



# Privacy preserving machine unlearning for smart cities

Kongyang Chen<sup>1,2,3</sup> · Yao Huang<sup>1</sup> · Yiwen Wang<sup>1</sup> · Xiaoxue Zhang<sup>4</sup> · Bing Mi<sup>5</sup> · Yu Wang<sup>1</sup>

Received: 6 September 2022 / Accepted: 23 March 2023 / Published online: 29 June 2023  
© Institut Mines-Télécom and Springer Nature Switzerland AG 2023

## Abstract

Due to emerging concerns about public and private privacy issues in smart cities, many countries and organizations are establishing laws and regulations (e.g., GDPR) to protect the data security. One of the most important items is the so-called *The Right to be Forgotten*, which means that these data should be forgotten by all inappropriate use. To truly forget these data, they should be deleted from all databases that cover them, and also be removed from all machine learning models that are trained on them. The second one is called *machine unlearning*. One naive method for machine unlearning is to retrain a new model after data removal. However, in the current big data era, this will take a very long time. In this paper, we borrow the idea of Generative Adversarial Network (GAN), and propose a fast machine unlearning method that unlearns data in an adversarial way. Experimental results show that our method produces significant improvement in terms of the forgotten performance, model accuracy, and time cost.

**Keywords** Machine unlearning · Privacy protection · Generative adversarial network · Membership inference attack

## 1 Introduction

In smart cities, many sensors continuously collect large quantities of raw data for further data analysis to support a variety of vertical applications [11]. For example, in the intelligent transport area, multiple sensors (e.g., GPS sensors, accelerometer sensors, magnetic sensors, laser radars, cameras) from crowded clients are collected and trained to support practical applications such as road surface quality monitoring [10], traffic congestion alleviation [23], and location-based services [8, 9]. As a powerful dataanalysis

tool, deep neural networks are also often used to better investigate the data characteristics in these applications.

Recently, security and privacy issues have attracted increasing attention in smart cities [5, 7]. To better protect the public and private data privacy, many countries and organizations have established laws and regulations such as GDPR [33] and CCPA [6]. Recently, a new concept of *The Right to be Forgotten* was introduced in this area, which means that data owners retain the ownership of their personal data, and they have the right to ask data users to forget their data. Generally, there are two meanings of data forgotten, namely *storage forgotten* and *model forgotten*. The first one means that the users should delete the original data (as well as variants) from their database, and the second one indicates that the users should erase the data information from all models which are previously trained with these data.

The model forgotten is indeed an essential task to protect the data privacy. Many studies have shown that machine learning models, especially over-parameterized models, could remember much information about the original training data [32, 37]. Malicious attackers can thus leverage model inversion attack [30] or data reconstruction attacks [42, 44] to recover the training data, or utilize membership inference attacks [3, 34] to discriminate whether a certain data are used to train the model. Obviously, these

✉ Kongyang Chen  
kychen@gzhu.edu.cn

<sup>1</sup> Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou 510006, China  
<sup>2</sup> Pazhou Lab, Guangzhou 510330, China  
<sup>3</sup> Jiangsu Key Laboratory of Media Design and Software Technology, Jiangnan University, Wuxi 214122, China  
<sup>4</sup> School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China  
<sup>5</sup> School of Public Finance and Taxation, Guangdong University of Finance and Economics, Guangzhou 510320, China

attacks will cause a major data leakage risk with a negative data protection and privacy implications.

To achieve the model forgotten, *Machine Unlearning* [39] has become a new research hotspot. Different from classic machine learning, in machine unlearning, it needs to remove a part of the trained data from a pre-trained model. A naive method for machine unlearning is to retrain a new model from the remaining data, which will not have any impact on the data to be deleted. In complex scenarios, however, model retraining creates a huge computational complexity and time cost. Thus, we aim to provide a computation efficient way for machine unlearning. There are several statistical data analysis based machine unlearning methods. For example, in [39], it introduced a summation term between model and raw dataset, where the model was trained on the summation term rather than raw data. Thus, to achieve model forgotten, it simply removed the data from the raw dataset and formed a new summation term to retrain a new model. In [27], it partitioned the original dataset into many non-overlapping data chunks. Each data chunk trains its own model independently, and the global model was integrated by individual models from these chunks. For model forgotten, only a small data chunk should be retrained to save computation complexity. Recently, AI-based methods have also been presented to support more practical scenarios. For example, in [14], in a Federated learning scenario, it assumed that all training parameters were stored, and the model unlearning process can be accelerated with a large iteration step by following the previous intermediate parameters [27]. In [22], it studied how to remove data from a pre-trained model with a classic machine learning method, such as random forests. However, these existing solutions have a strong assumption with limited application scenarios. For example, the statistic data analysis based machine unlearning methods are very inefficient with data forgotten from multiple summation terms or data chunks, and the AI-based methods suffer from a poor performance when the data to be forgotten has a distinct data distribution with the global one.

In this paper, we aim to provide a novel machine unlearning method in deep neural networks to tackle the privacy and security concerns in smart cities. We build a general problem formulation for machine unlearning, where its main objectives are to forget a small part of raw data from a pre-trained model and minimize the model performance degradation as much as possible. We also present a novel data forgotten idea to compare the model performance with a third-party data. After that, we propose a Generative Adversarial Networks (GAN) based model unlearning solution for fast deployment. Finally, we introduce a model unlearning evaluation method with membership interference attack to judge whether the target data is indeed forgotten in the final model. Compared with existing solutions, our approach does not need

the prior knowledge about the intermediate training parameters, and there is no limitation for the data to be forgotten. Our approach can be applied to most existing machine learning frameworks without any modification.

The main contributions of our paper can be summarized as follows:

- We introduce a third-party data and build a generic machine unlearning framework, where its main objectives are to produce similar model performance with the third-party data, while minimize as much of the model performance degradation as possible.
- We propose a Generative Adversarial Networks (GAN) based model unlearning method for fast deployment. We also present a sort function for a better discriminator design.
- We evaluate our approach with different datasets and different model architectures. Experimental results show that our approach achieves high model performances with an extremely fast model training speed (e.g.,  $3.3 \times$  to  $22.6 \times$  improvement than the standard retrain method).

The remainder of this paper is organized as follows: Section 2 gives the problem formulation of the machine unlearning. Section 3 proposes our GAN based machine unlearning method. Section 4 shows the experimental results of our method. Section 5 discusses the related work. Finally, Section 6 concludes this paper.

