearning. The key to the attack lies in the difference $\delta = \hat{Y}_{-(X_e,Y_e)} - \hat{Y}$, which is used to train the attack model. The structure of the attack model typically includes an encoder and a decoder, resembling the architecture of Variational Autoencoders (VAEs) [45]. We replicate the privacy inference attacks following the methodology outlined by [76], and the detailed attack process is depicted in Figure 6.1. The direct victims are those users who request for unlearning. Their unlearning requests prompt the ML server to execute an unlearning update, which is leveraged by this kind of attack to infer the privacy of the erased data (X_e, Y_e).

6.1.2 Implementation of Information Bottleneck

Information Bottleneck (IB) was first introduced in [88] to distort information of data inputs while maintaining the information of data targets in the representation. Traditional IB [88, 89] primarily utilizes the Blahut-Arimoto algorithm [10] to optimize the IB objective. The goal of this approach is to identify a compressed encoding distribution that is both adequate for the intended ML application and maximally distorts information from the original data. The IB objective is formulated as below,

$$\mathscr{L}_{\mathscr{IB}} = \beta I(Z;X) - I(Z;Y). \tag{6.1}$$

Here, $I(Z;X)$ represents the mutual information between the encoded representation Z and the inputs X, and $I(Z;Y)$ denotes the mutual information between Z and the outputs Y. The parameter β serves as a Lagrange multiplier, regulating the distortion ratio of X in the model.

An IB model splits the training process into two parts. Firstly, it employs a representer θ^r to compress the information from inputs X to a compact representation Z. Secondly, it employs an approximator θ^a to identify the target values based on the representation Z. The representation loss function of the representer is $\mathscr{L}_{rep} = \beta I_{\theta^r}(Z;X)$ and the corresponding approximation loss function of the approximator is $\mathscr{L}_{app} = -I_{\theta^a}(Z;Y)$. The training process follows the Markov chain $Y \rightarrow X \rightarrow Z \rightarrow \hat{Y}$, as depicted in the upper part of Figure 6.2. Prior studies, such as [1, 2, 5], have expanded on the mutual information terms and introduced two variational distributions, $q(Z)$ and $q(Y|Z)$. These distributions help in deriving an upper bound for the IB optimization loss function. The optimization of the learning process is then achieved by minimizing this upper bound. We consider a distribution $q(Z)$ where the elements in the space Z are mutually independent, expressed as $q(Z) = \prod_j q_j(z_j)$. Finally, an IB loss objective Equation (6.1) can be expanded in a per-sample way as

$$\mathscr{L} = \mathscr{L}_{app} + \mathscr{L}_{rep} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{z \backsim p_{\theta^r}(Z|x^i)}[-\log\ p_{\theta^a}(y^i|z)] + \beta \mathrm{KL}[p_{\theta^r}(Z|x^i)||\prod_{j}^{|Z|} q_j^i(z_j^i)]. \tag{6.2}$$

It's important to note that the assumed prior $q(Z)$ represents the ideal distribution, capturing minimal information about X while retaining sufficient information for the task target Y. A higher value of β leads to increased distortion, making $p(Z|X)$ more closely approximate the assumed general prior. On the contrary, a smaller β means the model pays less attention to compressing the information of X, causing the representation to contain more information about individual X.

6.2 Compressive Representation Forgetting Unlearning

6.2.1 Problem Definition

We define the problem of machine unlearning in the context of trained IB models as follows. Upon receiving an unlearning request, the full training dataset $D = (X, Y)$ will be partitioned into two distinct subsets: a smaller erased dataset $D_e = (X_e, Y_e)$ and a larger remaining dataset $D_r = (X_r, Y_r)$. These subsets are mutually exclusive, fulfilling $D = D_r \bigcup D_e$ and $D_r \bigcap D_e = \emptyset$. Then, the IB algorithm ($\mathscr{IB}$) is used to train the model $\mathscr{M}(\theta^r, \theta^a)$, consisting of a representer θ^r and an approximator θ^a. This model learns a representation Z, by minimizing $I(X;Z)$ and maximizing $I(Y;Z)$.

Typically, the machine unlearning process is initiated when a user requests the removal of their specific data samples, D_e, from the trained model $\mathscr{M}(\theta^r, \theta^a)$. In response, the server employs an unlearning algorithm $\mathscr{U}$ to erase the contribution of D_e while retaining the learned knowledge on the remaining dataset D_r. The unlearned model $\mathscr{M}_u(\theta^r, \theta^a)$ hopes to find the unlearned representation $p(Z|X_r)$, which is expected to be equal to the representation learned by retraining the model based on the remaining dataset. The problem of unlearning an IB model can be described as

Problem Statement. *Assume a gold-standard unlearned representation posterior $p(Z|X_r)$ for an IB model, retrained with $\mathscr{M}_u = \mathscr{IB}(D_r)$ based on the remaining dataset. An unlearning algorithm $\mathscr{U}$ is designed to unlearn a representation posterior $p(Z|X_{-X_e})$ of the model $\mathscr{M}_u = \mathscr{U}(D_e, \mathscr{IB}(D))$, unlearning based on the erased dataset $D_e = (X_e, Y_e)$. Then, the erased dataset-based unlearned posterior is hoped to match the remaining dataset-based retrained posterior, i.e.,*

$$p(Z|X_r) = p(Z|X_{-X_e}). \tag{6.3}$$

In this problem statement, the retrained posterior $p(Z|X_r)$ can be obtained by retraining the IB model based on the remaining dataset as

$$\min I(X_r;Z), \quad \text{s.t.} \quad I(X_r;Y_r) - I(Z;Y_r) = I(X_r;Y_r|Z) = 0. \tag{6.4}$$

Equation (6.3) means that an unlearned IB-trained model is still hoped to keep the property like a retrained model, squeezing as much information of X_r as possible from the representation Z while keeping a good utility of approximating Y_r. Retraining the

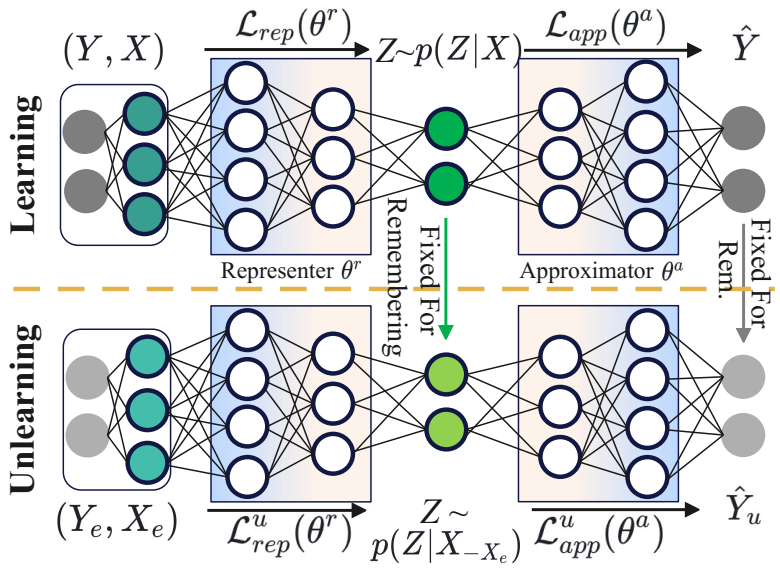


Figure 6.2: IB learning process (upper half) and CRFU unlearning process (lower half) with a fixed learned IB model (a trained and fixed upper half).

model from scratch remains a viable approach under this definition of unlearning. However, as previously discussed, retraining from scratch incurs significant storage and computational costs and has no protection to defend against reconstruction attacks on unlearning. Therefore, we propose the CRFU approach to solve this problem in the following.

6.2.2 CRFU Method

6.2.2.1 Overview of CRFU

We begin by briefly introducing the core concept of the proposed CRFU, which is illustrated in Figure 6.2. CRFU is tailored to unlearn models that were trained using the IB method (shown in the upper half in Figure 6.2). An IB method mainly learns a representation Z that compresses information of inputs while retaining sufficient information for the targets. The CRFU, the lower half in Figure 6.2, aims to erase the influence of $D_e = (X_e, Y_e)$ from Z. To achieve this goal, we minimize the mutual information $I(X_e;Z)$ and $I(Y_e;Z)$; however, directly minimizing them could result in the representation Z catastrophically forgetting everything, decreasing model utility. To mitigate the degradation of model utility, we introduce a constraint ensuring that the unlearned representation Z, derived from the distribution $p(Z|X_{-X_e})$, should be close to the representation Z obtained from the original-fixed distribution $p(Z|X)$. We deal with the

predicting approximation $\hat{Y}$ in the same way. Finally, combining the minimizing mutual information and a constraint, we formalize the losses $\mathscr{L}^u_{rep}$ and $\mathscr{L}^u_{app}$ to unlearn the trained IB model.

6.2.2.2 Theoretical Exact Compressive Representation Forgetting Unlearning

In an IB model, the objective is to learn a representation Z by minimizing the mutual information $I(X;Z)$, while simultaneously maximizing $I(Y;Z)$. In our CRFU approach, we specifically focus on minimizing $I(X_e;Z)$ for the inputs X_e designated for erasure. This is aimed at effectively eliminating the information of X_e from the previously learned representation Z. Similarly, we work on minimizing $I(Y_e;Z)$, the mutual information between the targets Y_e and the representation Z, thereby removing the information of Y_e from Z. This dual minimization ensures a comprehensive unlearning of both input and label information from the representation.

The ideal unlearned representation should contain no information about (X_e, Y_e), optimally satisfying the conditions where $I(X_e;Z) = 0$ and $I(Y_e;Z) = 0$, indicating zero mutual information between the erased data and the representation Z. However, directly minimizing $I(X_e;Z)$ and $I(Y_e;Z)$ poses a significant risk: it could lead the representation Z to inadvertently "forget" everything, including the valuable knowledge it had previously learned. Therefore, we must keep the unlearned representation Z and approximation $\hat{Y}$ remembering the knowledge learned in the former training.

To maintain the learned knowledge, we propose a solution that minimizes the Kullback–Leibler divergence (KLD) [43] between the unlearned and originally learned representations. This involves minimizing both $\mathrm{KL}[p(Z|X)||p(Z|X_{-X_e})]$ and $\mathrm{KL}[p(\hat{Y}|Z)||$ $p(\hat{Y}_{-(X_e,Y_e)}|Z)]$, ensuring that the unlearned representations remain closely aligned with the original ones. During unlearning, $p(Z|X)$ and $p(\hat{Y}|Z)$ are calculated using the trained IB model as a temporary reference model. Combining the above solutions, we formalize the unlearned representation loss as

$$\mathscr{L}^u_{rep} = I(X_e;Z) + \mathrm{KL}[p(Z|X)||p(Z|X_{-X_e})], \tag{6.5}$$

and the unlearned approximation loss as

$$\mathscr{L}^u_{app} = I(Y_e;Z) + \mathrm{KL}[p(\hat{Y}|Z)||p(\hat{Y}_{-(X_e,Y_e)}|Z)]. \tag{6.6}$$

Integrating these two loss functions, we formulate the optimization of CRFU as the proposition outlined below.

