Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version 11 January, 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.011100

Private Data Protection with Machine Unlearning for Next-Generation Networks

Kongyang Chen[1,2,3], Yiwen Wang[1], Lanlan Zhao[4], Caiyun Jiang[4], Haiyan Mai[4], Yuanying Wu[4], Huanqi Hong[4], Yuanyuan Shen[4], Jian Mo[4], Lin-Lu Huang[5], Jun Peng[5], Xiaoying Wang[4,6], Qintai Yang[4]

[1]School of Artificial Intelligence, Guangzhou University, Guangzhou, 510006, China
[2]Pazhou Lab, Guangzhou, 510330, China
[3]Yunnan Key Laboratory of Service Computing, Yunnan University of Finance and Economics, Kunming, 650221, China
[4]Third Affiliated Hospital of Sun Yat-sen University, Guangzhou 510630, China
[5]Xiangya School of Pharmaceutical Sciences, Central South University, Changsha 410013, China
[6]Guangdong Key Laboratory of Blockchain Security, Guangzhou University, Guangzhou 510006, China

CORRESPONDING AUTHORS: Xiaoying Wang (e-mail: wxying@mail.sysu.edu.cn), Qintai Yang (e-mail: yangqint@mail.sysu.edu.cn).

This work was supported by Key-Area Research and Development Program of Guangdong Province (No. 2020B0101130015), National Natural Science Foundation of China (No. 61802383), Research Project of Pazhou Lab for Excellent Young Scholars (No. PZL2021KF0024), University Research Project of Guangzhou Education Bureau (No. 2024312189), Foundation of Yunnan Key Laboratory of Service Computing (No.YNSC24115), Guangzhou Basic and Applied Basic Research Project (No. SL2024A03J00397), and the Open Research Fund of Guangdong Key Laboratory of Blockchain Security, Guangzhou University (No.2023B1212010013).

ABSTRACT In next-generation networks, distributed clients collaborate to generate an aggregated global model tailored for various vertical applications. However, this convenience comes at the cost of potential privacy risks, as personal information may be exposed within the global model aggregation process. In response, the *right to be forgotten* was introduced, granting individuals the right to withdraw their consent for the processing of their personal information. To address this challenge, *machine unlearning* has been developed, enabling models to erase any memory of private data. Previous approaches, such as retraining or incremental learning, often require additional storage or are difficult to implement in neural networks. Our method, by contrast, introduces a small perturbation to the model's weights, guiding it to iteratively move towards a model trained only on the remaining data subset until the contribution of the unlearned data is completely removed. In our approach, machine unlearning is conceptualized as a process that iteratively adjusts the initial model to remove any trace of the forgotten data. Our key contribution is the introduction of a reference model, trained on a subset of the remaining data, which guides the target unlearning model toward successfully forgetting the data. Additionally, we discuss two evaluation methods—membership inference and backdoor evaluation—that effectively assess the success of our machine unlearning approach. These methods verify whether the private data has truly been forgotten by the target unlearning model. Through experiments on five datasets, we demonstrate the effectiveness of our approach, which is 15× faster than the traditional retraining method.

INDEX TERMS Machine Unlearning, Data Privacy, Neural Networks

I. Introduction

In next-generation networks, distributed clients collaboratively train deep neural models to support vertical applications [1]–[3] such as autonomous driving, medical diagnosis, and industrial detection. While these advancements offer significant convenience, they also raise concerns about data privacy [4]. To power these services, large amounts of personal data are collected and analyzed, with systems often deriving additional insights beyond the original information. For example, systems may analyze user behavior to derive more statistical data. However, this process can inadvertently lead to privacy leaks, revealing sensitive information like family address from autonomous driving data [5], [6]. With growing concerns over these risks, users are increasingly demanding that their data, and any contributions it has made to models, should be "forgotten" from potential databases and pre-trained models. In response, machine unlearning has emerged as a solution, ensuring that private data can be erased from models while preserving privacy in distributed learning environments.

Differential privacy [7] offers a measure of protection for user privacy by adding noise to the data, making it impossible to determine whether any single sample is part of the dataset. It ensures that the influence of each training sample on the model is constrained. However, unlike machine unlearning, differential privacy does not allow for the complete removal of a data sample's contribution to the model, as doing so would compromise the model's ability to learn effectively. Machine unlearning, on the other hand, aims to entirely eliminate the impact of personal data from the model [8]. This involves not only removing user data from the database but also ensuring that the data's influence on the model is fully eradicated. While it might seem straightforward to achieve this by simply removing the relevant data and retraining the model, such an approach is often prohibitively time-consuming. Currently, most methods for machine unlearning focus on model retraining and updating through computable transformations [9], but these techniques face limitations and are challenging to implement within neural network frameworks.

Recent advances in machine unlearning have proposed various approaches, each facing notable challenges. Cao's method [9] relies on statistical queries to enable faster unlearning but is restricted to non-adaptive models, limiting its applicability to neural networks. Liu's federated learning-based approach [10] reduces retraining time but demands significant storage for saving intermediate parameters and fails to guarantee complete unlearning, as these parameters may retain information to be forgotten. Ginart et al.'s work [11] focuses on retraining models after data removal, with a primary application to k-means clustering, but its utility for supervised learning remains limited. Izzo et al. [12] addressed unlearning in linear computations, achieving scalability but offering narrow applicability. Methods like linear transformations fail to fully eliminate the contribution of forgotten data, while techniques based on convex functions, Bayesian adjustments, and gradient descent often face trade-offs between unlearning efficiency, computational cost, and privacy guarantees. Collectively, these methods struggle with scalability, completeness, or generalizability, particularly for complex models like neural networks.

In this paper, our objective is to provide a lightweight and efficient method that requires minimal time and space, while ensuring little to no loss in the accuracy of model unlearning. To avoid the significant time cost associated with retraining, we propose a simple correction to the pre-unlearning model by introducing a small perturbation to its weights. This perturbation eliminates the influence of the data that needs to be forgotten. The key is to first identify a reference direction for the model correction, allowing the pre-unlearning model to move towards this direction.

In our approach, we employ a reference model to guide the weight adjustments of the target model. The reference model must not contain any information related to the data to be forgotten, and its weight distribution should closely resemble that of the target model. This similarity ensures that the modifications to the target model's weights have minimal impact on its overall performance. Ideally, the retrained model would serve as the best reference model if not for the significant time cost it incurs. Thus, the reference model should be selected based on the retrained model, aiming to be as similar as possible. During the unlearning process, the data to be forgotten is input into the reference model to obtain its output distribution, which serves as the standard. The same data is then input into the target model, and the output distribution is iteratively adjusted to match the standard. Once the two distributions align, the target model's weights are updated accordingly, achieving the goal of unlearning. Additionally, we propose two evaluation methods—membership inference evaluation and backdoor evaluation—to assess the success of our machine unlearning approach. These methods verify whether the private data has indeed been forgotten by the target model.