## 2 Problem formulation

In machine unlearning, the misused dataset should be removed from all machine learning models which are trained from them. Formally, let  $M_{init}$  be a machine learning model trained from a dataset  $D$ , i.e.,  $D \rightarrow M_{init}$ . Let  $D_f$  be the dataset to be forgotten which is a subset of  $D$ , i.e.,  $D_f \in D$ . Our target is to build a new model  $M$  from the constructed dataset  $D_r = D - D_f$ , i.e.,  $D_r \rightarrow M$ . In current big data era, it will take a very long time to retrain  $M$  directly from  $D - D_f$ . Thus, we would like to ask one question: can we determine  $M$  rapidly if we exploit the pre-trained  $M_{init}$ ?

To answer this question, we must rethink the meaning of data forgotten. *If one dataset is forgotten by a model, it will perform similarly to a third-party dataset in this model.* Note that the third-party dataset should not be the training set or the test set of this model. Let  $D_{nonmember}$  be a third-party dataset with the same distribution with  $D$ , where  $D_{nonmember} \cap D = \phi$ .

Suppose we can develop an algorithm  $\mathcal{U}$  to determine  $M$  rapidly from the pre-trained  $M_{init}$ , that is

$$\mathcal{U}(M_{init}, D_f) = M. \quad (1)$$

After data forgotten,  $D_f$  will have the similar output distributions of the model  $M$  with  $D_{nonmember}$ , that is

$$M(D_{nonmember}) \approx M(D_f). \quad (2)$$

As  $D_{nonmember}$  is a third-party dataset for both  $M$  and  $M_{init}$ , and  $M$  and  $M_{init}$  have the same network architecture (e.g., ResNet), their output distributions for  $D_{nonmember}$  should also be very similar, that is

$$M(D_{nonmember}) \approx M_{init}(D_{nonmember}). \quad (3)$$

Considering Eqs. (2) and (3), we have

$$M(D_f) \approx M_{init}(D_{nonmember}). \quad (4)$$

It means that the output distributions of the model  $M$  with  $D_f$  will be very similar to the output distributions of the model  $M_{init}$  with  $D_{nonmember}$ . We can further rewrite the similarity function as follows:

$$d(M(D_f), M_{init}(D_{nonmember})) \approx 0, \quad (5)$$

where  $d$  can be a distance measurement function, such as KL divergence and JS divergence.

According to [34], the data size of  $D_f$  is usually much smaller (e.g., by several magnitudes) than  $D_r$ , so our machine unlearning speed is much faster than model retraining from  $D_r$ .

## 3 Our method

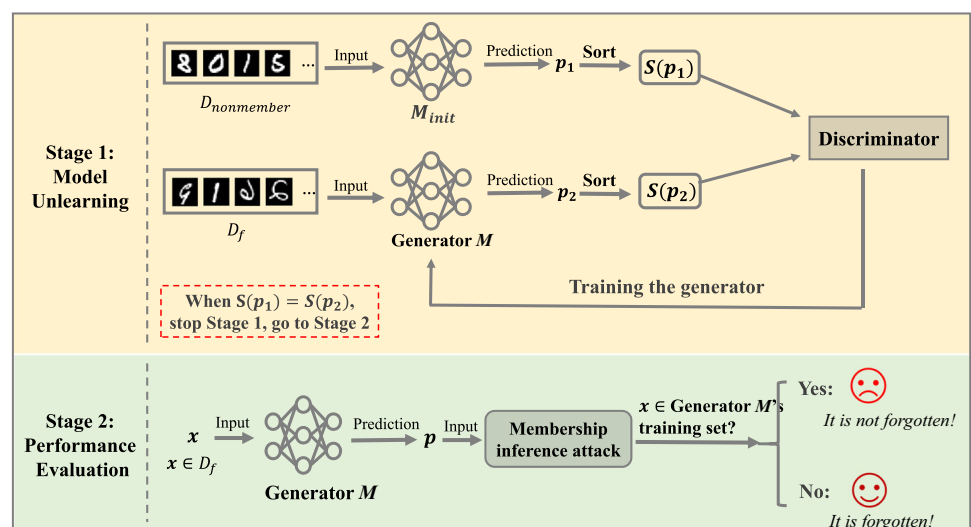
### 3.1 Overview

Inspired by the Generative Adversarial Networks (GAN), we present a fast machine unlearning method. The system architecture is shown in Fig. 1. It includes two stages, namely the model unlearning stage and the performance evaluation stage.

**Stage 1: Model Unlearning.** As illustrated in Eq. (5), the output distributions of  $M_{init}$  with  $D_{nonmember}$  are very similar to the output distribution of  $M$  with  $D_f$ . Thus, the given  $D_{nonmember}$  and  $M_{init}$  can be regarded as an optimization constraint, which is equivalent to the real input samples in GAN. Additionally,  $D_f$  and  $M$  can be regarded as the optimization function, which is equivalent to the generator in GAN. We can start with a random model  $M$  that has the same network architecture as  $M_{init}$ , and compare their output distributions in a discriminator. Then, we can iteratively train the generator to improve  $M$  until the discriminator can not distinguish the difference in their output distributions from  $M(D_f)$  and  $M_{init}(D_{nonmember})$ .

In Stage 1, the third-party dataset  $D_{nonmember}$  is fed into the  $M_{init}$  to determine a posterior  $P_1$ , while  $D_f$  is fed into the generator  $M$  to obtain a posterior  $P_2$ , where  $P_1$  and  $P_2$  are vectors in classification tasks while they are scalars in regression tasks. In this paper, we focus on classification scenarios. Then we sort  $P_1$  and  $P_2$  in a descending order (or an ascending order). We provide a clear explanation of the sort function in the following section. Here, compared with the standard GAN architecture [18],  $S(P_1)$  is equivalent to the *real data* and  $S(P_2)$  is equivalent to the *fake data*. We use  $S(P_1)$  and  $S(P_2)$  to train  $M$  and the *discriminator* alternatively with a standard GAN. When  $S_P(1)$  and  $S_P(2)$  are close enough, it means that  $D_f$  performs closely to  $D_{nonmember}$ . That is to

**Fig. 1** System framework of our method



say,  $D_f$  is successfully removed from the model  $M$ , so we can stop the model unlearning stage.

**Stage 2: Performance Evaluation.** In Stage 2, we leverage a membership inference attack (MIA) to evaluate whether  $D_f$  is truly forgotten in  $M$ . A membership inference attack is used to determine whether a data sample belongs to the training set of a trained machine learning model or not [34]. In MIA, they first trained different shadow models for each class. These shadow models are utilized to train an attack model (e.g., a binary classifier). According to the attack model's output, it can distinguish whether a data sample is used in the training dataset or not. [19] and [3] further investigate how to use one shadow model for all classes.