Proposition 1. *Define the CRFU loss function as*

$$\begin{aligned}\mathscr{L}^u &= \beta \cdot \mathscr{L}^u_{rep} + \mathscr{L}^u_{app} \\ &= \beta \cdot (I(X_e;Z) + KL[p(Z|X)||p(Z|X_{-X_e})]) \\ &+ I(Y_e;Z) + KL[p(\hat{Y}|Z)||p(\hat{Y}_{-(X_e,Y_e)}|Z)].\end{aligned} \tag{6.7}$$

Then, minimizing the loss in Equation (6.7) *to unlearn the erased dataset D_e from an IB model trained based on D is equivalent to retraining an IB model by minimizing $\mathscr{L}_r = \beta I(X_r;Z) - I(Y_r;Z)$ based on D_r.*

Since CRFU unlearns an IB model trained through Equation (6.2), we can combine the original IB loss and CRFU loss to prove that minimizing Equation (6.7) based on a trained IB model is equivalent to minimizing the loss of retraining an IB model on the remaining dataset. We first prove the representation loss. Given that $X = X_r \cup X_e$ and $X_r \cap X_e = \emptyset$, and considering that the data are IID, along with the conditional independence of X_r and X_e given Z, it follows that $I(X;Z) = I(X_r;Z) + I(X_e;Z)$. Since $I(X_e;Z) \geq 0$ and $\mathrm{KL}[p(Z|X)||p(Z|X_{-X_e})] \geq 0$, we can expand original and unlearned representation loss $\mathscr{L}_{rep} + \mathscr{L}^u_{rep}$ function as

$$\begin{aligned}\mathscr{L}_{rep} + \mathscr{L}^u_{rep} &= I(X;Z) + I(X_e;Z) + \mathrm{KL}[p(Z|X)||p(Z|X_{-X_e})] \\ &= I(X_r;Z) + 2I(X_e;Z) + \mathrm{KL}[p(Z|X)||p(Z|X - X_e)] \\ &\geq I(X_r;Z).\end{aligned} \tag{6.8}$$

It is clear that the sum of the representation loss $\mathscr{L}_{rep}$ and the unlearned representation loss $\mathscr{L}^u_{rep}$ serves as an upper bound for the mutual information $I(X_r;Z)$. Minimizing this upper bound during both original training and unlearning is equivalent to minimizing $I(X_r;Z)$ itself.

Similarly, for the original and unlearned approximation loss function $\mathscr{L}_{app} + \mathscr{L}^u_{app}$, we can obtain the expanded approximation as

$$\begin{aligned}\mathscr{L}_{app} + \mathscr{L}^u_{app} &= -I(Y;Z) + I(Y_e;Z) + \mathrm{KL}[p(\hat{Y}|Z)||p(\hat{Y}_{-(X_e,Y_e)}|Z)] \\ &= -(I(Y;Z) - I(Y_e;Z)) + \mathrm{KL}[p(\hat{Y}|Z)||p(\hat{Y}_{-(X_e,Y_e)}|Z)] \\ &= -I(Y_r;Z) + \mathrm{KL}[p(\hat{Y}|Z)||p(\hat{Y}_{-(X_e,Y_e)}|Z)] \\ &\geq -I(Y_r;Z).\end{aligned} \tag{6.9}$$

Since $Y = Y_r \cup Y_e$ and $Y_r \cap Y_e = \emptyset$ and the data are IID, it follows that $I(Y_r;Y_e) = 0$ and $H(Y) = H(Y_r,Y_e) = H(Y_r) + H(Y_e) - I(Y_r;Y_e) = H(Y_r) + H(Y_e)$. We can similarly achieve $H(Y_r|Z)$. And since $I(Y_r;Z) = H(Y_r) - H(Y_r|Z)$, thus Equation (6.9) holds. Minimizing the

original and unlearned approximation as Equation (6.9) is the upper bound of $-I(Y_r;Z)$ in retraining. Therefore, minimizing the CRFU loss function based on a trained IB model is equivalent to minimizing the loss function of retraining an IB model based on the remaining dataset.

6.2.2.3 Variational Optimization Method for CRFU

Optimizing the CRFU loss function Equation (6.7) in a deep learning scenario with a huge amount of training data is challenging. To address this issue, we present a variational method to make the CRFU loss function calculatable in deep learning. The unlearned representation loss can be expanded as

$$\begin{aligned}\mathscr{L}^u_{rep} &= I(X_e;Z) + \mathrm{KL}[p(Z|X)||p(Z|X_{-X_e})] \\ &= \mathrm{KL}[p_{\theta^r}(Z|X_e)||p_{\theta^r}(Z)] + \mathrm{KL}[p_{\theta^r_{fix}}(Z|X)||p_{\theta^r}(Z|X_{-X_e})].\end{aligned} \tag{6.10}$$

Generally, minimizing this loss is intractable due to the complexity involved in computing the KL term. This computation requires knowledge of the marginal distribution Z and $p_{\theta^r}(Z) = \int dx\ p(z|x)p(x)$, which is not easily obtainable. To address this issue, we introduce $q(Z)$ as a variational approximation to this marginal as the variational IB learning in [1, 2]. Since $\mathrm{KL}[p(Z)||q(Z)] \geq 0 \Longrightarrow \int dz p(z)\log p(z) \geq \int dz p(z)\log q(z)$, we have the following upper bound of Equation (6.10):

(6.11)

$$\begin{aligned}\mathscr{L}^u_{rep} &= I(X_e;Z) + \mathrm{KL}[p(Z|X)||p(Z|X_{-X_e})] \\ &\leq \underbrace{\mathrm{KL}[p_{\theta^r}(Z|X_e)||q(Z)]}_{\text{Forgetting the erased inputs from the representation}} + \underbrace{\mathrm{KL}[p_{\theta^r_{fix}}(Z|X)||p_{\theta^r}(Z|X_{-X_e})]}_{\text{Remembering the original learned representation}}.\end{aligned}$$

This unlearned representation loss can be explained as forgetting the impact of the erased data's inputs (first term) and remembering the knowledge of original full training inputs (second term). The proof of the correctness of Equation (6.11) is similar to the proof of Eq. (14) in [2]. Hence, we omit the detailed proof here.

Following the solutions to $\mathscr{L}_{app}$ in [1, 2], the unlearned approximation loss function can be optimized in a similar way as

(6.12)

$$\begin{aligned}\mathscr{L}^u_{app} &= I(Y_e;Z) + \mathrm{KL}[p(\hat{Y}|Z)||p(\hat{Y}_{-(X_e,Y_e)}|Z)] \\ &\simeq \underbrace{\int dz\ p(Z|X_{-X_e})\log\ p_{\theta^a}(Y_e|Z)}_{\text{Forgetting the erased targets from the approximation}} + \underbrace{\mathrm{KL}[p_{\theta^a_{fix}}(\hat{Y}|Z)||p_{\theta^a}(\hat{Y}_{-(X_e,Y_e)}|Z)]}_{\text{Remembering the original learned approximation}}\end{aligned}$$

The computable optimization loss Equation (6.12) represents a balance between completely unlearning the information of the erased labels Y_e (first term) and not entirely

Algorithm 3: Compressive Representation Forgetting Unlearning.

Input: The trained model $\mathscr{M}(\theta^r, \theta^a)$, the erased dataset (X_e, Y_e) and training epochs E

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

Establish the fixed temporary model: $\mathscr{M}_{fix}(\theta^r_{fix}, \theta^a_{fix}) \leftarrow \mathscr{M}(\theta^r, \theta^a)$

for E *epochs* **do**

  Draw a minibatch of m samples $\{(x^i, y^i)\}_{i=1}^m$ from erased dataset $D_e = (X_e, Y_e)$;

  Generate $z^i \backsim p_{\theta^r}(\cdot|x^i)$;

  Compute the loss function Equation (6.11) for unlearning representation on a per-sample basis $\mathscr{L}^u_{rep}(\beta_u) = \frac{\beta_u}{m}\sum_{i=1}^m \mathrm{KL}[p_{\theta^r}(Z|x^i)||\prod_j^{|Z|} q^i_j(z^i_j)] + \frac{1}{m}\sum_{i=1}^m \mathrm{KL}[p_{\theta^r_{fix}}(Z|x^i)||p_{\theta^r}(Z|x^i)]$

  Compute the loss function Equation (6.12) for unlearning approximation on a per-sample basis $\mathscr{L}^u_{app}(\beta_u) = \beta_u \cdot \frac{1}{m}\sum_{i=1}^m \log\ p_{\theta^a}(y^i|z^i) + \frac{1}{m}\sum_{i=1}^m \mathrm{KL}[p_{\theta^a_{fix}}(\hat{y^i}|z^i)||p_{\theta^a}(\hat{y^i}_{-(X_e,Y_e)}|z^i)]$

  Update the model based on the gradients of the integrated loss functions Equation (6.13) as $(\theta^r, \theta^a) \leftarrow (\theta^r, \theta^a) - \eta\nabla_{(\theta^r,\theta^a)}(\beta \cdot \mathscr{L}^u_{rep}(\beta_u) + \mathscr{L}^u_{app}(\beta_u))$

Return $\mathscr{M}_u(\theta^r, \theta^a)$;

forgetting the originally learned information of Y in representation Z (second term). Combining Equations (6.11) and (6.12) and optimizing them together is the CRFU loss Equation (6.7) that we proposed in proposition 1. When minimizing this loss, we should notice that the second term in Equations (6.11) and (6.12) is calculated based on the original full training dataset; however, storing all original datasets and training it again is impractical. A simple way we used is fixing the trained model (i.e., the model before unlearning) as a temp model that is specially used to calculate the first term but only based on the erased dataset.