The contributions of this paper can be summarized as follows:

1) Novel general machine unlearning method: we propose a new machine unlearning approach that is compatible with all neural network architectures without requiring server assistance or multiple users. Our method involves making a simple correction to the model before unlearning by introducing a small perturbation to its weights, effectively removing the influence of data that needs to be forgotten. A reference direction is determined to guide this correction, ensuring the pre-unlearning model adjusts accordingly.
2) Evaluation metrics for unlearning success: we introduce two effective evaluation methods—membership inference evaluation and backdoor evaluation—to assess the success of our machine unlearning method. These techniques confirm whether the private data has indeed been completely removed from the target model.
3) Extensive performance evaluation: we evaluate the performance of our method on five datasets. Experimental results demonstrate that our approach is 15 $\times$ faster than traditional retraining methods, with little to no reduction in model accuracy, underscoring the effectiveness of our technique.

The remainder of this paper is organized as follows: Section II presents the background and motivation for our work. Section III introduces our method. Section IV describes the performance evaluation approaches. Section V discusses the experimental results. Section VI reviews related work. Finally, Section VII concludes the paper.

II. Background and Motivation

In next-generation networks, the seamless communication and collaboration among distributed individuals and devices will enable a wide range of vertical applications, revolutionizing sectors such as smart cities, autonomous vehicles, and healthcare [13]. These smart systems rely on the continuous collection of raw data from participating users, sensors, and other connected entities to build comprehensive, real-time datasets. Through collaborative learning frameworks, such as federated learning, this data is used to jointly train a global, aggregated model that possesses advanced inference and reasoning capabilities [14]. Such models are crucial for enhancing decision-making processes in smart city infrastructures, optimizing traffic flow, improving energy management, and enhancing public safety measures. The ability to derive accurate insights from distributed data sources ensures that these systems can adapt quickly to dynamic environments while maintaining a high level of accuracy.

However, the practical use of data suffers from significant issues, including information leakage and data misuse. For instance, some illegal organizations collect data to conduct targeted fraud, while several companies engage in big data pricing based on users' historical data. To address these issues, privacy-preserving computing techniques have emerged in recent years and were recognized by Gartner as one of the top ten strategic technology trends [15]. Prominent methods include federated learning [16] [17] and differential privacy, which aim to improve machine learning models without compromising user privacy. However, recent studies [18] indicate that federated learning and similar approaches still face serious security risks, as malicious users can almost reconstruct local data after several iterations of the training process. This vulnerability arises because machine learning models inherently "remember" the local data used during training. Additionally, techniques such as membership inference attacks [19] can identify whether specific data was used in model training, further aggravating data privacy concerns.

To tackle data privacy concerns, we require a different approach that enables forgetting specific private data from a pre-trained machine learning model, known as *machine unlearning*. Actually, numerous laws and regulations have emphasized the need for machine unlearning, demonstrating that users be allowed to cancel data authorization and erase the impact of their data. For example, the EU General Data Protection Regulation (GDPR) [20] enshrines the *right to be forgotten*, allowing data subjects to rectify, erase, or block incomplete or incorrect data. Similarly, Chinese Personal Information Protection Law [21], specifically Article 16, grants individuals the right to withdraw consent for the processing of their personal information.

Several studies have explored machine unlearning. The most intuitive method involves deleting the specific data from the input sample and retraining the model using the remaining data [22]. However, this approach is time-consuming. Some earlier research proposed using statistical queries [9] to extract dataset characteristics rather than directly training on the data, thereby achieving unlearning more quickly than full retraining. Unfortunately, this method is only effective for non-adaptive machine learning models where late training does not depend on early training, and it is challenging to apply to neural networks. Bourtoule [23] proposed a method where the dataset is partitioned, with each part trained as a separate sub-model, which are then combined through incremental learning. To forget a sample, retraining begins from the first intermediate model that contains the sample's contribution. While this method reduces training time, it requires significant storage space. Another approach [10] involves saving the aggregated model's updated parameters during regular training and deleting the data after training, thereby reducing the number of client training iterations. However, this approach also increases storage demands because the saved parameters themselves retain information meant to be forgotten, theoretically compromising complete unlearning.

In summary, existing machine unlearning methods face limitations in terms of computational efficiency, application scenarios, and storage requirements. To enable a fast and accurate private data forgotten from its pre-trained machine unlearning model, our paper will propose a lightweight machine unlearning method applicable to general scenarios.

III. Our Machine Unlearning Method

A. Overview

In next generation networks, aggregated models require extensive training on a large number of relevant data samples, often leading to prolonged training times. Consequently, one straightforward yet time-consuming approach to achieving erasure of specific data in a target model is to delete the data to be forgotten and retrain the model from scratch. To circumvent this issue, our method focuses on fine-tuning the model's parameters, preserving the contributions of the remaining (non-forgotten) data to the model. Specifically, we utilize a reference model to guide the fine-tuning of the target model. By gradually adjusting the target model towards the reference model which is unfamiliar with the forgotten data, we aim to achieve the goal of forgetting the specified data while maintaining the overall performance of the target model.

B. Our Unlearning Method

Unlike the approach of deleting the relevant data to be forgotten and retraining the model, our method aims to fine-tune the model's parameters while retaining the contributions of the remaining data that do not need to be forgotten.

Before performing the forgetting operation, we prepare a reference model that guides the adjustment of the target model during the forgetting process, as illustrated in Figure 1. During the forgetting process, the reference model remains unchanged. We input the data to be forgotten into the reference model to obtain a posterior distribution, and similarly, input the same data into the target model to obtain another posterior distribution. The goal is to minimize the similarity between these two distributions. The feedback from this similarity measure is then used to iteratively update the target model until its posterior distribution for the data to be forgotten matches that of the reference model. At this point, the fine-tuning of the target model's parameters is complete, achieving the desired forgetting.

Let D represent the dataset composed of images x_i, each corresponding to a label $y_i \in \{1, \dots, k\}$, representing a particular class. $D_f \subset D$ is a subset of the dataset D, defined as $D_f = \{\{x_i, y_i\}\}_{i=1}^{N}$, which represents the data we intend to erase. The remaining subset, denoted as $D_r = D - D_f$, represents the data that do not need to be erased. The model trained on the complete dataset $D = D_f \cup D_r$ is denoted as f_ω, with parameters ω. The model trained on the remaining data D_r is denoted as f_θ, with parameters θ. Let $S(\omega; D_f)$ represent our forgetting method applied to ω. Our objective is to ensure that an attacker cannot extract any information about D_f from ω, thereby preventing the reconstruction of D_f. We assume that in f_ω, (x_i, y_i) follows the distribution $q(x, y)$, and in f_θ, (x_i, y_i) follows the distribution $p(x, y)$. Our goal is to ensure that an attacker cannot extract any information about D_f from f_ω, thereby preventing the reconstruction of D_f.