In Stage 2, we utilize MIA to evaluate the effectiveness of our model unlearning. We first initialize a shadow model with the same architecture as  $M_{init}$ , and then train it [3]. We also train an attack model from the output of this shadow model with both training data and non-training data. After that, we feed  $D_f$  into  $M$  which is determined in Stage 1. By calculating the output of the attack model, we can judge whether  $D_f$  is removed. In other words, if the attack model still infers  $D_f$  as a training set of  $M$ , our approach failed to delete  $D_f$  from  $M$  in Stage 1. Otherwise, our method removes  $D_f$  from  $M$  successfully in Stage 1.

### 3.2 From GAN to WGAN GP

We use a variant GAN, called WGAN with gradient penalty(WGAN GP), in our Stage 1. The basic reason is that the original GAN suffers from the gradient vanishing and mode collapse problems [18, 28]. To solve these problems, some researchers proposed the WGAN with gradient penalty(WGAN GP) method, which replaces the loss function of the discriminator by an Earth Mover distance, and add an extra term of gradient penalty into the loss function of the discriminator to accelerate the training process [19, 29]. Thus, we adapt WGAN GP as our base algorithm.

Based on WGAN GP, we redesign the loss functions for our machine unlearning stage. Let  $L(C)$  and  $L(G)$  be the loss functions of the discriminator and the generator  $M$ , respectively, we have

$$L_C = -\mathbb{E}_{x \sim \mathbb{P}_{nonmember}} [C(S(M_{init}(x)))] + \mathbb{E}_{z \sim \mathbb{P}_f} [C(S(G(z)))] + \lambda \mathbb{E}_{\hat{x} \sim \hat{\mathbb{P}}} [||\nabla_{\hat{x}}||_p - 1]^2, \quad (6)$$

$$L_G = \alpha(-\mathbb{E}_{z \sim \mathbb{P}_f} [C(S(G(z)))] + (1 - \alpha) \int_{(x,y) \in C_f} L(G(x), y), \quad (7)$$

where  $S(\cdot)$  is the sort function as discussed in the next subsection,  $\hat{x} = \epsilon S(P_1) + (1 - \epsilon)S(P_2)$ , and  $\epsilon \sim Uniform(0, 1)$ . It means that  $\hat{x}$  is obtained by a uniform sampling of two

probability distributions  $S(P_1)$  and  $S(P_2)$ . In Eq. (7), we add an additional term of  $L_{(x,y) \in D_f}(G(x), y)$  into the loss function of  $G$  to maintain the performance of  $G$ , where  $L$  is the loss function of the training model  $M_{init}$ . According to our evaluation, the performance decreases significantly without this compensation. Thus, the hyperparameter  $\alpha$  is used to balance the data removal and the performance.

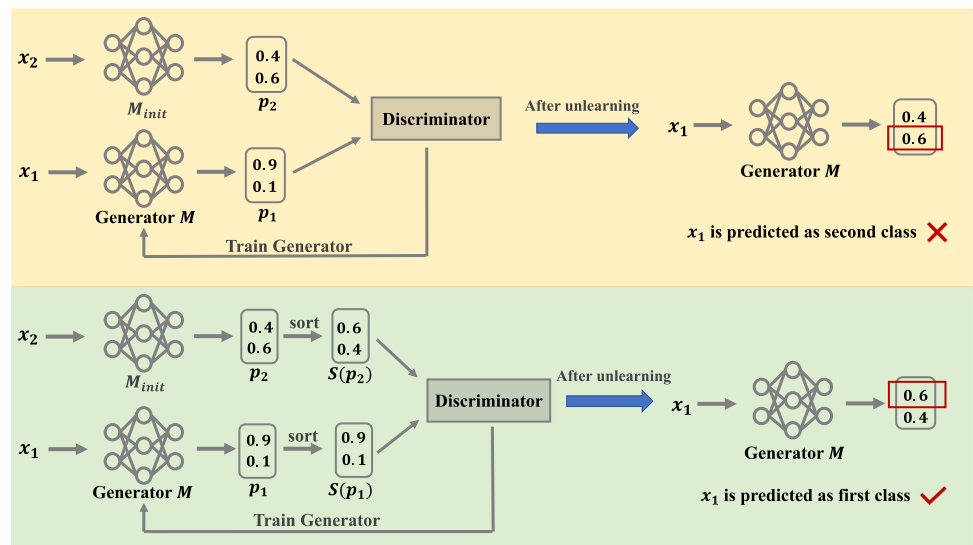
### 3.3 The sort function

The sort function allows us to be less strict in preparing the third-party data. Taking binary classification as an example, let  $M_{init}$  be the target model,  $x_1$  be the forgotten data with label “[1,0]”(one-hot encoding),  $x_2$  be the third-party data with label “[0, 1]”. Since  $x_1$  is part of  $M_{init}$ 's training data,  $M_{init}$  would give a high confidence prediction  $p_1$ . In contrast,  $M_{init}$  has never seen  $x_2$ , thus  $M_{init}$  would give a low confidence prediction  $p_2$ . Let's assume  $p_1 = [0.9, 0.1]$ ,  $p_2 = [0.4, 0.6]$ . We expect to reduce the confidence of  $p_1$ , like reducing  $p_1$  from  $[0.9, 0.1]$  to  $[0.4, 0.6]$ . The decreasing confidence implies that the model gradually becomes unfamiliar with the forgotten data. However, the vector  $[0.4, 0.6]$  means the forgotten data  $x_1$  is predicted as class “2” (the index where the maximum probability value is located) while its ground truth label is class “1”, which has a negative impact on the performance of the unlearned model. Figure 2 shows a demonstration of this process. To avoid this, there are two solutions. First, we can elaborately prepare the third-party data such that the samples in the third-party data are in the same category as the samples in the forgotten data. Second, we can sort the model's prediction to neglect the class information and only focus on prediction confidence. Preparing third-party data with the same categories as the forgotten data is time-consuming and impractical, especially when the forgotten data contains huge numbers of categories. Therefore, we use a sort function here.