6.2.2.4 An Unlearning Rate for CRFU

To make the unlearning method adaptive to different tasks, we introduced an unlearning rate parameter, β_u, which serves to regulate the extent of unlearning during both the representation and approximation unlearning process. Specifically, we add an unlearning rate parameter β_u before the forgetting terms in Equations (6.11) and (6.12) to optimize the balance between forgetting the information of erased samples and remembering the representation learned before. Adjusting the unlearning rate β_u will

also impact the unlearning speed. The final equation can be described as

$$
\begin{aligned}
\mathscr{L}^u &= \beta \cdot \mathscr{L}^u_{rep}(\beta_u) + \mathscr{L}^u_{app}(\beta_u) \\
&\simeq \beta \cdot (\beta_u \cdot \mathrm{KL}[p_{\theta^r}(Z|X_e)||q(Z)] + \mathrm{KL}[p_{\theta^r}(Z|X)||p_{\theta^r_{fix}}(Z|X_{-X_e})]) \\
&+ \beta_u \cdot \int dz\ p(Z|X_{-X_e}) \log\ p_{\theta^a}(Y_e|Z) + \mathrm{KL}[p_{\theta^a_{fix}}(\hat{Y}|Z)||p_{\theta^a}(\hat{Y}_{-(X_e,Y_e)}|Z)]
\end{aligned} \tag{6.13}
$$

We present a pseudocode of CRFU in Algorithm 3. At the beginning of executing representation forgetting unlearning, we first prepare a trained IB model $\mathscr{M}(\theta^r, \theta^a)$, the erased dataset $D_e = (X_e, Y_e)$, and training epochs E. Then, we fix the original optimal model $\mathscr{M}_{fix}(\theta^r_{fix}, \theta^a_{fix}) \leftarrow \mathscr{M}(\theta^r, \theta^a)$ as a temp model for the later unlearned representation and approximation loss calculation, as shown in Line 1 of Algorithm 3. When executing unlearning training, lines 2 to 7, we draw a minibatch of m data samples $\{(x_i, y_i)\}_{i=1}^m$ from D_e. Using the representer θ^r, we generate the corresponding representations z_i for these samples. The unlearning loss function is then calculated as per Equations (6.11) and (6.12). Following this, we update the parameters of the representer and approximator (θ^r, θ^a) in the IB model in accordance with Equation (6.13) at line 7.

6.3 Theoretical Analysis of Privacy Leakage Defense

In this section, we give a theoretical explanation of why our proposed CRFU can effectively defend against privacy leakage attacks of unlearning in a black-box ML setting.

6.3.1 Reasons behind Effective Privacy Leakage Attacks

Before discussing why our proposed CRFU is effective against privacy leakage attacks, we first explain why the existing privacy inference attacks can effectively recover users' privacy. Similar to most model inference attacks, the privacy inference attack on unlearning aims to recover training samples by analyzing a black-box ML model's outputs. These attacks focus on differences between the outputs of the original and updated unlearning models to reconstruct the privacy of the deleted data in this unlearning process, as discussed in [15, 29, 41]. The effectiveness of such attacks lies in the requirement that the unlearning mechanism must ensure the unlearned model forgets the specified samples. Therefore, effective unlearning, which erased the information of specified data from the model, will result in different outputs for the same inputs before and after unlearning. Attackers can exploit private information from erased data by comparing the

model's different outputs before and after unlearning. They mimic the changes between the original and unlearned models using an IID auxiliary dataset, thereby training a model to infer the privacy of the erased data.

6.3.2 How CRFU Defends against Privacy Leakage Attacks

CRFU can effectively defend against such privacy leakage attacks because both learning and unlearning are based on the IB framework, which discards maximized information of inputs X of the bottleneck Z, leaving little information in the model output for adversaries to infer. Specifically, CRFU implements the data erasure of inputs X_e by further minimizing $I(X_e;Z)$ based on the minimized $I(X;Z)$. Ideally, we denote the information remaining after unlearning as $I(X_r;Z)$. Therefore, the maximum information that an adversary can infer from the different representations Z before and after unlearning is $I(X;Z) - I(X_r;Z) = I(X_e;Z)$. During the learning process, the $I(X_e;Z)$ is minimized with a constraint of maintaining enough information for targets, i.e., $I(X_e;Z) \geq I(Y_e;Z)$. Therefore, the ideal protection of CRFU is $I(X_e;Z) = I(Y_e;Z)$. Attackers are only able to infer information about the labels, but they cannot reconstruct the samples that have been erased.

It can also be explained from the perspective of a Markov chain, which is applied to both the model learning and unlearning processes. In the IB model training, the Markov chain is established as $Y \rightarrow X \rightarrow Z \rightarrow \hat{Y}$. For the CRFU training, the corresponding chain is $Y_e \rightarrow X_e \rightarrow Z \rightarrow \hat{Y}_{-(X_e,Y_e)}$. It ensures that attackers cannot infer more information from $\hat{Y}$ than Z in a black-box ML scenario because $\hat{Y}$ is derived from Z. Due to the Markov chain principle, once information is lost in one layer, it cannot be regained in subsequent layers. This process can be described as

$$\begin{cases} I(X;Y) \geq I(Z;Y) \geq I(\hat{Y},Y) \\ I(X_e;Y_e) \geq I(Z;Y_e) \geq I(\hat{Y}_{-(X_e,Y_e)},Y_e) \end{cases} \tag{6.14}$$

Therefore, for an IB model that has undergone unlearning through CRFU, the upper bound of privacy inference attacks based on the different outputs is the reconstruction capability on the different representation Z. This is because the outputs of the original trained model ($\hat{Y}$) and the unlearned model ($\hat{Y}_{-(X_e,Y_e)}$) are derived from the representations before and after unlearning, respectively. The information that the adversary can infer will be no more than $I(Y_e;Z)$.

6.3.3 β-Compression Defense of CRFU

We provide an example of the representation term's process using the Gaussian distribution case, which is widely employed in many studies [2, 51]. Let the assumed prior $q(Z) = \mathcal{N}(Z; 0, \mathbf{I})$ and the posterior $p(Z|X) = \mathcal{N}(Z; \mu^i, \sigma^i)$ are Gaussian. Let J be the dimensionality of Z, μ^i and σ^i are the mean and standard deviation output by the representer θ^r at datapoint x^i, and let μ^i_j and σ^i_j denote the j-th element of these vectors. Then we have

$$\mathrm{KL}[p_{\theta^r}(Z|x^i)||q_{\theta^r}(Z)] = \frac{1}{2}\sum_{j=1}^{J}((\mu^i_j)^2 + (\sigma^i_j)^2 - \log((\sigma^i_j)^2) - 1) \tag{6.15}$$

The KL divergence of Equation (6.15) is one optimizing term of Equations (6.2) and (6.13). And the mean μ^i and s.d. σ^i are determined by x^i and the representer parameters θ^r. The representer network θ^r transforms the input data x^i into the parameters of a complex, high-dimensional joint Gaussian distribution. Specifically, for each input x^i, the representer outputs a mean vector $\mu^i_J = \{\mu^i_1, \mu^i_2, ..., \mu^i_j\}$ and a log-variance vector $\log((\sigma^i_J)^2) = \{\log((\sigma^i_1)^2), \log((\sigma^i_2)^2), ..., \log((\sigma^i_j)^2)\}$. Here, for each input, the representer θ^r will output J joint Gaussian distribution. These parameters describe a high-dimensional Gaussian distribution for the data point x^i as

$$\mu^i_J, \log((\sigma^i_J)^2) = \theta^r(x^i). \tag{6.16}$$

Then, we can calculate the representation Z^i_J using the reparameterization trick. We sample a latent vector Z^i_j from this distribution:

$$Z^i_J = \mu^i_J + \sigma^i_J \cdot \epsilon, \tag{6.17}$$

where $\epsilon \sim \mathcal{N}(0, I)$. When Equation (6.15) is minimized, such as in the case of $\mathrm{KL}[p_{\theta^r}(Z|x^i) ||q_{\theta^r}(Z)] = 0$, the learned representation posterior $p(Z|x^i)$ becomes identical to the assumed prior, $\mathcal{N}(Z; 0, \mathbf{I})$. In this scenario, the representation Z loses all information about the sample $x^i \in X$ and $x^i \in X_e$, limiting attackers to inferring only the information of the general prior, $\mathcal{N}(Z; 0, \mathbf{I})$. Greater distortion thus offers a stronger defense against privacy inference attacks.

However, to ensure model utility, there is a constraint that Z must contain sufficient information about Y. A distortion rate β is introduced in Equations (6.2) and (6.13) to balance the information compression and the model utility. We can define the protection capability of CRFU as follows:

Definition 3. *CRFU achieves the β-**Compression Defense** against the privacy leakage attacks on unlearning if the original IB model is trained using Equation* (6.2) *and CRFU unlearns the trained IB model using Equation* (6.13)*, where Equations* (6.2) *and* (6.13) *are optimized using the same compressive parameter β.*

Since the compression ratio is controlled by β in Equations (6.2) and (6.13), an increasing β leads to greater information distortion of X and a reduced privacy inference effect from Z while concurrently diminishing the model utility about Y. To balance the tradeoff between prediction utility and defense against privacy leakage attacks, selecting a suitable β is crucial. Our upcoming experiments will further assess the defense effectiveness against privacy leakage attacks at varying levels of β.

6.4 Performance Evaluation

6.4.1 Experiment Setup

Datasets. We assess the effectiveness of the proposed Compressive Representation Forgetting Unlearning (CRFU) method using four benchmark datasets: MNIST, Fashion-MNIST [102], CIFAR10 [48], and STL-10 [18]. The four datasets are benchmark datasets for image classification tasks, which cover a wide range of object categories with different learning complexities.

Models. In our experiments, we utilize two model architectures of different sizes, a 5-layer multi-layer perceptron (MLP) with ReLU activations and ResNet-18, to construct the IB model. Specifically, we employ two 5-layer MLP models, one as the representer and one as the approximator, to construct the IB models trained on MNIST and Fashion-MNIST. We employ one ResNet-18 as the representer and one 5-layer multi-layer MLP as the approximator to construct the IB models trained on CIFAR10 and STL-10.

For experimental simplicity, we set a consistent minibatch size of $m = 20$. We set the learning rate $\eta = 0.001$ on both the MNIST and Fashion MNIST datasets. For experiments conducted on the CIFAR10 and STL-10 datasets, we employ the learning rate to $\eta = 0.0005$. In the performance evaluation for defense, we set the unlearning rate $\beta_u = 0.1$. In the evaluation of unlearning utility, we set the distortion rate $\beta = 0.001$ on MNIST and Fashion MNIST and $\beta = 0.0001$ on CIFAR10 and STL-10. All methods were implemented using PyTorch and tested on a computing cluster equipped with four NVIDIA 1080ti GPUs.

Evaluation Metrics for Attacking Methods and Unlearning Benchmarks. Our evaluation focuses on two aspects: the defense against privacy leakage attacks and the utility of the unlearned model. From the perspective of defense capabilities, we test all unlearning methods against the state-of-the-art attack implementations as described in [15, 76], including the reconstruction attack and the membership inference attack. For an effective attack, we implement the reconstruction and membership inference attacks directly on the representation Z, representing the upper bound of such attacks in CRFU, based on the outputs $\hat{Y}_{-(X_e,Y_e)}$. As discussed in Chapter 6.3, an attacker cannot infer more information from $\hat{Y}$ and $\hat{Y}_{-(X_e,Y_e)}$ than from Z before and after unlearning, since both the original model and CRFU base their outputs on Z. Following the approach in [76], we measure the quality of reconstruction using mean square error (MSE). We rely on the traditional AUC metric to measure the absolute performance of the membership inference according to [15].