During the forgetting phase, our objective is to find a specific perturbation to the weights ω of the target model f_ω that minimizes the similarity between the output distribution q of D_f in the target model f_ω and the output distribution p in the reference model f_θ, thereby simulating the output distribution p of the forgotten data D_f in the reference model f_θ. The loss function for this process is:

$$Loss = \mathbb{S}(f_\omega(x), f_\theta(x)). \tag{1}$$

It is important to note that when evaluating the effectiveness of forgetting, we must ensure that the target model forgets the data to be erased while maintaining its overall performance. In other words, the accuracy of the target model should not significantly decrease after the forgetting process. In practice, we introduce a penalty term to constrain the model parameters. Let $x' \in D_r$ with a corresponding label Y. We compute the cross-entropy loss between the predicted label and the true label to correct any parameter drift in the model:

$$H_{CE}\ (f_\omega(x'), Y) = -[f_\omega(x') logY + (1 - f_\omega(x')\ ) log(1 - Y)]. \tag{2}$$

Here, λ represents the penalty coefficient. We use different weights for different models to balance the forgetting effect and the model's performance:

$$Loss = \lambda \mathbb{S}(f_\omega(x), f_\theta(x)) + (1 - \lambda)\ H_{CE}\ (f_\omega(x'), Y). \tag{3}$$

In other words, if the following condition holds:

$$\mathbb{S}(f_\omega(D_f), f_\theta(D_f)) \approx 0, \tag{4}$$

then we consider the forgetting process to be complete, where $\mathbb{S}$ is a distance function (e.g., KL distance) to evaluate the similarity between $f_\omega(D_f)$ and $f_\theta(D_f)$.

C. Reference Model

In our method, the reference model is defined as follows. Given a dataset D, with the data to be forgotten denoted as D_f and the remaining data as D_r, where $D_r \cup D_f = D$. A model f_θ is trained on D_r, with parameters θ. We assume that the model f_θ follows a posterior distribution P such that $h_\theta(D_r) \sim Q$. If another model f_σ satisfies $f_\sigma(D_r) \sim Q'$ and $KL(Q, Q') \approx 0$, then f_σ is considered a reference model.

During the forgetting phase, we input the data to be forgotten into the target model and measure the similarity between the resulting distribution and the distribution obtained from the reference model. This process aims to gradually align the two distributions, ultimately achieving the effect of forgetting. In other words, we treat forgetting as a process of gradually transitioning from recognizing the data to be forgotten to not recognizing it. Therefore, the direction of not recognizing the data to be forgotten, represented by the reference model, is crucial.

The selection of the reference model significantly impacts the outcome of the target model's forgetting process. To ensure that the target model forgets the data to be erased, its parameters must be fine-tuned toward the direction of not recognizing the data. Consequently, the reference model must be trained on a dataset that is disjoint from the data to be forgotten, ensuring effective forgetting.

Furthermore, to preserve the target model's original accuracy after forgetting, it is important to choose a reference model whose weight distribution is not too dissimilar from the target model's. A reference model with a vastly different weight distribution may lead to the target model successfully forgetting the data but suffering catastrophic damage to its original performance.

The optimal reference model would ideally be one trained on the remaining data that does not need to be forgotten. However, retraining such a model can be time-consuming. Therefore, in this paper, we select a subset of the remaining data and use it to train the reference model. This approach achieves the desired forgetting effect without incurring significant time costs.

IV. Unlearning Evaluation Methods

To assess the success of our machine unlearning approach, in this section we introduce two effective unlearning eval-

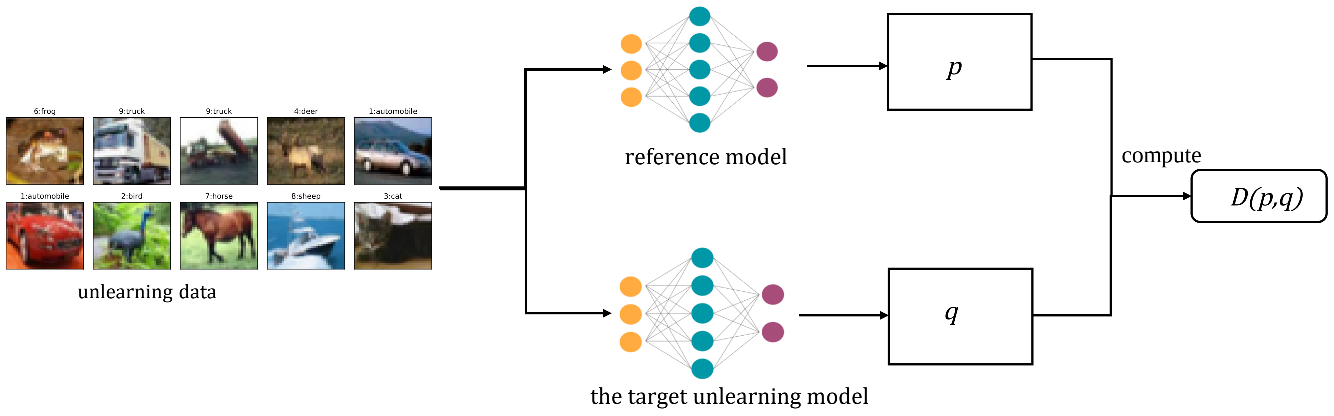


FIGURE 1: System framework of our machine unlearning method.

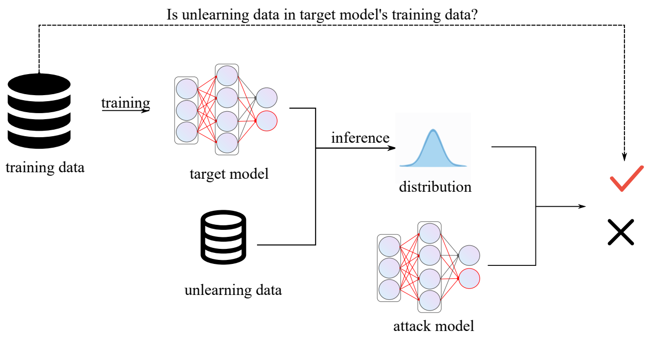


FIGURE 2: Membership inference evaluation.

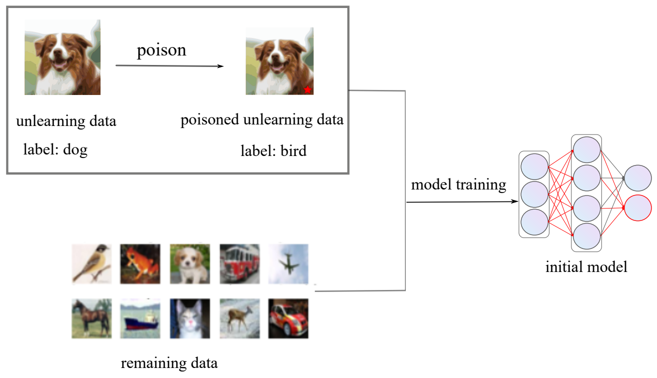


FIGURE 3: Backdoor injection.

uation methods, called membership inference attack-based unlearning evaluation and backdoor attack-based unlearning evaluation.