### 3.4 Membership inference attack

We use membership inference attack (MIA) to evaluate whether the dataset  $D_f$  is successfully forgotten or not. In our MIA scenario, we introduce two new disjoint third-party datasets  $D_{in}$  and  $D_{out}$ , and train the shadow model  $M_{shadow}$  with  $D_{in}$ . Let the output of the shadow model for  $D_{in}$  and  $D_{out}$  be  $P_{in}$  and  $P_{out}$ , respectively. We can label  $P_{in}$  and  $P_{out}$  with training category and a non-training category to form a new dataset, which can be used to train the attack model  $Q$  (usually a binary classifier). A well trained attack model  $Q$  can determine whether a sample was trained by the target model  $M_{init}$  or not. Let  $P_f$  be the model output of  $M$  with the input dataset  $D_f$ , it is then judged by the attacker  $Q$ . Finally,

Fig. 2 Without/with sort function



if  $Q$  infers  $D_f$  is not trained by  $M$ , then  $D_f$  is successfully forgotten; otherwise  $D_f$  is not forgotten.

## 4 Evaluation

### 4.1 Evaluation metrics

We evaluate our method with three metrics including *effectiveness*, *performance*, and *time*.

**Effectiveness:** It indicates that whether the data  $D_f$  were completely removed from the model  $M$ , which is evaluated by the member inference attack results in our experiment. Specifically, we focus more on the False Negative Rate (FNR). FNR is defined by  $FNR = \frac{FN}{TP+FN}$ , where TP means that MIA infers  $D_f$  as training data of  $M$ , while FN means that MIA infers  $D_f$  as non-training data of  $M$ . After successful machine unlearning (or data removal),  $D_f$  will be marked as non-training data for  $M$ . Therefore, FN will increase and TP will decrease, which generates a large FNR.

**Performance:** It demonstrates how the model  $M$  performs on test datasets after machine unlearning, which is evaluated by the classification accuracy on different test datasets.

Table 1 Model Architectures for different datasets

Dataset	Model
FASHION	LeNet
SVHN	2×Conv+2×FC
CIFAR10	ResNet18
CIFAR100	ResNet18
Purchase[10,20,50,100]	3×FC

**Time:** It shows that how long it takes to complete the machine unlearning, which is evaluated by the time cost.

### 4.2 Datasets

We evaluate our method on different datasets such as Fashion, SVHN, CIFAR10, CIFAR100 and Purchase.

**Fashion** The Fashion dataset [15] contains 70,000 different samples of products from 10 categories, such as T-shirts, pants, and skirts. The training dataset includes 60,000 samples, and the testing dataset includes 10,000 samples. Each sample is a grayscale image with a data size of  $28 \times 28$ .

**SVHN** The Street View House Numbers (SVHN) Dataset [40] comes from house numbers in Google Street View images, which contains 73,257 samples of training data and 26,032 of testing data. Each sample is an image with  $32 \times 32$  RGB digit number. SVHN has two different formats. One of them is the image with character level bounding boxes, and the other is the image centered around a single character. We use the second one in our experiment.

Table 2 Overfit level for different datasets

Datasets	Training accuracy	Testing accuracy
FASHION	0.98	0.846
SVHN	0.993	0.829
CIFAR10	0.997	0.536
CIFAR100	0.98	0.319
Purchase10	0.99	0.811
Purchase20	1	0.751
Purchase50	1	0.693
Purchase100	1	0.617



**CIFAR** The CIFAR dataset is obtained from the real-world, such as pictures of airplanes, birds, cats, etc. It consists of 60,000 RGB images with a size of  $32 \times 32$ , wherein 5000 images are used for training data, and 10,000 are used for testing data. There are two versions of CIFAR, called CIFAR10 and CIFAR100. CIFAR10 has 10 classes with 6000 samples per class, while CIFAR100 contains 100 classes with 600 images per class.

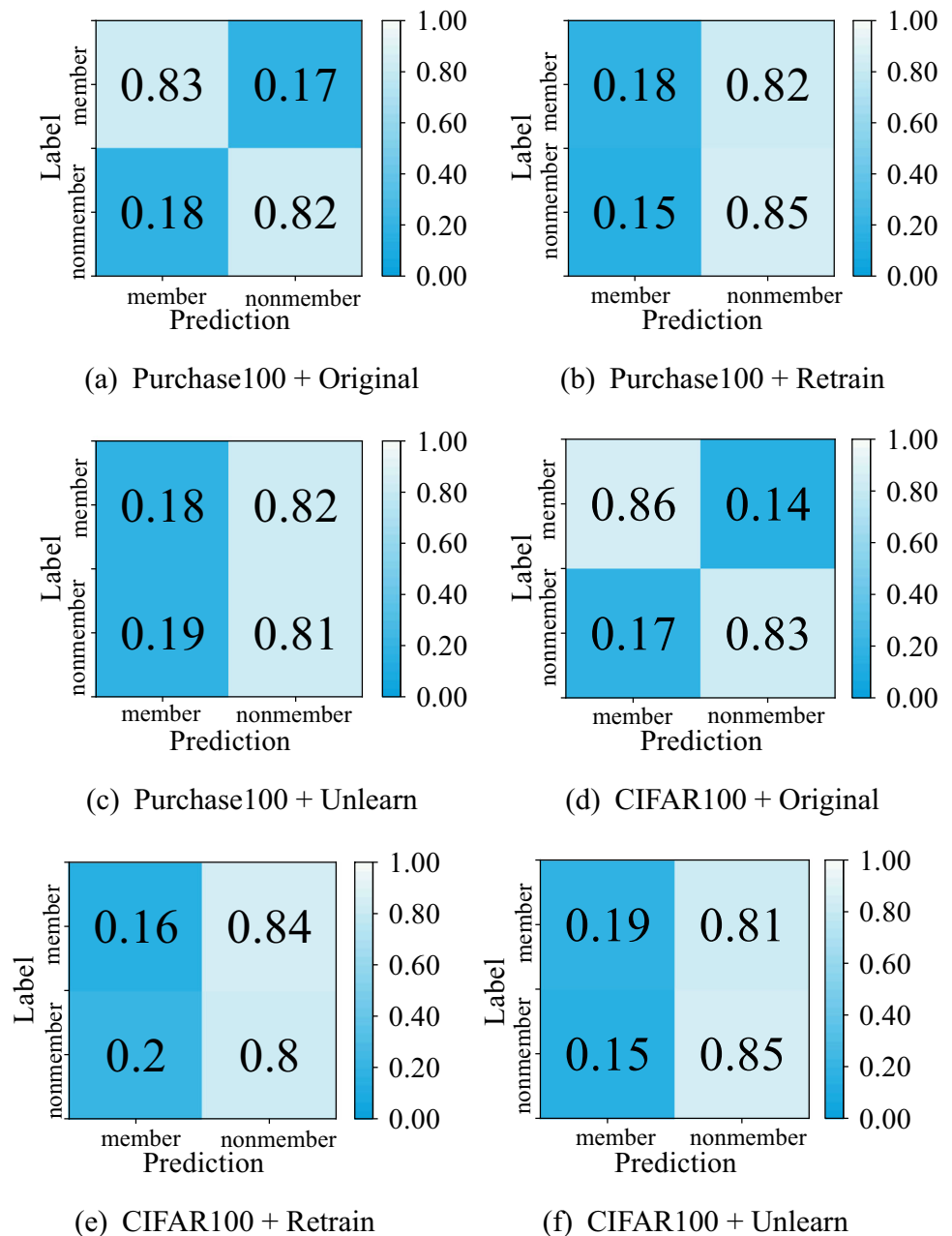
**Purchase** The Purchase dataset is an unlabeled dataset from Kaggle’s ‘acquire valued shoppers’ competition. The purpose of this competition is to design a coupon recommendation strategy. Each data sample contains transaction