From the perspective of unlearned model utility, we compare the effectiveness and efficiency of CRFU and state-of-the-art two main kinds of approximate unlearning methods, including Hessian-matrix-based unlearning (HBU) [36, 78] and variational bayesian unlearning (VBU) [28, 67]. To rigorously assess the effectiveness of unlearning methods, we adopt a widely-used technique as outlined by Hu et al. [40], which involves embedding backdoor triggers into the samples that are to be erased during the initial training of the original IB model. The objective of all unlearning methods is to eliminate the influence of these embedded backdoors from the trained models. After unlearning, we evaluate the success of these methods by checking whether the backdoor still poses a threat to the unlearned model. The effectiveness of the unlearning methods is gauged in two ways: firstly, by measuring the model's accuracy on a test dataset, and secondly, by assessing the backdoor accuracy on the erased dataset [40]. Additionally, we analyze the efficiency of unlearning by timing the model's run, computed as the product of the per-batch training time and the total number of training epochs.

6.4.2 Evaluations of Defense Capability

As analyzed in Chapter 6.3, CRFU performs unlearning based on a trained IB model. Increasing the value of β during training leads to a learned representation that more closely approximates the general prior. A smaller KLD, KL$[p(Z|X)||q(Z)]$, implies better distortion quality. However, increased distortion can compromise prediction accuracy. To explore the relationship between β and its defense effectiveness, we conducted experiments with varying β values in both the original model and the CRFU on the

Table 6.2: Reconstructing quality of different β.

Dataset	Original Images	$\beta = 0$ (Normal unl. models HBU and VBU)	$\beta = 0.001$	$\beta = 0.01$	$\beta = 0.1$	$\beta = 1$
MNIST						
Reconstruction MSE	-	231.31	252.23	312.87	478.76	668.07
Inference AUC	-	0.674	0.653	0.573	0.526	0.515
KLD to Prior $q(Z)$	-	12.39	5.00	1.45	0.37	0.14
Accuracy	-	97.45%	97.32%	96.99%	96.45%	95.75%
Fashion MNIST						
Reconstruction MSE	-	220.31	240.41	284.79	386.08	561.73
Inference AUC	-	0.656	0.638	0.579	0.519	0.515
KLD to Prior $q(Z)$	-	12.23	5.77	1.78	0.33	0.13
Accuracy	-	87.95%	87.72%	87.29%	87.15%	86.15%
CIFAR10						
Reconstruction MSE	-	1174.33	1180.84	1188.57	1267.78	1550.24
Inference AUC	-	0.821	0.810	0.769	0.669	0.571
KLD to Prior $q(Z)$	-	9.09	3.75	1.76	0.53	0.11
Accuracy	-	84.75%	84.75%	84.00%	83.96%	82.95%

MNIST, Fashion MNIST, and CIFAR10 datasets. For clarity of illustration, we have not added backdoors in the erased data during this experiment. We set a fixed unlearning rate $\beta_u = 0.1$, and only 9 erased samples of the full training data. To facilitate the experimental process easily, we carried out the reconstruction and membership inference attack on the representation Z, marking the highest level of attack an adversary can achieve with $\hat{Y}$ and $\hat{Y}_{-(X_e,Y_e)}$. The results showcasing the reconstruction effect and model utility across different β values are presented in Table 6.2.

In our study on the MNIST dataset, as detailed in rows 2 to 6 of Table 6.2, we observed that the quality of reconstruction and the AUC of membership inference is optimal when $\beta = 0$. Under this condition, the method can recover the highest level of detail from the original images. This is reflected in the recorded MSE for reconstruction, which is the lowest observed value at 231.31. At the same time, the inference AUC is the highest at 0.674. When $\beta = 0$, the model will not distort the information of X from the representation Z, making the model similar to normal unlearning methods (HBU and VBU) that have not considered information compression.

The reconstruction MSE rises, and the inference AUC decreases, with increasing β, aligning with our prior analysis that a larger β results in greater distortion. Conse-

quently, Z contains less information about X_e. As β increases to 1 in our experiments, the KLD between $p(Z|X)$ and $q(Z)$ attains its minimal value. This indicates that the representation Z becomes most akin to the general prior, achieving an optimal level of distortion. This heightened distortion renders the reconstruction process more challenging. At such a distortion ratio, attackers are likely to reconstruct only a blurry image from Z, which lacks much of the detailed information present in the original images. Correspondingly, the MSE for reconstruction is at its highest at this point, registering at 668.07. Conversely, as β increases, there is a slight decrease in prediction accuracy. Specifically, the accuracy drops from 97.45% when $\beta = 0$ to 95.75% when $\beta = 1$.

In our experiments with the Fashion MNIST dataset, detailed in rows 7 to 11 of Table 6.2, we noted trends akin to those observed in the MNIST dataset. Specifically, as β increases, the model's learned representation increasingly resembles the assumed general prior. This similarity makes the tasks of reconstructing the original images and inferring membership more challenging. Specifically, the increase in β from 0 to 1 leads to a rise in the difficulty of reconstruction and membership inference. The images reconstructed under higher β values tend to lose more detailed information from the original images. Correspondingly, the MSE for reconstruction increases from 220.31 to 561.73, and the AUC decreases from 0.656 to 0.515. At the same time, the KLD to the assumed prior decreases from 12.23 to 0.13, indicating that the representation is becoming more similar to the general prior. Alongside these changes, the prediction accuracy of the model experiences a minor decrease, moving from 87.95% at $\beta = 0$ to 86.15% at $\beta = 1$.

The defense evaluations on the CIFAR10 dataset, as detailed in row 12 of Table 6.2, demonstrate the challenges of CIFAR10 image recovery. Given the dataset's complexity, unlike MNIST and Fashion MNIST, the intricate nature of CIFAR10 images makes the visual assessment of reconstruction quality less straightforward. However, we can derive conclusions similar to those from the other datasets by examining the reconstruction MSE, membership inference AUC, and model accuracy. When $\beta = 0$, attackers are able to reconstruct images from CIFAR10 with an MSE of 1174.33. As β increases, the difficulty of reconstruction also rises, with the MSE climbing to 1550.24 at $\beta = 1$. This trend indicates that a larger β leads to the discarding of more information from the model's representation, albeit at the cost of a slight decrease in prediction accuracy. Specifically, on the CIFAR10 dataset, the accuracy decreases from 84.75% at $\beta = 0$ to 82.95% at $\beta = 1$. At the same time, more information distorted hinders membership inference attacks, resulting in decreasing inference AUC from 0.821 to 0.571.

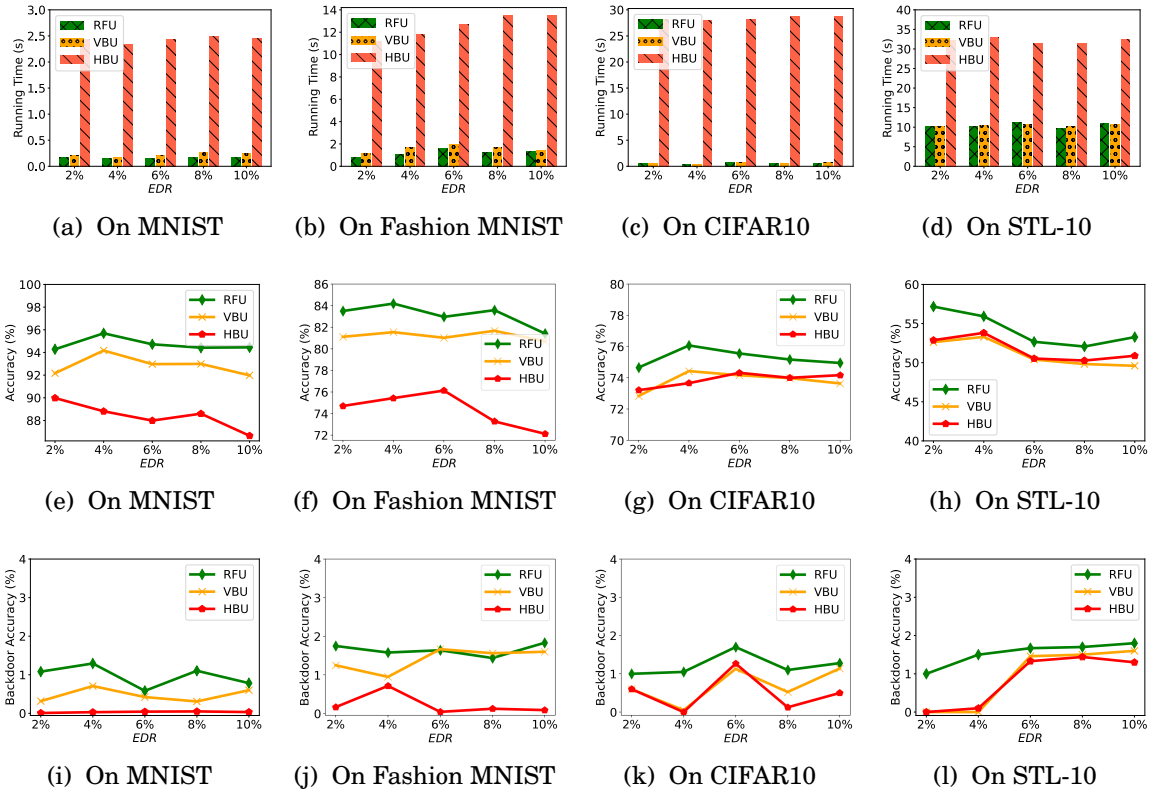
Table 6.3: Overall Unlearning Effectiveness and Efficiency Evaluation.