A. Membership Inference Attack-based Unlearning Evaluation

In our experiments, we evaluate the extent to which the forgotten model retains information about the customer data that needs to be erased by utilizing membership inference attacks. Since the attack classifier is trained on data derived from the original global model, it can accurately distinguish information related to the data to be forgotten. The worse the performance of the membership inference attack, the less the forgotten data impacts the global model. To execute membership inference attacks on the forgotten model, we employ a shadow model training strategy to infer the data and construct the attack classifier. The goal of the membership inference attack is to determine whether a given dataset was used to train a given machine learning (ML) model. Therefore, the performance of the membership inference attack can measure how much information remains in the forgotten global model. We use the attack accuracy on the customer data that needs to be forgotten as an indicator. This metric represents the proportion of the forgotten data that was likely involved in training the target model. In other words, the attack accuracy measures the degree of privacy leakage of the data that should be forgotten. In our method, we use the membership inference attack [19] to assess how much forgotten information remains in the model after forgetting. In [19], the membership inference attack model is trained based on the original model before forgetting, allowing it to accurately distinguish between learned or forgotten data by judging whether it is member data according to the difference between the posterior distribution of the input sample and that of the remaining data.

B. Backdoor Attack-based Unlearning Evaluation

Backdoor attacks, as a common attack method, also help evaluate the effectiveness of forgetting [24]. The concept of backdoor attacks was first proposed in [25], [25]. The attack process is mainly divided into two parts. First, in the data preprocessing stage before model training, the attacker selects a portion of the data from the complete and clean training set to implant a backdoor and then uniformly modifies it to a specific label, thereby completing data poisoning. In the second part, during the training stage, the target model is trained using both the clean, unpoisoned data and the

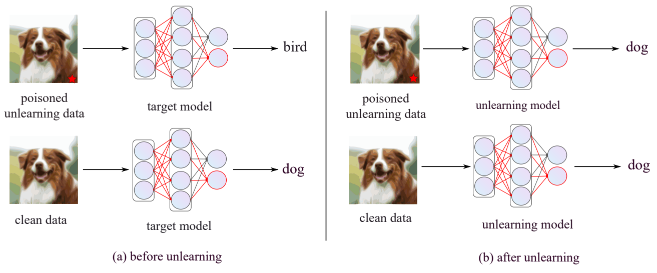


FIGURE 4: Backdoor evaluation.

poisoned data, allowing the model to learn and remember the special backdoor information, completing the entire poisoning process. During the inference stage, clean, unpoisoned data yields normal prediction results, but since the infected model has already remembered the backdoor information, when the same backdoor is detected, the prediction result is the specific label preset by the attacker. That is, due to the memory of the backdoor, the performance of the poisoned data on the target model is affected. Applying this principle to our method, as shown in Figure 3, we first tag the data that needs to be forgotten with a backdoor and uniformly change its label to a specific one. In the second step, this data is combined with the remaining clean data that does not need to be forgotten to train the target model, completing the poisoning process. During the inference stage, as shown in Figure 4, due to the target model's memory of the backdoor, the poisoned data that needs to be forgotten will yield inaccurate prediction results, i.e., the label we preset in advance. When our forgetting method is applied to the infected model that has remembered the backdoor data, ideally, the expected forgetting effect is achieved, and the backdoor information is completely forgotten by the target model. At this point, the model becomes a healthy one that does not remember the backdoor information, and during the inference stage, it can produce accurate prediction results for the forgotten poisoned data, i.e., its original label.

V. Experimental Results

In this section, we evaluation our machine unlearning method across different datasets to confirm its effectiveness.

A. Experimental Settings

1) Experiment Datasets and Model Architectures

In the experiment, we evaluate our method across different datasets, including MNIST, Fashion-MNIST, SVHN, and CIFAR-10. We tested the effectiveness of our approach on various models, including multilayer perceptron (MLP), LeNet, ResNet-18, and VGG16, as shown in Table 1.

1) The MNIST dataset is a benchmark for segmentation and consists of centered, handwritten grayscale images [26]. We used 60,000 training examples and 10,000 test examples, with each image being 28×28 pixels in size. For the target model, we selected LeNet-5 [26], a network architecture comprising 2 convolutional layers (CONV), 1 pooling layer (POOL), and 1 fully connected layer (FC), which is highly efficient for handwritten character recognition.
2) The Fashion-MNIST dataset contains Zalando article images [27], including a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 grayscale image associated with labels from 10 categories, representing different clothing items. The dataset shares the same image size and training/test split structure as MNIST. For this dataset, we used a multilayer perceptron as the target model.
3) The SVHN (Street View House Numbers) dataset [28] is derived from Google's Street View images. The training set consists of 73,257 three-channel color images of size 32×32, representing the numbers 1 to 10, while the test set contains 26,032 images. For the SVHN dataset, we adopted a multilayer perceptron as the target model, comprising 8 convolutional layers, 1 pooling layer, and 1 fully connected layer.
4) CIFAR-10 is a benchmark dataset for evaluating image recognition algorithms. It consists of 60,000 three-channel color images of size 32×32, divided into 10 categories such as "airplane," "dog," and "cat." CIFAR-10 is a balanced dataset with 6,000 randomly selected images per category. We used 50,000 images for training and 10,000 for testing. For CIFAR-10, we used ResNet-18 [29] as the target model, simply recorded as 'CIFAR10.R'. To better compare the effects of different models, we also used VGG-16 [30] as an additional target model for the CIFAR-10 dataset, simply recorded as 'CIFAR10.V'.

For each machine unlearning evaluation, the size of the forgotten data samples was 1/100 of the total number of samples in the dataset, which amounted to 600, 600, 732, and 500 samples for each dataset, respectively. The default parameter settings are listed in Table 2. We evaluate the unlearning performance with several metrics such as unlearning accuracy, unlearning time cost, membership inference evaluation, and backdoor evaluation.

B. Unlearning Accuracy

Accuracy is a critical indicator of target model training performance. To implement machine unlearning effectively without significantly impacting the performance of the target model, it is essential to evaluate accuracy both before and after the unlearning process. Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{P + N}, \tag{5}$$

where the number of correctly classified samples is divided by the total number of samples. The following terms are used:

TABLE 1: Datasets and models architectures.