records of a user over a year, such as product name, quality, and date. Similar to [34], we use a simplified Purchase database with 19,324 samples, where each sample contains 600 features. We further leverage K-Means algorithm to cluster them into 10, 20, 50, and 100 categories to form four datasets. We call them Purchase10, Purchase20, Purchase50, and Purchase100, respectively.

### 4.3 Model architecture

Considering the complexity of different datasets, we use different model architectures for them, as shown in Table 1.

**Fig. 3** Confusion matrix of MIA on CIFAR100 and Purchase100



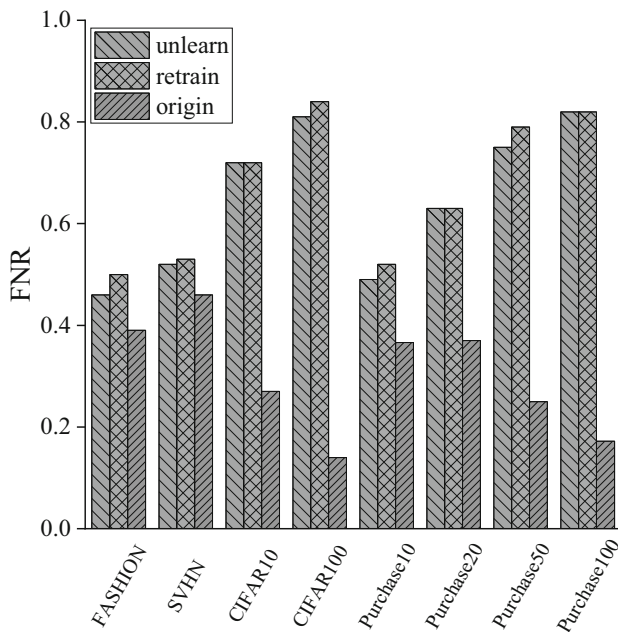


Fig. 4 FNR of membership inference attack

In the membership inference attack, it is important to overfit the training data to better distinguish them from untrained data [34]. We listed the overfit levels (i.e., training accuracy and test accuracy) for different datasets in Table 2.

#### 4.4 Experimental results

In this section, we introduce the experimental results in detail. The third-party data  $D_{nonmember}$  consist of 500 randomly sampling data from the same dataset with  $D$ , where  $D_{nonmember} \cap D = \phi$ . We compare with our method ‘unlearn’ with a baseline methods ‘retrain’, which retrain a machine learning method directly with the remaining data  $D_f$ . We also compare ‘unlearn’ with the original model  $M_{init}$ , namely ‘Original’ for short.

##### 4.4.1 Effectiveness of machine unlearning

We have implemented MIA on both the original model  $M_{init}$  and the unlearned model  $M$ . The MIA model aims to determine whether  $D_f$  is trained by  $M_{init}$  (or  $M$ ) or not. In ideal case, the MIA model will judge  $D_f$  and  $D_{nonmember}$  as training data and non-training data of  $M_{init}$  before machine unlearning, while the MIA model will judge both  $D_f$  and  $D_{nonmember}$  as non-training data of  $M$  after machine unlearning.

Figure 3(a, c) shows the confusion matrix of MIA on the Purchase100 dataset. For the original model  $M_{init}$ , its TP and TN of MIA are 0.83 and 0.82, respectively, which indicates that MIA can distinguish most of  $D_f$  and  $D_{nonmember}$ . For

the retain model, the TP of MIA decreases to 0.18 while FN increases to 0.82. It means that most of  $D_f$  are determined as non-training data of  $M$  by MIA, which demonstrates that the data removal in retrain is very effective. For our unlearning method, the TP, FN and TN are 0.18, 0.82, and 0.81, respectively, which are very close to those of the retrain method. It indicates that our unlearning method performs quite well. Figure 3(d, f) shows the confusion matrix of MIA on CIFAR100 dataset, producing a similar solution as the Purchase dataset.

Figure 3(a, c) presents the FNRs of MIA. The FNR of the original model before data removal is relatively low for each dataset, which means that the MIA model classifies correctly as non-training data for most of  $D_f$ . The FNR of MIA on the unlearned model is larger than the original model for each dataset, which indicates that our unlearning method could indeed remove the information of  $D_f$  from a trained model. Additionally, for CIFAR10, Purchase20, and Purchase100, our unlearning method is almost the same as the retrain method, which confirms our effectiveness. The results of FNR are shown in Fig. 4.

##### 4.4.2 Accuracy on test datasets

As shown in Fig. 5, the accuracy of the original model is slightly higher than both the retrain model and the unlearned model on all datasets. This is reasonable because data removal will cause a slight model performance decrease. Meanwhile, the accuracy of our unlearned method is very close to that of the retrain method for each dataset.

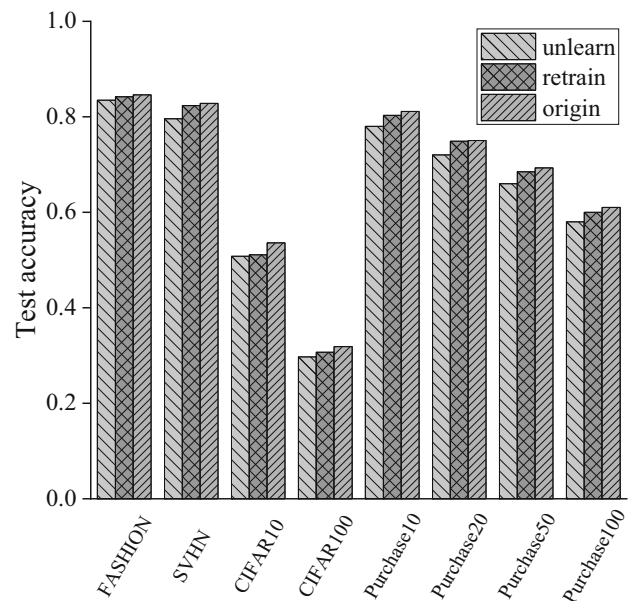


Fig. 5 Model’s accuracy on test data

**Table 3** Time cost of our approach on CIFAR10

Size of Training Set	Size of Forget Set	Model	Size of Model Parameters	Time of Retrain (second)	Time of Unlearning (second)	Speed up
5000	500	ResNet18	1.2 million	111.5	34	3.3×
10000	500	ResNet50	2.6 million	570	71	8×
25000	500	ResNet101	4.5 million	2507	111	22.6×