MNIST	$EDR = 6\%$, Rep.: MLP, App.: MLP				
	Origin	HBU	VBU	CRFU	Retrain
Running Time (s)	44	2.42	0.20	**0.15**	41.36
Acc. on test dataset	97.6%	87.99%	92.97%	**94.72%**	97.63%
Backdoor Acc.	100%	**0.04%**	0.42%	0.58%	0.08%
Fashion MNIST	$EDR = 6\%$, Rep.: MLP, App.: MLP				
	Origin	HBU	VBU	CRFU	Retrain
Running Time (s)	190.8	12.67	1.99	**1.56**	179.35
Acc. on test dataset	88.4%	76.13%	81.02%	**82.96%**	88.6%
Backdoor Acc.	100%	**0.04%**	1.66%	1.63%	0.08%
CIFAR10	$EDR = 6\%$, Rep.: Resnet18, App.: MLP				
	Origin	HBU	VBU	CRFU	Retrain
Running Time (s)	552	28.21	0.77	**0.72**	518.88
Acc. on test dataset	81.16%	74.32%	74.16%	**75.17%**	86.65%
Backdoor Acc.	99.83%	1.27%	**1.13%**	1.7%	0.03%
STL-10	$EDR = 6\%$, Rep.: Resnet18, App.: MLP				
	Origin	HBU	VBU	CRFU	Retrain
Running Time (s)	497	31.44	**10.60**	11.24	467
Acc. on test dataset	63.38%	50.53%	50.40%	**52.67%**	61.65%
Backdoor Acc.	99.93%	**1.33%**	1.46%	1.67%	0.01%

In short, CRFU can achieve a better reconstruction and membership inference attack defense when the distortion ratio β is larger. Simultaneously, as β increases, the accuracy drops a little. However, compared with the improvement of the defense effect on the reconstruction and membership inference attacks, we consider this accuracy reduction insignificant in the experimental parameters range.

6.4.3 Evaluations of Model Utility

After demonstrating the defense effect of CRFU against reconstruction attacks on unlearning, we compare the unlearning performance of CRFU and other existing state-of-the-art HBU and VBU methods. As introduced at the experiment setup, we refer to [40] to add backdoors in the erased dataset to test the unlearning effect. Although all methods successfully unlearn the erased dataset, continued unlearning training will deteriorate the model's utility. Therefore, we set a threshold of 2% backdoor accuracy on the erased data, halting the model training upon reaching this threshold. This backdoor



(a) On MNIST (b) On Fashion MNIST (c) On CIFAR10 (d) On STL-10

(e) On MNIST (f) On Fashion MNIST (g) On CIFAR10 (h) On STL-10

(i) On MNIST (j) On Fashion MNIST (k) On CIFAR10 (l) On STL-10

Figure 6.3: Performance of different unlearning methods of various EDR.

accuracy threshold guarantees the unlearning effect much better than randomly selecting, which is 10% backdoor accuracy. Moreover, we set fixed β as introduced before and set $\beta_u = 0.1$ here.

We evaluate the utility of different unlearning methods from two aspects: effectiveness and efficiency. Overall results on four datasets, MNIST, Fashion MNIST, CIFAR10, and STL-10, are shown in Table 6.3, where the erased data ratio (EDR) is 6% of the training data and unlearning rate $\beta_u = 0.1$. Since these models are backdoored, accuracy on the test dataset may decrease, especially on CIFAR10, only around 81.16% here. All unlearning methods successfully diminish the impact of backdoored data from the trained model lower than 2% backdoor accuracy. HBU achieves the best removal effect on MNIST, Fashion MNIST, and STL-10, but it consumes the longest running time and significantly degrades the model's accuracy. CRFU achieves the best performance in both running time and accuracy on the test dataset most of the time. Though CRFU has not achieved the best backdoor removal effect, it effectively implements the unlearning of backdoored samples, reducing the backdoor accuracy to lower than 2%. Detailed comparisons of different unlearning methods on MNIST, Fashion MNIST, CIFAR10, and

STL-10 are demonstrated in Figure 6.3 and will be introduced in the following.

6.4.3.1 Efficiency of Unlearning

We evaluate the efficiency through the running time of three unlearning methods: CRFU, VBU and HBU. Figures 6.3a to 6.3d show the results of the running time of all compared methods on MNIST, Fashion MNIST, CIFAR10, and STL-10. When EDR is larger, it backdoors the model deeper and takes increasing time to remove the influence of these injected backdoors. It is proven on all three datasets that the running time has a slight increase as the EDR increases. CRFU consumes a similar running time as VBU, both achieving a speedup exceeding $10\times$ when compared to HBU on MNIST, Fashion MNIST, and CIFAR10. HBU demands the most running time due to its requirement to compute the Hessian matrix using the remaining dataset to estimate the contribution of the erased data, which consumes much more time than directly unlearning based on the erased dataset. Even though HBU consumes the highest running time, it still can achieve a huge speedup compared to retraining from scratch.

6.4.3.2 Effectiveness of Unlearning

The effectiveness of unlearning is assessed based on two metrics: the model's accuracy on the test dataset and the backdoor accuracy on the erased dataset. These are presented in Figures 6.3e to 6.3h for test dataset accuracy, and Figures 6.3i to 6.3l for backdoor accuracy, respectively. First, the accuracy of the unlearned models decreases slightly as EDR increases on all three datasets. The only exception is when $EDR = 2\%$, unlearning this small backdoored dataset has a huge accuracy degradation than bigger EDR. Through extensive experimentation, we discovered that while $EDR = 2\%$ backdoored samples can successfully implant a backdoor in the model, they do not embed it as deeply as a larger EDR would. As a result of this shallow backdooring, unlearning the backdoor trigger using these erased samples (most features are valid) tends to damage the model more significantly. In this context, RFU achieves the best accuracy maintenance compared with the other two state-of-the-art unlearning methods, improving around 2% accuracy.

Second, from the perspective of backdoor accuracy of the unlearned model, all methods effectively eliminate the impact of backdoored data, making the backdoor accuracy lower than 2%, which is much lower than randomly selecting 10%. Though HBU performs the worst in accuracy maintenance, it achieves the best backdoor removal in most

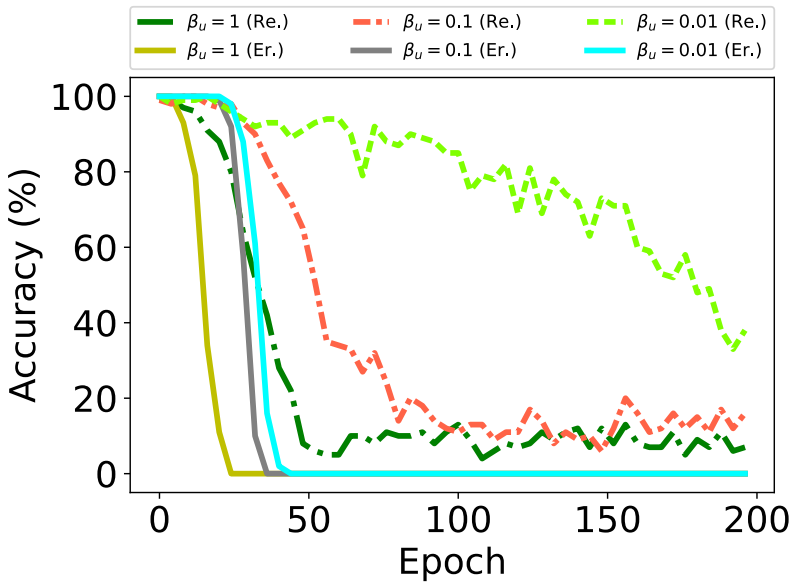


Figure 6.4: The variations in accuracy on the remaining (abbreviated as Re.) dataset and backdoor accuracy on the erased (abbreviated as Er.) dataset during unlearning of various unlearning rate β_u on MNIST.

of the results on the four datasets. CRFU achieves a similar backdoor removal effect as VBU on Fashion MNIST but slightly higher than VBU on CIFAR10 and STL-10.

6.4.4 Influence of the Proposed Unlearning Rate

In this section, we evaluate the influence of the introduced unlearning rate. Since only our method has this parameter, we directly show the training progress of our method of different unlearning rate β_u to analyze the influence of β_u in CRFU. Specifically, we show the changes in model accuracy and backdoor accuracy in each CRFU training epoch on the remaining (abbreviated as Re.) dataset and the erased (abbreviated as Er.) dataset, respectively. To better illustrate the training process, we continue the model unlearning even when reaching the 2% backdoor accuracy threshold set before. Moreover, we set $\beta = 0.001$ for MNIST and Fashion MNIST and $\beta = 0.0001$ for CIFAR10 and set $EDR = 6\%$.

Figure 6.4 illustrates the results of variations in accuracy and backdoor accuracy, using different β_u, from 0.01 to 1, on MNIST. When β_u is large, CRFU unlearns the backdoored samples faster than β_u is small because the larger unlearning rate β_u means CRFU updating with a larger content on the forgetting terms of Equations (6.11) and (6.12). On CIFAR10, when $\beta_u = 0.01$, the accuracy of the model on the remaining

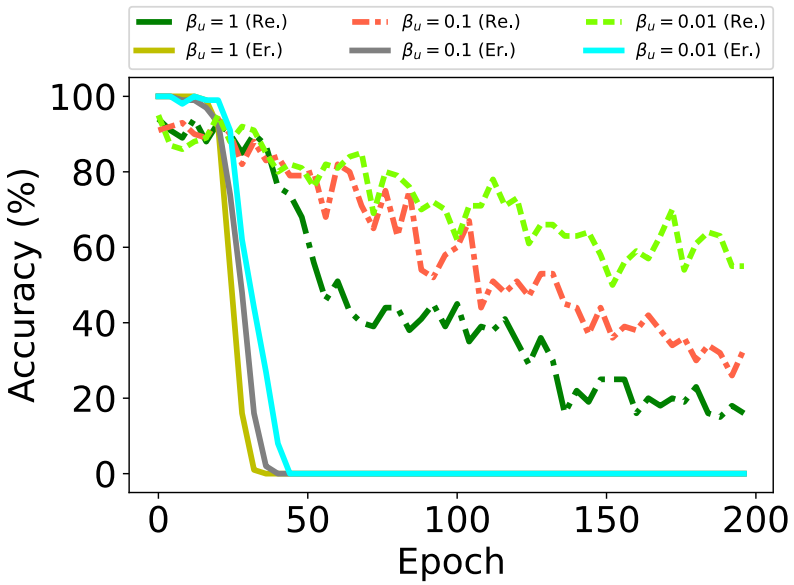


Figure 6.5: The variations in accuracy on the remaining (abbreviated as Re.) dataset and backdoor accuracy on the erased (abbreviated as Er.) dataset during unlearning of various β_u on Fashion MNIST.

dataset even does not decrease in the former 150 epochs training, but the model still removes the backdoor samples within ten epochs at the same time, as shown in Figure 6.6.

Controlling the unlearning speed is one of the advantages of the unlearning rate β_u; slowing down the accuracy degradation during unlearning training is another better advantage. A slower unlearning accuracy degradation speed makes the unlearning catastrophic controllable. For example, when we choose a small β_u, such as 0.01, when the backdoor accuracy decreases to 0, the accuracy of the model still performs like the original model and drops slowly in the continuing training. It gives us more time to observe the unlearning process and stop the model training if accuracy degradation appears. By contrast, if we do not have the unlearning rate β_u, i.e., the $\beta_u = 1$ in all situations, the catastrophic unlearning appears quickly after unlearning the erased samples, which is obvious in Figures 6.4 and 6.5.