Dataset	Model architecture	Number of instances	Classes
MNIST	LeNet	60,000	10
Fashion-MNIST	MLP	60,000	10
SVHN	MLP	73,257	10
CIFAR10.R	ResNet-18	50,000	10
CIFAR10.V	VGG-16	50,000	10

TABLE 2: Default parameter settings.

dataset	model architecture	number of instances	value of λ	number of instances of the reference model
MNIST	LeNet	60,000	0.01	6,000
Fashion-MNIST	MLP	60,000	0.01	6,000
SVHN	MLP	73,257	0.01	7,325
CIFAR10.R	ResNet-18	50,000	0.01	5,000
CIFAR10.V	VGG-16	50,000	0.01	5,000

1) True Positives (TP): The number of instances that are correctly classified as positive cases, i.e., cases that are actually positive and are classified as such by the model.
2) False Positives (FP): The number of instances that are incorrectly classified as positive by the model.
3) False Negatives (FN): The number of instances that are actually positive but are incorrectly classified as negative by the model.
4) True Negatives (TN): The number of instances that are correctly classified as negative, i.e., cases that are actually negative and classified as such by the model.

The accuracy of different datasets after unlearning is illustrated in Figure 5. For the MNIST dataset, the first column shows an accuracy of 98.97% before unlearning. After retraining following the deletion of 1/100 of the samples, the second column shows an accuracy of 98.96%. Since the reduction in sample size is minimal, the third column indicates an accuracy of 98.12% after applying our unlearning method. This minor decrease in accuracy, when compared to the retrained model, demonstrates that our method does not significantly affect the performance of the target model.

For the CIFAR-10 dataset using the ResNet architecture, the accuracy before unlearning is 90.87% (first column). After retraining, the accuracy drops by 4%, primarily due to the reduced sample size. With our unlearning method, the accuracy is 89.64%, slightly higher than that of the retrained model. A similar trend is observed for the CIFAR-10 dataset using VGG16, where the overall accuracy difference before and after unlearning is less than 3% compared to ResNet. This disparity arises because VGG16's architectural limitations, in comparison to ResNet18, influence its training performance, though the unlearning method remains unaffected.

In Figure 5, it is evident that for the SVHN, CIFAR10.R, and CIFAR10.V datasets, *the accuracy of our method after unlearning is higher than that of the retrained model*. We believe that this outcome is primarily due to *uneven sampling*. As illustrated in Figure 6, all points in the figure represent the entire dataset prior to unlearning. The blue dots correspond to correctly classified samples (a total of 80), while the orange dots represent incorrectly classified samples (a total of 20), yielding an overall accuracy of 80%. The data marked for forgetting, obtained through random sampling, may be distributed within a circle. This subset includes 9 correctly classified samples and 1 incorrectly classified sample, resulting in a classification accuracy of 90%. After removing this portion of the data and retraining the model on the remaining samples, the model has 71 correctly classified samples and 19 incorrectly classified samples, resulting in a classification accuracy of 78.8%. This demonstrates that random sampling can lead to a higher accuracy after unlearning compared to retraining, and in some cases, the accuracy may even exceed the pre-unlearning accuracy.

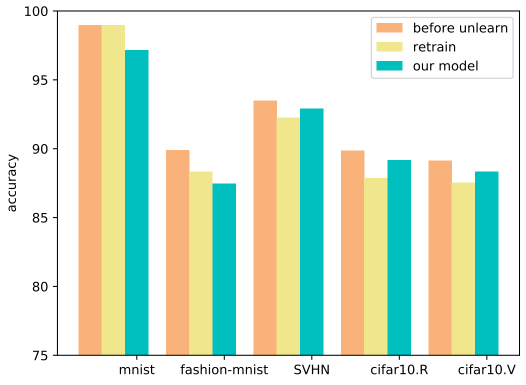


FIGURE 5: Comparison of accuracy across different datasets before and after unlearning.

TABLE 3: Comparison of unlearning time cost across different datasets.

Datasets	Unlearning instances	Our model			Retraining (s)	Speedup
		Training (s)	Training for reference model (s)	Total (s)		
MNIST	600	3.81	42.81	46.62	750.69	16.10×
Fashion-MNIST	600	10.42	67.59	70.01	1283.21	16.66×
SVHN	732	11.03	140.29	151.31	2371.02	15.67×
CIFAR10.R	500	22.51	143.70	166.21	2512.34	15.11×
CIFAR10.V	500	23.33	141.95	165.28	2753.96	16.66×

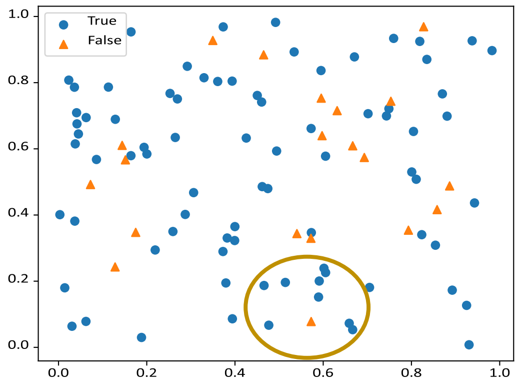


FIGURE 6: Uneven data sampling for unlearning.

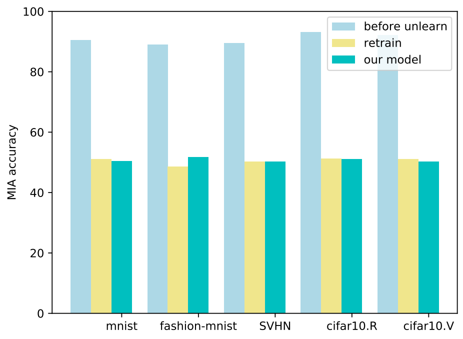


FIGURE 7: Comparison of membership inference evaluation results across different datasets before and after unlearning.

C. Unlearning Time Cost

In addition to performance, efficiency is a critical factor in data unlearning. The objective is to achieve nearly identical results to retraining but in significantly less time. Using the retraining model as a baseline, our method can reduce the time cost by an order of magnitude. As shown in Table 3, for the MNIST dataset, where the data to be forgotten represents 1/100 of the total dataset (600 items), our method requires two iterations to converge, consuming only 3.81 seconds. Training the reference model takes 42.81 seconds, bringing the total time to 46.62 seconds. In comparison, retraining the model after deleting the forgotten data takes 750.69 seconds. Thus, our method offers a speedup of 16.10 times. This trend is consistent across other datasets as well.

D. Membership Inference Evaluation Results

In the inference stage, the data to be forgotten is input into the attack model, which then deduces whether it belongs to the member data of the target model based on the output distribution. If the accuracy of membership inference is too high or too low, it indicates that the attack model has a higher confidence in classifying the forgotten data as either member or non-member data.

Before unlearning, the forgotten data is part of the target model's member data, meaning its posterior distribution closely resembles other member data. Consequently, the attack model can easily identify it as member data, as shown in Figure 7, where the accuracy reaches 90%. This implies that the attack model correctly identifies the forgotten data as member data 90% of the time.