#### 4.4.3 Time cost for data removal

Time consumption is an important metric for the data removal. Intuitively, the time cost is related to the amount of the training data and parameters of the model [27]. We have evaluated the time cost on different scenarios. As shown in Table 3, we can infer that our unlearning method is much better than the retrain method in three scenarios, producing 3.3 × to 22.6 × improvement. When the scenario becomes more complex (e.g., a larger data size of training set, a more complex model, a larger size of model parameters), our model generates a better improvement over the retrain method.

#### 4.4.4 Impact of alpha

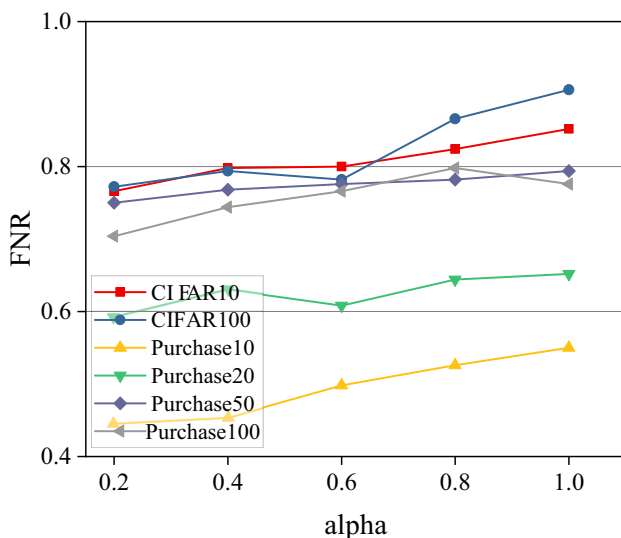
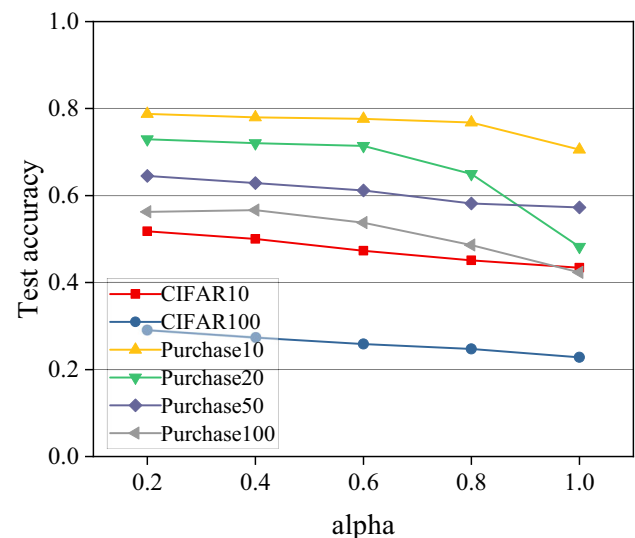
We evaluate the impact of the hyperparameter  $\alpha$  in the range of [0.2, 0.4, 0.6, 0.8, 1.0], which is presented in Figs. 6 and 7. When  $\alpha$  increases, the FNR of MIA also increases while the test accuracy decreases, which indicates that data removal is more complete with a larger alpha, while the performance is less competitive after data removal. For different datasets and different models, the value of  $\alpha$  should not be constant. For example, by increasing alpha from 0.6 to 0.8 on Purchase20,

its test accuracy of our unlearned model drops from 71% to 64% while its FNR of MIA increases from 61% to 64%. Thus,  $\alpha$  should be a trade-off parameter to balance the performance and data removal. A larger  $\alpha$  tends to a better data removal, while a smaller  $\alpha$  tends to a better classification performance.