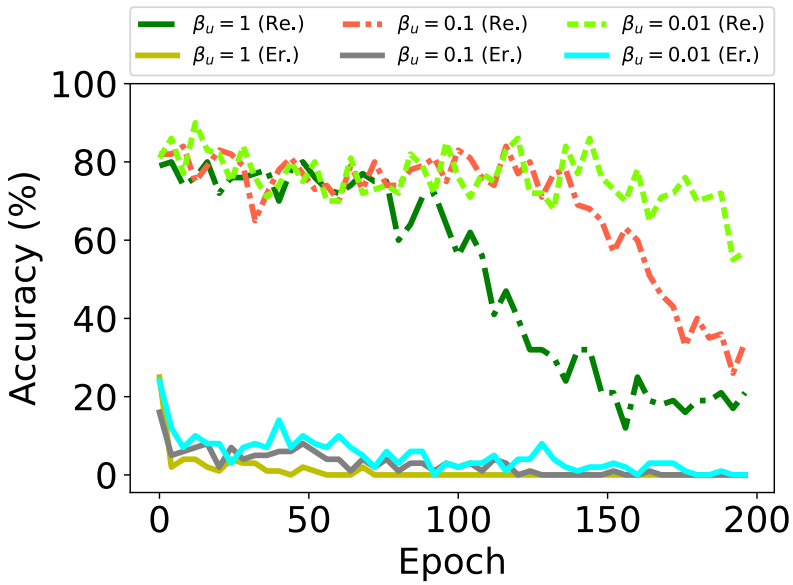


Figure 6.6: The variations in accuracy on the remaining (abbreviated as Re.) dataset and backdoor accuracy on the erased (abbreviated as Er.) dataset during unlearning of various unlearning rate β_u on CIFAR10.

6.5 Summary

In the chapter, we propose the CRFU scheme to defend against privacy leakage attacks on machine unlearning. Specifically, we minimize the information of the erased data remaining in the learned representation to remove the contribution of specified data from the trained model. Since the representation extracts only pertinent information about labels from inputs while distorting other details, it effectively shields against privacy leakage attacks on unlearning. To avoid catastrophic unlearning, we design the remembering constraint term during data erasure and propose an unlearning rate to control the unlearning extent. Our theoretical analysis focuses on the security of CRFU in defense against reconstruction attacks. Additionally, comprehensive experimental evidence shows that our protocol can effectively counter both reconstruction and membership inference attacks based on unlearning model updates while minimizing the impact on unlearning accuracy.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

This thesis focuses on the under-explored challenges arising in machine unlearning. There are many uncharted and important areas in machine unlearning as it is a new solution to support the "right to be forgotten" and draws increasing attention. We narrow our focus to exploring four critical challenges: (1) enhancing the effectiveness and utility preservation of traditional unlearning, (2) implementing effective and efficient unlearning in federated learning, (3) evaluating the efficacy of machine unlearning, and (4) safeguarding privacy during the unlearning process. In the following, we summarize a concise outline of the contributions of this thesis.

A novel representation forgetting technique with parameter self-sharing (RFU-SS) for centralized machine unlearning. We noticed that existing approximate unlearning methods cause dramatic model utility degradation, which is called catastrophic unlearning. To address the catastrophic unlearning problem in centralized machine learning scenarios, we start with formulating machine unlearning as a two-objective optimization problem, including data erasure and accuracy preservation. Then, we proposed the RFU-SS method to maximize the removal of erased samples (forgetting) and preserve the model utility (remembering) during the machine unlearning process. In RFU-SS, we have two technical contributions. First, we propose the representation forgetting method (RFU), 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. Second, we propose a parameter self-sharing scheme to identify a Pareto optimal solution for the defined two-objective unlearning problem, striking the ideal balance between removing the erased data's impact and preserving the model utility.

A Federated Unlearning (FedU) method to facilitate effective and efficient unlearning through user-side influence approximation forgetting. The users in federated learning (FL) definitely also have unlearning requirements to erase their private information from the trained FL models. When unlearning a FL model, we should follow the FL mechanism, i.e., the server has no access to any users' local data, including the unlearning data. We propose FedU and the full version FedU-U to solve the federated unlearning problem. In FedU, only users who have unlearning requirements execute the proposed influence approximation forgetting method locally to remove the influence of the erased samples. Other users and the server just conduct the same operations as they did in FL. Moreover, to mitigate the side effects of unlearning, we propose a utility preservation method that simultaneously trains the unlearned model based on the unlearning requesters' remaining local dataset. We design an adaptive optimization method to balance the forgetting and utility preservation effectiveness optimally during the unlearning process.

An evaluation of machine unlearning (EMU) method for assess the unlearning effectiveness. Post-unlearning, evaluation and auditing methods are critical and necessary. To this end, we investigate the unlearning evaluation problems and propose an EMU approach to assess how much information is unlearned based on the model difference before and after unlearning. In EMU, to generate the unlearning model difference for evaluation efficiently, we propose a model difference simulation scheme based on influence function theory. Then, to enhance the scalability of EMU based on model difference, we design a multi-task information bottleneck structure, which only uses the representation layer difference instead of the whole model difference. Based on the influence function and information bottleneck, we provide a theoretical analysis of how the similarity between erased and remaining samples, as well as task types, impact unlearning effectiveness.

A compressive representation forgetting method to protect against privacy leakage in machine unlearning. Existing studies pointed out that machine unlearning leaks the privacy of the erased samples because the adversary can infer the private information from the model difference before and after unlearning. We propose the compressive representation forgetting unlearning (CRFU) to defend against privacy leak-

age attacks on machine unlearning. To achieve this, our mechanism is designed in the information bottleneck framework, which learns the representation that maximizes the compression of the input data and maximizes the remaining of the task related information from the input data. For machine unlearning, we minimize the information of the erased data remaining in the learned representation. Since the representation extracts only pertinent information about labels from inputs while distorting other details, it effectively shields against privacy leakage attacks on unlearning. To avoid catastrophic unlearning, we design the remembering constraint term during data erasure and propose an unlearning rate to control the unlearning extent.

7.2 Future Work

Although the three challenges in machine unlearning have been solved, there are still lots of uncharted areas and problems. We summarize some problems that we could investigate in future work.

In the common centralized unlearning scenario, retraining from scratch can achieve the best unlearning effect, but it is expensive in both computation and storage. Existing methods try to design new unlearning mechanisms to reduce the cost from the two aspects. Although they proposed many methods, they just mitigated the influence of the main challenges and still have not solved them. Existing unlearning methods were at the beginning phase, which is trying to implement effective unlearning. They proposed mechanisms, but most of them cannot guarantee the final results, so they also bound the unlearned item [36] or set an unlearning threshold [67]. Although they also introduced many metrics to verify the unlearning effect, the unlearning and verification processes are split. This means that the unlearning result is uncertain during the unlearning period before the verification is finished. Therefore, we find and list some open questions in centralized unlearning.

a) Optimizing the existing challenges, such as eliminating the degradation of approximate unlearning and mitigating the stochasticity of unlearning results. Even though many studies exist, there is still a long way to go to solve these problems completely.
b) How do we unlearn with a certain or exact goal? In other words, can we unlearn as learning? If we know the purpose, to what extent we have unlearned, and when we can finish or stop the unlearning process.

Machine unlearning in distributed scenarios has many differences from centralized scenarios. We take federated unlearning as a representative example of distributed unlearning. The first difference is that federated unlearning can only be implemented locally on the client's side if they want to unlearn some specific samples because clients do not upload their data to the FL server in a federated scenario. To avoid interacting with clients during unlearning, researchers [56, 99] proposed to unlearn the contribution of a whole client while not some samples of the client. The second difference is that when unlearning requests come during the FL training process, the FL server must first execute the unlearning process and broadcast the unlearned model for later updating to avoid other clients wasting computation on the before-unlearned model. The third difference is that federated unlearning is more vulnerable to catastrophic degradation than centralized unlearning because if the FL server broadcasts the catastrophic unlearned model and other clients update based on the unlearned model, it will vanish the efforts of other clients that trained before. After introducing these differences, we can see that the challenges in federated unlearning are more complex than in centralized unlearning, and we conclude the following open problems in federated unlearning.

a) Federated unlearning cannot use the traditional fast retraining methods because data is out of reach for the server. Therefore, federated unlearning can only be implemented using approximate unlearning methods. However, as we know, approximate unlearning easily causes catastrophic unlearning, and federated learning is more vulnerable to degradation, so controlling the catastrophic in federated unlearning will be more urgent than in centralized unlearning.
b) Existing federated unlearning methods require the activation of all users, including those without unlearning requests, to assist in the unlearning process. These approaches are impractical and inefficient, particularly when unlearning requests are frequent. Therefore, it is crucial to study how to balance the unlearning effect and efficiency in federated unlearning.

Besides exploring machine unlearning based on regularly structured data, researchers tried to implement unlearning in graph data. Exact unlearning may be suitable for centralized graph unlearning if graph data is sparse. However, the challenges of approximate unlearning may be more difficult than structured-data-based unlearning because, in graph unlearning, the relationship and influence between data samples are more complex than structured data [16]. In particular, graph data includes not only the node feature value but also the connecting edge information. Therefore, in graph unlearning,

the estimation of the contribution of a node will be more difficult than in regular data unlearning. The original problems in regular data unlearning will be more challenging in graph unlearning. Besides these problems, graph unlearning also faces unique problems that are related to edge structure information. In graph unlearning, unlearning some edges or sub-graphs is a big question.

After designing machine unlearning algorithms, effective verification and auditing methods are necessary [86]. Most existing unlearning verification methods rely on backdooring techniques. However, these methods inherently degrade model utility because they require mixing backdoored samples into the model training process. Investigating ways to preserve model utility in backdoor-based unlearning verification methods is an under-explored area. Additionally, backdoor-based verification methods must include backdoored samples in the unlearning requests to verify the effectiveness of these requests. This requirement restricts these methods from supporting verification for single-sample unlearning requests. Exploring new strategies that can effectively verify unlearning requests without compromising model utility or being suitable for both single-sample and multi-sample unlearning scenarios remains a significant challenge in this field.

Another important part is the privacy and security issues in machine unlearning. Machine unlearning was first proposed to protect users' privacy, but it brings new threats in that adversaries have a chance to infer the information about the removed data. Literature [15, 105] have pointed out that updates of unlearning will leak privacy information, and they proposed corresponding attacks to infer this private information. However, the most recent unlearning privacy leakage attacks have been similar to those in a learning situation. An attack that is tailored to unlearning mechanisms is expected. A similar situation exists in unlearning applications. Although researchers proposed to unlearn a backdoor trigger [57] or pollution [13], they only used a few unlearning techniques and paid more effort to detect those anomalies. One important reason is that the unlearning mechanism is not mature enough. We are at the beginning of machine unlearning investigation, and there are still many under-explored problems in unlearning itself. Finding machine unlearning applying situation and tailoring unlearning techniques to this situation is the direction of unlearning application.