In the retraining model, the forgotten data is no longer part of the member data. Due to a significant difference in the posterior distribution, the attack model struggles to determine whether it is member data, resulting in an accuracy rate of around 50%.

After applying our unlearning method, the accuracy of the attack remains comparable to that of the retraining method. This indicates that the attack model finds it challenging to clearly classify the forgotten data as either member or non-member. In other words, for the forgotten model, the target unlearning model no longer remembers the information of the data to be forgotten, similar to the retraining model.

E. Backdoor Evaluation Results

To determine whether unlearning is complete, we utilize a backdoor attack to assess the effectiveness of the unlearning process. In the experiment, we first embedded the data to be forgotten into a backdoor, assigning fixed labels to this data, while the remaining normal data was used for training the target model. At this stage, the trained target model has "remembered" the backdoor data (i.e., the data to be forgotten). As a result, the model's predictions for 98% of the backdoor data align with the fixed labels, yielding a test accuracy of up to 98% for this subset.

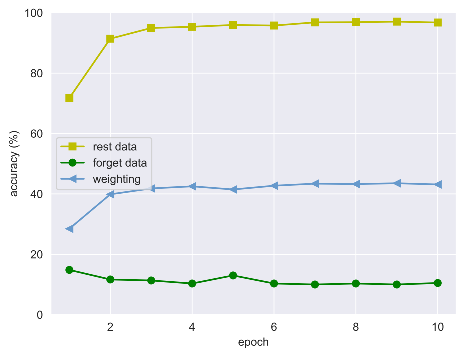


FIGURE 8: Comparison of backdoor evaluation result for the MNIST dataset before and after unlearning.

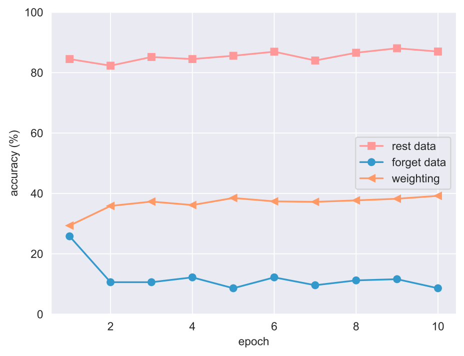


FIGURE 9: Comparison of backdoor evaluation result for the CIFAR10.R dataset before and after unlearning.

The next step involves applying our unlearning method to the target model. After unlearning, when the backdoor data is reintroduced for inference, the accuracy drops to below 20%. This demonstrates that for 80% of the backdoor data, the model's predictions no longer match the fixed labels. The reduction in test accuracy from 98% to 20% clearly indicates that our method successfully made the model forget the backdoor information that had been memorized during training, thus validating the success of the unlearning process.

As shown in Figure 8, during the unlearning process on the MNIST dataset, the test accuracy for the data that does not need to be forgotten remains high, reflecting that the accuracy for the remaining data after unlearning also stays at a high level. For the data to be forgotten, the test accuracy drops from 98% to a significantly lower level, demonstrating the effectiveness of the unlearning. To balance unlearning performance with post-unlearning accuracy, we assign equal weights of 0.5 to each factor. The higher the resulting score, the better the overall performance, as illustrated by the blue line in Figure 8. A similar trend can also be observed for the CIFAR10.R dataset, as shown in Figure 9.

F. Impact of the Parameter λ

We use the precision penalty term to constrain the precision loss of the target model after unlearning, where λ represents the coefficient of the standard unlearning term, and $1 - \lambda$ denotes the penalty term coefficient. To achieve optimal unlearning effects without significant loss in training accuracy, it is essential to balance these two terms. If λ is too large, the loss of precision will be substantial; conversely, if λ is too small, the effectiveness of unlearning will be diminished.

In the experiment, we evaluated the impact of different λ values on unlearning performance. As illustrated in Figure 10 (a), using the MNIST dataset as an example, the accuracy of the target model decreases notably when λ is set to 1 or 0.1. However, when λ is set to 0.01, 0.001, or 0.0001, the accuracy remains stable. As shown in Figure 10 (b), the backdoor evaluation result is lowest when λ is 0.01, indicating the best unlearning effect. To balance accuracy and unlearning effectiveness, a λ value of 0.01 is considered to the most suitable. We can also infer the optimal λ values for other datasets. For example, as shown in Figure 11, the optimal λ value is 0.001 for the CIFAR10.R dataset.

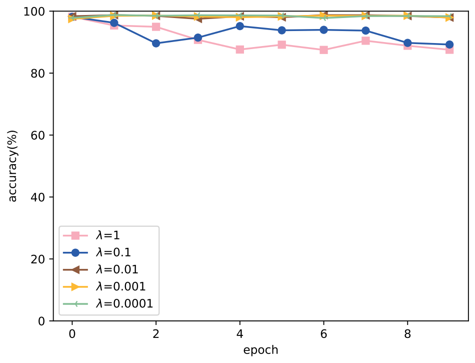
G. Comparison of Posterior Distributions

To verify that the machine unlearning process using our method does not adversely affect the performance of the model itself, we evaluated the posterior distributions of various datasets before and after unlearning, along with the weight distributions of the target model. As shown in Figure 12 (a), the output distributions of the MNIST dataset before (yellow line) and after (green line) unlearning nearly overlap. Similarly, as depicted in Figure 12 (b), the output distributions of the CIFAR10.R dataset before (blue line) and after (red line) unlearning are also closely aligned. This indicates that using our method, the performance of the target model remains nearly unaffected before and after unlearning.

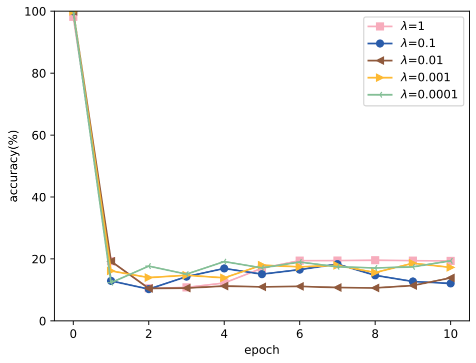
H. Impact of the Reference Model Construction

To eliminate the influence of forgotten data on the target model, we aim to iteratively adjust the target unlearning model towards the direction of the model retrained on D_r. Considering the time cost, we select a subset D_s from D_r and generate a reference model $M_{reference}$ by training on D_s.