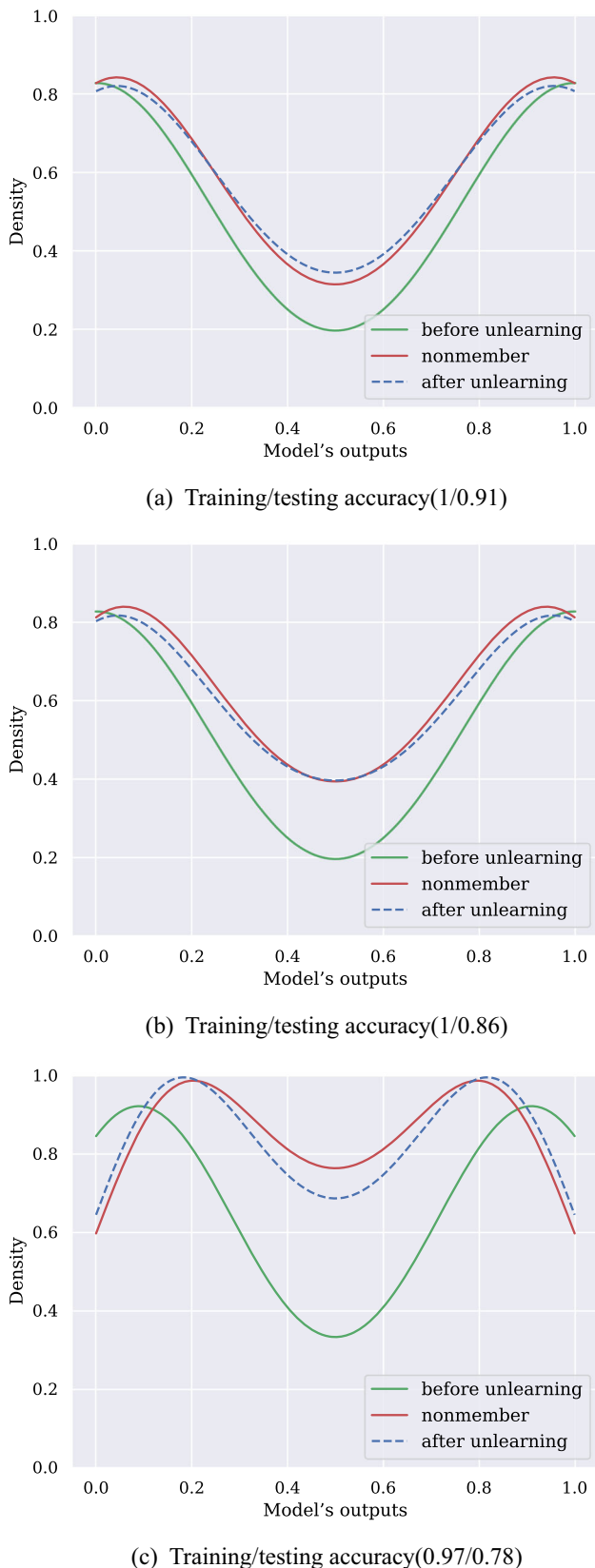
#### 4.4.5 Impact of model overfitting

We further study the impact of model overfitting. We divided the Purchase dataset into two categories with the K-Means algorithm, and then trained the model with different levels of overfitting. In this training model, we have three fully connected layers, where the activation function is *softmax* and the output dimension of the model is two. Before and after data removal, we feed the data  $D_f$  and the third-party data  $D_{nonmember}$  into the original model respectively, and then draw the curves of the Kernel Density Estimation map (KDE) of these corresponding outputs.

The experimental results are shown in Fig. 8, where the green and red solid lines are the KDE curves of  $D_f$  and  $D_{nonmember}$  before the model unlearning. These two lines do not fully overlap, which indicates that the initial model could distinguish  $D_f$  and  $D_{nonmember}$ . The blue dashed line

**Fig. 6** FNR v.s alpha**Fig. 7** Model performance v.s. alpha





**Fig. 8** Impact of overfitting

is the KDE curve after our model unlearning method, which is almost overlapped with the red solid line. It means that there is little difference between  $D_f$  and  $D_{nonmember}$  for our unlearning model method. Thus, our method can successfully forget  $D_f$  from the initial model. From (a) to (c), we gradually strengthen the level of overfitting. The shape and location of these red solid lines and blue dashed lines are similar in the three situations, which shows that our method can also forget data effectively for different overfitting levels.

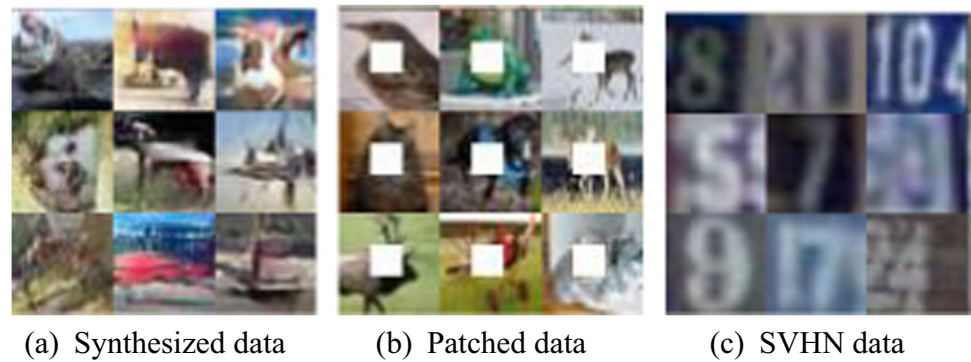
#### 4.4.6 Impact of third-party data

In practical scenario, the third-party data with the same distribution as the forgotten data may not be easily available. In this part, we test our method with different distributions of third-party data. In this experiment, the model is pre-trained on the CIFAR10 dataset. We prepare three types of data as third-party data, which are the synthesized data (synthesized by GAN technique [18]), the patched data (a white pattern covered on CIFAR10 sample), and the SVHN data, respectively. From synthesized data to SVHN data, their distribution is increasingly uncorrelated with that of CIFAR10. Figure 9 shows the visualized demonstration of these data. The experiment results are shown in Fig. 10. We can see that using SVHN data as third-party data achieves larger FNR than using synthesized and patched data but obtains the worse test accuracy. While using synthesized data as the third-party data achieves a relatively small FNR but maintains the test accuracy as possible, compared with patched and SVHN data. Please note that, a larger FNR does not always mean better forgotten performance, forgotten performance should be assessed in the context of good model performance. For example, an unlearning method that aims to randomize the model's parameters may obtain a good forgotten performance but is meaningless since it loses the model performance. Therefore, if the unlearning method could achieve comparable FNR as retrain (baseline method, experiments results of CIFAR10 please see Fig. 4), we should pay more attention to model performance. For this experiment, the synthesized data is a better choice.

## 5 Related works

In smart cities, there are many solutions to address the emerging privacy and security concerns. For example, [24, 38, 43] discussed the serious privacy leakage problems in deep neural networks. Hou et al. [17] presented a similarity-based integrity protection method for deep learning based data analysis systems. In [25], the authors investigated an adversarial example generation algorithm, serving an alternative way

**Fig. 9** Visualized demonstration of different third-party data

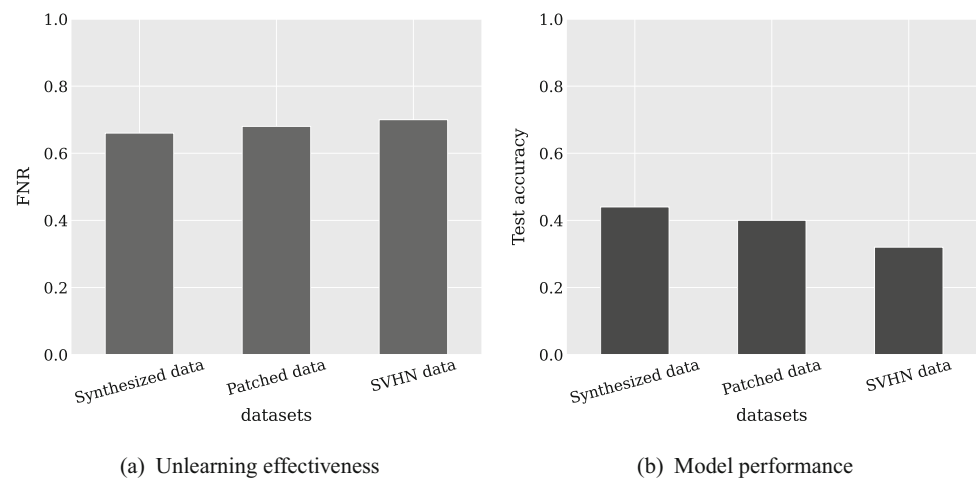


for raw data sharing to avoid potential data leakage. Reza et al. [34] first studied the membership inference attack against ML models. They proposed a shadow training-based attack which trains multiple shadow models to imitate the behavior of the target model, then they constructed a attack model (i.e., a binary classifier) to attack the target model based on these shadow models. Ahmed et al. [3] further investigated the shadow training-based attack in black-box settings. They showed that the shadow training-based attack is still effective even without knowing the architecture and the training data distribution of the target model. Except for the shadow training-based attack, there is another type of attack called the metric-based attack. In metric-based attack, the adversary first calculated the prediction vectors of data data records, then compared the calculated prediction vectors with the preset threshold to discriminate whether the data records are members or not. Depending on the metric, it can be divided into Prediction Correctness based [1, 35] and Prediction Entropy based [26]. In addition, the membership inference attack has been extended to different machine learning domains, such as Federated Learning [20, 21, 31], Unsupervised Learning [16], and Graph Learning [4, 13].