BIBLIOGRAPHY

[1] A. ACHILLE AND S. SOATTO, *Information dropout: Learning optimal representations through noisy computation*, IEEE transactions on pattern analysis and machine intelligence, 40 (2018), pp. 2897–2905.

[2] A. A. ALEMI, I. FISCHER, J. V. DILLON, AND K. MURPHY, *Deep variational information bottleneck*, in 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017.

[3] K. E. AVRACHENKOV, J. A. FILAR, AND P. G. HOWLETT, *Analytic perturbation theory and its applications*, SIAM, 2013.

[4] J. BAE, N. NG, A. LO, M. GHASSEMI, AND R. B. GROSSE, *If influence functions are the answer, then what is the question?*, Advances in Neural Information Processing Systems, 35 (2022), pp. 17953–17967.

[5] S. BANG, P. XIE, H. LEE, W. WU, AND E. XING, *Explaining a black-box by using a deep variational information bottleneck approach*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 11396–11404.

[6] S. BASU, X. YOU, AND S. FEIZI, *On second-order group influence functions for black-box predictions*, in International Conference on Machine Learning, PMLR, 2020, pp. 715–724.

[7] T. BAUMHAUER, P. SCHÖTTLE, AND M. ZEPPELZAUER, *Machine unlearning: Linear filtration for logit-based classifiers*, Machine Learning, (2022), pp. 1–24.

[8] M. I. BELGHAZI, A. BARATIN, S. RAJESHWAR, S. OZAIR, Y. BENGIO, A. COURVILLE, AND D. HJELM, *Mutual information neural estimation*, in International conference on machine learning, PMLR, 2018, pp. 531–540.

[9] T. BERTRAM, E. BURSZTEIN, S. CARO, H. CHAO, R. CHIN FEMAN, P. FLEISCHER, A. GUSTAFSSON, J. HEMERLY, C. HIBBERT, L. INVERNIZZI, ET AL., *Five years of the right to be forgotten*, in Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 959–972.

[10] R. BLAHUT, *Computation of channel capacity and rate-distortion functions*, IEEE transactions on Information Theory, 18 (1972), pp. 460–473.

[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), IEEE, 2021, pp. 141–159.

[12] Y. CAO AND J. YANG, *Towards making systems forget with machine unlearning*, in 2015 IEEE Symposium on Security and Privacy, IEEE, 2015, pp. 463–480.

[13] Y. CAO, A. F. YU, A. ADAY, E. STAHL, J. MERWINE, AND J. YANG, *Efficient repair of polluted machine learning systems via causal unlearning*, in Proceedings of the 2018 on Asia Conference on Computer and Communications Security, 2018, pp. 735–747.

[14] C. CHEN, F. SUN, M. ZHANG, AND B. DING, *Recommendation unlearning*, in Proceedings of the ACM Web Conference 2022, 2022, pp. 2768–2777.

[15] M. CHEN, Z. ZHANG, T. WANG, M. BACKES, M. HUMBERT, AND Y. ZHANG, *When machine unlearning jeopardizes privacy*, in Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021, pp. 896–911.

[16] M. CHEN, Z. ZHANG, T. WANG, M. BACKES, M. HUMBERT, AND Y. ZHANG, *Graph unlearning*, in Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 499–513.

[17] V. S. CHUNDAWAT, A. K. TARUN, M. MANDAL, AND M. KANKANHALLI, *Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 7210–7217.

[18] A. COATES, A. NG, AND H. LEE, *An analysis of single-layer networks in unsupervised feature learning*, in Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[19] L. DE LA TORRE, *A guide to the california consumer privacy act of 2018*, Available at SSRN 3275571, (2018).

[20] L. DENG, *The mnist database of handwritten digit images for machine learning research [best of the web]*, IEEE signal processing magazine, 29 (2012), pp. 141–142.

[21] J.-A. DÉSIDÉRI, *Multiple-gradient descent algorithm (mgda) for multiobjective optimization*, Comptes Rendus Mathematique, 350 (2012), pp. 313–318.

[22] M. DU, Z. CHEN, C. LIU, R. OAK, AND D. SONG, *Lifelong anomaly detection through unlearning*, in Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1283–1297.

[23] L. DUONG, T. COHN, S. BIRD, AND P. COOK, *Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser*, in Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers), 2015, pp. 845–850.

[24] C. DWORK, *Differential privacy*, in Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33, Springer, 2006, pp. 1–12.

[25] J. FLIEGE AND B. F. SVAITER, *Steepest descent methods for multicriteria optimization*, Mathematical methods of operations research, 51 (2000), pp. 479–494.

[26] C. W. FOX AND S. J. ROBERTS, *A tutorial on variational bayesian inference*, Artificial intelligence review, 38 (2012), pp. 85–95.

[27] M. FREDRIKSON, E. LANTZ, S. JHA, S. M. LIN, D. PAGE, AND T. RISTENPART, *Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing*, in Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014, USENIX Association, 2014, pp. 17–32.

[28] S. FU, F. HE, AND D. TAO, *Knowledge removal in sampling-based bayesian inference*, in The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, OpenReview.net, 2022.

[29] J. GAO, S. GARG, M. MAHMOODY, AND P. N. VASUDEVAN, *Deletion inference, reconstruction, and compliance in machine (un)learning*, Proc. Priv. Enhancing Technol., 2022 (2022), pp. 415–436.

[30] K. GAO, Y. BAI, J. GU, Y. YANG, AND S.-T. XIA, *Backdoor defense via adaptively splitting poisoned dataset*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4005–4014.

[31] G. GIORGI, B. JIMÉNEZ, AND V. NOVO, *Approximate karush–kuhn–tucker condition in multiobjective optimization*, Journal of Optimization Theory and Applications, 171 (2016), pp. 70–89.

[32] A. GOLATKAR, A. ACHILLE, A. RAVICHANDRAN, M. POLITO, AND S. SOATTO, *Mixed-privacy forgetting in deep networks*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 792–801.

[33] A. GOLATKAR, A. ACHILLE, AND S. SOATTO, *Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations*, in Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, eds., vol. 12374 of Lecture Notes in Computer Science, Springer, 2020, pp. 383–398.

[34] L. GRAVES, V. NAGISETTY, AND V. GANESH, *Amnesiac machine learning*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 11516–11524.

[35] A. GRIEWANK AND A. WALTHER, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, SIAM, 2008.

[36] C. GUO, T. GOLDSTEIN, A. Y. HANNUN, AND L. VAN DER MAATEN, *Certified data removal from machine learning models*, in Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, vol. 119 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 3832–3842.

[37] P. GUO, C.-Y. LEE, AND D. ULBRICHT, *Learning to branch for multi-task learning*, in International Conference on Machine Learning, PMLR, 2020, pp. 3854–3863.

[38] Y. GUO, Y. ZHAO, S. HOU, C. WANG, AND X. JIA, *Verifying in the dark: Verifiable machine unlearning by using invisible backdoor triggers*, IEEE Transactions on Information Forensics and Security, (2023).

[39] A. HALIMI, S. KADHE, A. RAWAT, AND N. BARACALDO, *Federated unlearning: How to efficiently erase a client in fl?*, arXiv preprint arXiv:2207.05521, (2022).

[40] H. HU, Z. SALCIC, G. DOBBIE, J. CHEN, L. SUN, AND X. ZHANG, *Membership inference via backdooring*, in Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022, L. D. Raedt, ed., ijcai.org, 2022, pp. 3832–3838.

[41] H. HU, S. WANG, T. DONG, AND M. XUE, *Learn what you want to unlearn: Unlearning inversion attacks against machine unlearning*, in 2024 IEEE Symposium on Security and Privacy (SP), Los Alamitos, CA, USA, may 2024, IEEE Computer Society, pp. 262–262.

[42] Y. HUANG, X. LI, AND K. LI, *EMA: auditing data removal from trained models*, in Medical Image Computing and Computer Assisted Intervention - MICCAI 2021 - 24th International Conference, Strasbourg, France, September 27 - October 1, 2021, Proceedings, Part V, M. de Bruijne, P. C. Cattin, S. Cotin, N. Padoy, S. Speidel, Y. Zheng, and C. Essert, eds., vol. 12905 of Lecture Notes in Computer Science, Springer, 2021, pp. 793–803.

[43] J. M. JOYCE, *Kullback-leibler divergence*, in International Encyclopedia of Statistical Science, Springer, Berlin, Heidelberg, 2011, pp. 720–722.

[44] A. KENDALL, Y. GAL, AND R. CIPOLLA, *Multi-task learning using uncertainty to weigh losses for scene geometry and semantics*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7482–7491.

[45] D. P. KINGMA AND M. WELLING, *Auto-encoding variational bayes*, stat, 1050 (2014), p. 1.

[46] J. KIRKPATRICK, R. PASCANU, N. RABINOWITZ, J. VENESS, G. DESJARDINS, A. A. RUSU, K. MILAN, J. QUAN, T. RAMALHO, A. GRABSKA-BARWINSKA,

ET AL., *Overcoming catastrophic forgetting in neural networks*, Proceedings of the national academy of sciences, 114 (2017), pp. 3521–3526.

[47] P. W. KOH AND P. LIANG, *Understanding black-box predictions via influence functions*, in International conference on machine learning, PMLR, 2017, pp. 1885–1894.

[48] A. KRIZHEVSKY, G. HINTON, ET AL., *Learning multiple layers of features from tiny images*, (2009).

[49] H. W. KUHN AND A. W. TUCKER, *Nonlinear programming*, in Traces and emergence of nonlinear programming, Basel: Springer Basel, 2013, pp. 247–258.

[50] M. KURMANJI, P. TRIANTAFILLOU, J. HAYES, AND E. TRIANTAFILLOU, *Towards unbounded machine unlearning*, Advances in Neural Information Processing Systems, 36 (2024).

[51] M. J. KUSNER, B. PAIGE, AND J. M. HERNÁNDEZ-LOBATO, *Grammar variational autoencoder*, in International conference on machine learning, PMLR, 2017, pp. 1945–1954.

[52] Y. LI, C. WANG, AND G. CHENG, *Online forgetting process for linear regression models*, in The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event, A. Banerjee and K. Fukumizu, eds., vol. 130 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 217–225.

[53] J. LIN, L. XU, Y. LIU, AND X. ZHANG, *Composite backdoor attack for deep neural network by mixing existing benign features*, in Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 113–131.