As shown in Figure 13 (a), the posterior distribution of the retrained model on the MNIST dataset almost coincides with that of the reference model. The peaks are concentrated around 0.5 and 1, as the output values of the LeNet model used for MNIST are positive due to the ReLU function. Values of 0 are transformed to 0.5 after passing through the sigmoid function, while values of 1 remain unchanged. Similarly, Figure 13 (b) shows that the posterior distribution of the retraining model is also very close to that of the reference model on the CIFAR10.R dataset. This indicates that the training direction of the reference model aligns with that of the retraining model, validating our approach of using

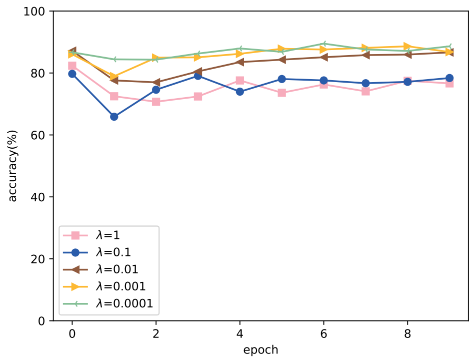


(a) Unlearning accuracy.

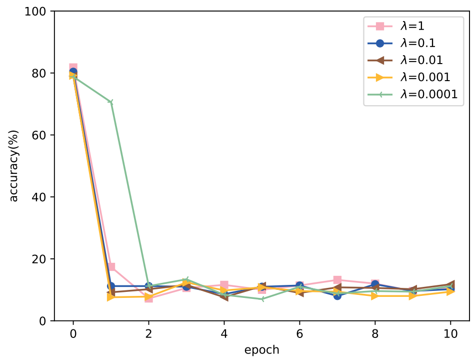


(b) Backdoor evaluation result.

FIGURE 10: Influence of different λ values for the MNIST dataset.

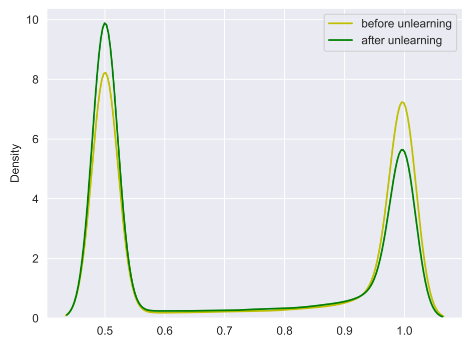


(a) Unlearning accuracy.

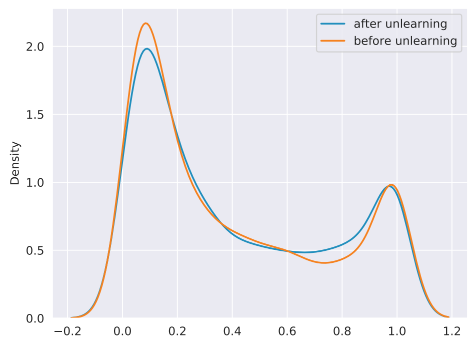


(b) Backdoor evaluation result.

FIGURE 11: Influence of different λ values for the CIFAR10.R dataset.



(a) MNIST



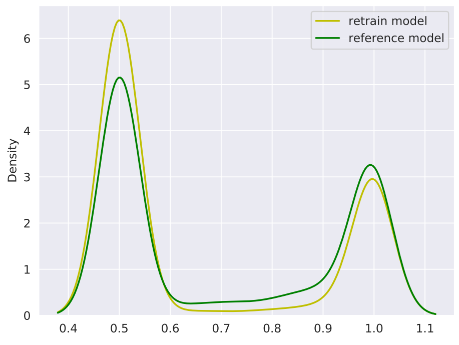
(b) CIFAR10.R

FIGURE 12: Posterior distributions of the initial models and unlearning models across different datasets.

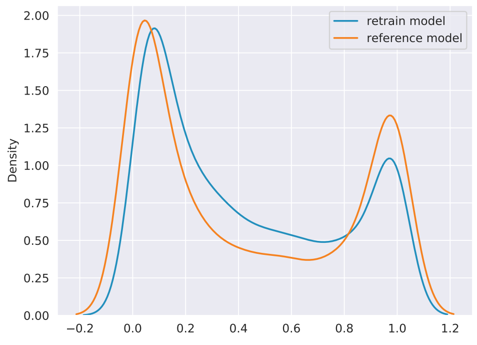
the reference model to guide unlearning and reduce time cost.

VI. Related work

Machine unlearning: Recently, machine unlearning has become an emerging research area. Existing work includes



(a) MNIST



(b) CIFAR10.R

FIGURE 13: Comparison of the output distributions in the retraining model and the reference model across different datasets.

the method proposed by Cao [9], which utilizes statistical queries to obtain dataset features rather than directly training them, thereby enabling unlearning in less time than traditional retraining. However, this statistical query method is only applicable to non-adaptive machine learning models where later training does not depend on earlier training. Liu [10] proposed a data unlearning method based on the federated learning framework, which involves saving the update parameters of each round during the normal training model aggregation stage. Although this approach reduces training time, it also requires more storage space due to the need to save parameters. Additionally, because the saved parameters themselves carry information to be forgotten, this method theoretically cannot guarantee complete unlearning. Ginart et al. [11] studied the problem of retraining models after removing data to be forgotten, particularly applied to the k-means algorithm, which is difficult to apply to supervised learning. Izzo et al. [12] primarily studied unlearning data in linear computation and achieved linear scaling on data dimensions. [31] proposed a linear transformation method applied to classifiers to re-scale the logarithmic probabilities predicted by the classifier, but this method does not eliminate the relevant information's contribution to the model. [32], based on differentiable convex functions (e.g., logistic regressors), proposed an unlearning method that uses Newton's method to remove weight information and differential privacy to mask unlearning residuals. The study of unlearning methods in Bayesian models [33] introduces new techniques for adjusting likelihood and reverse KL divergence. [34] uses gradient descent to remove data from a convex function.

In comparison, our method is highly applicable to neural networks. We introduce a small perturbation to the parameters of the original target model, take the KL divergence of the posterior probability between the target model and the reference model as the loss function, and add a precision penalty term to avoid catastrophic unlearning. This approach does not alter the target model or require additional storage space, resulting in relatively low time and space costs and minimal loss of model accuracy.

Membership inference attack: Our work is also inspired by membership inference attacks to evaluate the effects of unlearning. Membership inference attacks [19], [35], [36], [37], [38] address the question of whether a specific sample was used to train a machine learning model. The method proposed by Song et al. [39] can determine whether a user's data participated in the training of a machine learning model, but it requires multiple shadow models and datasets, resulting in high time and space costs. Later, [35] suggested that training multiple shadow models is unnecessary, and the performance of membership inference attacks is not affected by the number of shadow models. [40], [41] argue that there are few methods to evaluate machine unlearning, and membership inference attacks are well-suited for this purpose. In this paper, we also apply membership inference attacks to assess the amount of residual information after machine unlearning.