[54] X. LIN, H. ZHEN, Z. LI, Q. ZHANG, AND S. KWONG, *Pareto multi-task learning*, in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 12037–12047.

[55] G. LIU, X. MA, Y. YANG, C. WANG, AND J. LIU, *Federated unlearning*, arXiv preprint arXiv:2012.13891, (2020).

[56] 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), IEEE, 2021, pp. 1–10.

[57] Y. LIU, M. FAN, C. CHEN, X. LIU, Z. MA, L. WANG, AND J. MA, *Backdoor defense with machine unlearning*, in IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, May 2-5, 2022, IEEE, 2022, pp. 280–289.

[58] 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, London, United Kingdom, May 2-5, 2022, IEEE, 2022, pp. 1749–1758.

[59] Z. LIU, P. LUO, X. WANG, AND X. TANG, *Large-scale celebfaces attributes (celeba) dataset*, Retrieved August, 15 (2018), p. 11.

[60] Z. LU, H. LIANG, M. ZHAO, Q. LV, T. LIANG, AND Y. WANG, *Label-only membership inference attacks on machine unlearning without dependence of posteriors*, International Journal of Intelligent Systems, (2022).

[61] 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 Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 216–223.

[62] P. MA, T. DU, AND W. MATUSIK, *Efficient continuous pareto exploration in multi-task learning*, in International Conference on Machine Learning, PMLR, 2020, pp. 6522–6531.

[63] M. S. MAHDAVINEJAD, M. REZVAN, M. BAREKATAIN, P. ADIBI, P. BARNAGHI, AND A. P. SHETH, *Machine learning for internet of things data analysis: A survey*, Digital Communications and Networks, 4 (2018), pp. 161–175.

[64] G. MAI, K. CAO, P. C. YUEN, AND A. K. JAIN, *On the reconstruction of face images from deep face templates*, IEEE transactions on pattern analysis and machine intelligence, 41 (2018), pp. 1188–1202.

[65] A. MANTELERO, *The EU proposal for a general data protection regulation and the roots of the 'right to be forgotten'*, Comput. Law Secur. Rev., 29 (2013), pp. 229–235.

[66] R. MEHTA, S. PAL, V. SINGH, AND S. N. RAVI, *Deep unlearning via randomized conditionally independent hessians*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10422–10431.

[67] Q. P. NGUYEN, B. K. H. LOW, AND P. JAILLET, *Variational bayesian unlearning*, Advances in Neural Information Processing Systems, 33 (2020), pp. 16025–16036.

[68] 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 ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022, ACM, 2022, pp. 351–363.

[69] O. OF THE PRIVACY COMMISSIONER OF CANADA, *Announcement: Privacy commissioner seeks federal court determination on key issue for canadians' online reputation*, (2018).

[70] M. PAN, Y. ZENG, L. LYU, X. LIN, AND R. JIA, *Asset: Robust backdoor data detection across a multiplicity of deep learning paradigms*, in 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 2725–2742.

[71] N. PAPERNOT, P. MCDANIEL, I. GOODFELLOW, S. JHA, Z. B. CELIK, AND A. SWAMI, *Practical black-box attacks against machine learning*, in Proceedings of the 2017 ACM on Asia conference on computer and communications security, 2017, pp. 506–519.

[72] 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.

[73] F. POIRION, Q. MERCIER, AND J.-A. DÉSIDÉRI, *Descent algorithm for nonsmooth stochastic multiobjective optimization*, Computational Optimization and Applications, 68 (2017), pp. 317–331.

[74] D. ROLNICK, A. AHUJA, J. SCHWARZ, T. LILLICRAP, AND G. WAYNE, *Experience replay for continual learning*, Advances in neural information processing systems, 32 (2019).

[75] S. RUDER, J. BINGEL, I. AUGENSTEIN, AND A. SØGAARD, *Latent multi-task architecture learning*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 4822–4829.

[76] A. SALEM, A. BHATTACHARYA, M. BACKES, M. FRITZ, AND Y. ZHANG, *Updates-leak: Data set inference and reconstruction attacks in online learning*, in 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020, USENIX Association, 2020, pp. 1291–1308.

[77] S. SCHÄFFLER, R. SCHULTZ, AND K. WEINZIERL, *Stochastic method for the solution of unconstrained vector optimization problems*, Journal of Optimization Theory and Applications, 114 (2002), pp. 209–222.

[78] A. SEKHARI, J. ACHARYA, G. KAMATH, AND A. T. SURESH, *Remember what you want to forget: Algorithms for machine unlearning*, Advances in Neural Information Processing Systems, 34 (2021).

[79] O. SENER AND V. KOLTUN, *Multi-task learning as multi-objective optimization*, in Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018, pp. 525–536.

[80] R. SHOKRI AND V. SHMATIKOV, *Privacy-preserving deep learning*, in Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1310–1321.

[81] R. SHOKRI, M. STRONATI, C. SONG, AND V. SHMATIKOV, *Membership inference attacks against machine learning models*, in 2017 IEEE symposium on security and privacy (SP), IEEE, 2017, pp. 3–18.

[82] D. M. SOMMER, L. SONG, S. WAGH, AND P. MITTAL, *Athena: Probabilistic verification of machine unlearning*, Proceedings on Privacy Enhancing Technologies, 3 (2022), pp. 268–290.

[83] N. SU AND B. LI, *Asynchronous federated unlearning*, in IEEE INFOCOM 2023-IEEE Conference on Computer Communications, IEEE, 2023, pp. 1–10.

[84] X. SUN, R. PANDA, R. FERIS, AND K. SAENKO, *Adashare: Learning what to share for efficient deep multi-task learning*, Advances in Neural Information Processing Systems, 33 (2020), pp. 8728–8740.

[85] A. THUDI, G. DEZA, V. CHANDRASEKARAN, AND N. PAPERNOT, *Unrolling sgd: Understanding factors influencing machine unlearning*, in 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), IEEE, 2022, pp. 303–319.

[86] A. THUDI, H. JIA, I. SHUMAILOV, AND N. PAPERNOT, *On the necessity of auditable algorithmic definitions for machine unlearning*, in 31st USENIX Security Symposium (USENIX Security 22), 2022, pp. 4007–4022.

[87] Z. TIAN, L. CUI, J. LIANG, AND S. YU, *A comprehensive survey on poisoning attacks and countermeasures in machine learning*, ACM Computing Surveys, 55 (2022), pp. 1–35.

[88] N. TISHBY, F. C. PEREIRA, AND W. BIALEK, *The information bottleneck method*, arXiv preprint physics/0004057, (2000).

[89] N. TISHBY AND N. ZASLAVSKY, *Deep learning and the information bottleneck principle*, in 2015 ieee information theory workshop (itw), IEEE, 2015, pp. 1–5.

[90] B. WANG AND N. Z. GONG, *Stealing hyperparameters in machine learning*, in 2018 IEEE symposium on security and privacy (SP), IEEE, 2018, pp. 36–52.

[91] H. WANG, Z. XIANG, D. J. MILLER, AND G. KESIDIS, *Mm-bd: Post-training detection of backdoor attacks with arbitrary backdoor pattern types using a maximum margin statistic*, in 2024 IEEE Symposium on Security and Privacy (SP), IEEE Computer Society, 2023, pp. 15–15.

[92] J. WANG, S. GUO, X. XIE, AND H. QI, *Federated unlearning via class-discriminative pruning*, in Proceedings of the ACM Web Conference 2022, 2022, pp. 622–632.

[93] W. WANG, Z. TIAN, C. ZHANG, A. LIU, AND S. YU, *Bfu: Bayesian federated unlearning with parameter self-sharing*, in Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security, 2023, pp. 567–578.

[94] W. WANG, C. ZHANG, Z. TIAN, AND S. YU, *Crfu: Compressive representation forgetting against reconstruction attacks on machine unlearning*, IEEE Transactions on Dependable and Secure Computing, **Major Revision**, pp. –.

[95] W. WANG, C. ZHANG, Z. TIAN, AND S. YU, *Evaluating machine unlearning based on model difference*, IEEE Transactions on Information Forensics and Security, **Under Review**, pp. –.

[96] W. WANG, C. ZHANG, Z. TIAN, AND S. YU, *Fedu: Federated unlearning via user-side influence approximation forgetting*, IEEE Transactions on Dependable and Secure Computing, **Minor Revision**, pp. –.

[97] W. WANG, C. ZHANG, Z. TIAN, AND S. YU, *Machine unlearning via representation forgetting with parameter self-sharing*, IEEE Transactions on Information Forensics and Security, 19 (2024), pp. 1099–1111.

[98] A. WARNECKE, L. PIRCH, C. WRESSNEGGER, AND K. RIECK, *Machine unlearning of features and labels*, 31th Annual Network and Distributed System Security Symposium, NDSS 2024, (2024).

[99] C. WU, S. ZHU, AND P. MITRA, *Federated unlearning with knowledge distillation*, arXiv preprint arXiv:2201.09441, (2022).

[100] G. WU, M. HASHEMI, AND C. SRINIVASA, *PUMA: performance unchanged model augmentation for training data removal*, in Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, AAAI Press, 2022, pp. 8675–8682.

[101] Y. WU, E. DOBRIBAN, AND S. DAVIDSON, *Deltagrad: Rapid retraining of machine learning models*, in International Conference on Machine Learning, PMLR, 2020, pp. 10355–10366.

[102] H. XIAO, K. RASUL, AND R. VOLLGRAF, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, arXiv preprint arXiv:1708.07747, (2017).

[103] P. XU, F. ROOSTA, AND M. W. MAHONEY, *Second-order optimization for non-convex machine learning: An empirical study*, in Proceedings of the 2020 SIAM International Conference on Data Mining, SIAM, 2020, pp. 199–207.

[104] H. YAN, X. LI, Z. GUO, H. LI, F. LI, AND X. LIN, *ARCANE: an efficient architecture for exact machine unlearning*, in Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022, L. D. Raedt, ed., ijcai.org, 2022, pp. 4006–4013.

[105] S. ZANELLA-BÉGUELIN, L. WUTSCHITZ, S. TOPLE, V. RÜHLE, A. PAVERD, O. OHRIMENKO, B. KÖPF, AND M. BROCKSCHMIDT, *Analyzing information leakage of updates to natural language models*, in Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 363–375.

[106] K. ZHANG, W. WANG, Z. FAN, X. SONG, AND S. YU, *Conditional matching gan guided reconstruction attack in machine unlearning*, in GLOBECOM 2023-2023 IEEE Global Communications Conference, IEEE, 2023, pp. 44–49.

[107] P. ZHAO, P. CHEN, P. DAS, K. N. RAMAMURTHY, AND X. LIN, *Bridging mode connectivity in loss landscapes and adversarial robustness*, in 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020.