Backdoor attack: Backdoor attacks are another evaluation metric in this paper. [24] selects a subset of the training set, implants a backdoor into these samples by setting a small area to zero in terms of color or brightness, and uniformly modifies the corresponding labels to a specific label. The target model is then trained using both the backdoored and original data. The target model ultimately learns to associate the originally set specific label with the backdoor. Tang et al. [42] designed a method that does not require prior training or modification of the samples themselves. Instead, a small Trojan horse module is inserted into the model without altering the original model's parameters. Liu et al.'s [43] research considers that modifying training data and its labels can be easily detected by input filtering defense strategies. Further solutions discusses the backdoor attack in the federated learning and contrastive leaning paradigms [44]. Their approach is more natural and does not require label modifications. Through the mathematical modeling of the physical reflection model, the radiation image of the object is used as the backdoor implantation model.

VII. Conclusion and Future Work

Machine unlearning aims to completely eliminate the contribution of information that needs to be forgotten from a model. This paper proposes a method to address this issue without requiring model retraining, significantly reducing time and computational costs. The approach is particularly well-suited for neural network models. By utilizing a reference model that excludes the unlearning information, the target model is iteratively adjusted until the contribution of the forgotten data is fully removed. Our method holds substantial promise for privacy protection and can be applied to various interesting areas, including rapid adjustment of model parameters, malicious contribution erasure, model defense, and data transactions and sharing.

REFERENCES

[1] P. Yang, X. Deng, L. Wang, S. Lin, J. Gui, X. Chen, S. Wan, and Y. Qian, "Energy-efficient symbiotic uav-enabled mec networks via ris: Joint trajectory and phase-shift control optimization," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2024.

[2] L. Zhao, C. Mao, S. Wan, A. Hawbani, A. Y. Al-Dubai, G. Min, and A. Y. Zomaya, "Cast: Efficient traffic scenario inpainting in cellular vehicle-to-everything systems," *IEEE Transactions on Mobile Computing*, pp. 1–15, 2024.

[3] P. Jiang, X. Deng, W. Wu, L. Lin, X. Chen, C. Chen, and S. Wan, "Weather-aware collaborative perception with uncertainty reduction," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2024.

[4] K. Chen, W. Wang, Z. Wang, Y. Huang, Y. Xiao, W. Zhang, Z. Li, Z. Guo, Z. Luo, L. Yin, H. Mai, X. Wang, and Q. Yang, "Private data leakage in federated contrastive learning networks," *IEEE Open Journal of the Communications Society*, pp. 1–1, 2024.

[5] W. Lan, K. Chen, Y. Li, J. Cao, and Y. Sahni, "Deep reinforcement learning for privacy-preserving task offloading in integrated satellite-terrestrial networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 10, pp. 9678–9691, 2024.

[6] W. Lan, K. Chen, J. Cao, Y. Li, N. Li, Q. Chen, and Y. Sahni, "Security-sensitive task offloading in integrated satellite-terrestrial networks," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2024.

[7] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, 2011.

[8] K. Chen, Y. Huang, Y. Wang, X. Zhang, B. Mi, and Y. Wang, "Privacy preserving machine unlearning for smart cities," *Annals of Telecommunications*, vol. 79, no. 1, pp. 61–72, 2024.

[9] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," pp. 463–480, 2015.

[10] G. Liu, Y. Yang, X. Ma, C. Wang, and J. Liu, "Federated unlearning," *arXiv preprint arXiv:2012.13891*, 2020.

[11] A. Ginart, M. Y. Guan, G. Valiant, and J. Zou, "Making AI forget you: Data deletion in machine learning," pp. 3513–3526, 2019.

[12] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 130. PMLR, 2021, pp. 2008–2016.

[13] P. Ye, W. Wang, B. Mi, and K. Chen, "Edgestreaming: Secure computation intelligence in distributed edge networks for streaming analytics," *ACM Transactions on Multimedia Computing, Communications, and Applications*, pp. 1–1, 2024.

[14] K. Chen, X. Zhang, X. Zhou, B. Mi, Y. Xiao, L. Zhou, Z. Wu, L. Wu, and X. Wang, "Privacy preserving federated learning for full heterogeneity," *ISA Transactions*, vol. 141, pp. 73–83, 2023.

[15] D. CeArley, B. Burke, S. Searle, and M. J. Walker, "Top 10 strategic technology trends for 2021," *The Top*, vol. 10, pp. 1–246, 2020.

[16] K. A. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. ACM, 2017, pp. 1175–1191.

[17] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1387–1395.

[18] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," 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. 14 747–14 756.

[19] 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 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 3–18.

[20] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, p. 3152676, 2017.

[21] L. Determann, Z. J. Ruan, T. Gao, and J. Tam, "China's draft personal information protection law," *Journal of Data Protection & Privacy*, vol. 4, no. 3, pp. 235–259, 2021.

[22] D. Zhou, W. Chen, K. Chen, and B. Mi, "Fast and accurate snn model strengthening for industrial applications," *Electronics*, vol. 12, no. 18, pp. 1–12, 2023.

[23] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," pp. 141–159, 2021.

[24] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *CoRR*, vol. abs/1708.06733, 2017.

[25] K. Chen, H. Zhang, X. Feng, X. Zhang, B. Mi, and Z. Jin, "Backdoor attacks against distributed swarm learning," *ISA Transactions*, vol. 141, pp. 59–72, 2023.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: http://arxiv.org/abs/1708.07747

[28] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:16852518

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 770–778, 2016.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[31] T. Baumhauer, P. Schöttle, and M. Zeppelzauer, "Machine unlearning: linear filtration for logit-based classifiers," *Mach. Learn.*, vol. 111, no. 9, pp. 3203–3226, 2022.

[32] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten, "Certified data removal from machine learning models," *arXiv preprint arXiv:1911.03030*, 2019.

[33] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Variational bayesian unlearning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[34] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," *arXiv preprint arXiv:2007.02923*, 2020.

[35] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[36] C. Wang, G. Liu, H. Huang, W. Feng, K. Peng, and L. Wang, "Miasec: Enabling data indistinguishability against membership inference attacks in mlaas," *IEEE Transactions on Sustainable Computing*, vol. 5, no. 3, pp. 365–376, 2019.

[37] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adver-

sarial examples," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 259–274.

[38] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," *IEEE Transactions on Services Computing*, 2019.

[39] C. Song and V. Shmatikov, "Auditing data provenance in text-generation models," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 196–206.

[40] D. M. Sommer, L. Song, S. Wagh, and P. Mittal, "Towards probabilistic verification of machine unlearning," *arXiv preprint arXiv:2003.04247*, 2020.

[41] A. Sablayrolles, M. Douze, C. Schmid, and H. Jégou, "Radioactive data: tracing through training," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8326–8335.

[42] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. ACM, 2020, pp. 218–228.

[43] Y. Liu, W. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 1265–1282.

[44] Y. Huang, K. Chen, J. Cao, J. Shen, S. Wang, Y. Peng, W. Peng, and K. Cai, "BAGEL: backdoor attacks against federated contrastive learning," *CoRR*, vol. abs/2311.16113, 2023.