As a novel data privacy preserving approach in deep neural networks, machine unlearning was first introduced by

[39]. They proposed a statistical query-based unlearning, which could efficiently erase data information from a trained model, but it failed to apply to neural networks. Tony et al. [36] proposed two data removal algorithms for the K-Means algorithm. Lucas et al. [27] proposed a similar method to ensemble learning, which divided the original dataset into multiple blocks. Chuan et al. [12] proposed a certified data removal method by differential privacy. They applied the Newton’s method to delete data in a linear model, and added random noise to the loss function of the model to make it indistinguishable between the unlearned model and the retrained model. Aditya et al. [2] studied adding noise to the model parameters to delete a specific class (or a subset of a specific class) in the classification task, which still suffers from its high computational complexity. Gaoyang et al. [14] studied data removal in a federated learning way. During the training phase, the central parameter server saved updated parameters for each round. Thus, in the retrain phase, they loaded intermediate parameters to speed up the training speed. [41] studied the data deletion in linear models and logistic regression models, and proposed an approximate deletion method whose computational cost is linear with the data dimension. Recently, [22] studied how to delete data in random forests. In general, existing solutions cannot work in

**Fig. 10** The difference of using different third-party data



a generic neural networks, or need a strong hypothesis on the data distribution. Our method turns to the adversarial learning of pre-trained model, serving as a novel general approach in this area.

## 6 Conclusions

In smart cities, considerable raw sensor data from crowded clients are collected for centralized or distributed model training in vertical applications. However, many countries and organizations have significant concerns about the data security in smart cities. They have a recent focus on the *The Right to be Forgotten* to remove inappropriate data usage from pre-trained models. In this paper, we borrow the idea of Generative Adversarial Network (GAN), and present a fast machine unlearning method which learns a new method in an adversarial way. Experimental results show that our method achieves high model performances with quite fast model unlearning speed.

**Acknowledgements** This work is supported by National Natural Science Foundation of China (No. 61802383), Research Project of Pazhou Lab for Excellent Young Scholars (No. PZL2021KF0024, No. PZL2021KF0002), Guangzhou Basic and Applied Basic Research Foundation (No. 202201010330, No. 202201020162, No. 202201020221), Guangdong Philosophy and Social Science Planning Project (No. GD19YYJ02), Guangdong Regional Joint Fund Project (No. 2022A1515110157), Research on the Supporting Technologies of the Metaverse in Cultural Media (No. PT252022039), Jiangsu Key Laboratory of Media Design and Software Technology (No. 21ST0202), Shaanxi Key Laboratory of Blockchain and Secure Computing, Guangzhou University Research Project (grant no. PT252022039), and Innovation Research for the Postgraduates of Guangzhou University (No. 2021GDJC-M33).

## Declarations

**Conflict of interest** The authors declare no competing interests.

## References

1. A CCC, Florian T, Nicholas C, et al (2021) Label-only membership inference attacks. In: International conference on machine learning, PMLR, pp 1964–1974
2. Aditya G, Alessandro A, Stefano S (2020) Eternal sunshine of the spotless net: Selective forgetting in deep networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9304–9312
3. Ahmed S, Yang Z, Mathias H, et al (2019) MI-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In: Network and Distributed Systems Security Symposium 2019, Internet Society
4. Bang W, Xiangwen Y, Shirui P, et al (2021) Adapting membership inference attacks to gnn for graph classification: Approaches and implications. In: 2021 IEEE international conference on data mining (ICDM), IEEE, pp 1421–1426
5. Blanc G, Liu Y, Lu R et al (2022) Interactions between artificial intelligence and cybersecurity to protect future networks. *Annals of Telecommunications* 77:727–729
6. (CCPA) CCPA (2018) <https://oag.ca.gov/privacy/ccpa>
7. Charmet F, Tanuwidjaja HC, Ayoubi S et al (2022) Explainable artificial intelligence for cybersecurity: a literature survey. *Annals of Telecommunications* 77:789–812
8. Chen K, Tan G (2019) Bikegps: Localizing shared bikes in street canyons with low-level GPS cooperation. *ACM Trans Sens Networks* 15(4):45:1–45:28
9. Chen K, Tan G (2021) Satprobe: Low-energy and fast indoor/outdoor detection via satellite existence sensing. *IEEE Trans Mob Comput* 20(3):1198–1211
10. Chen K, Tan G, Lu M et al (2016) CRSM: a practical crowdsourcing-based road surface monitoring system. *Wirel Networks* 22(3):765–779
11. Chen K, Tan G, Cao J et al (2020) Modeling and improving the energy performance of gps receivers for location services. *IEEE Sensors Journal* 20(8):4512–4523
12. Chuan G, Tom G, Awni H, et al (2020) Certified data removal from machine learning models. In: International Conference on Machine Learning, PMLR, pp 3832–3842
13. Oi E, Wolfgang N, Megha K (2021) Membership inference attack on graph neural networks. 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), IEEE, pp 11–20
14. Gaoyang L, Yang Y, Xiaoqiang M, et al (2020) Federated unlearning. arXiv preprint [arXiv:2012.13891](https://arxiv.org/abs/2012.13891)
15. Han X, Kashif R, Roland V (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint [arXiv:1708.07747](https://arxiv.org/abs/1708.07747)
16. Hongbin L, Jinyuan J, Wenjie Q, et al (2021) Encodermi: Membership inference against pre-trained encoders in contrastive learning. In: Proceedings of the 2021 ACM SIGSAC conference on computer and communications security, pp 2081–2095
17. Hou R, Ai S, Chen Q et al (2022) Similarity-based integrity protection for deep learning systems. *Inf Sci* 601:255–267
18. Ian G, Jean PA, Mehdi M, et al (2014) Generative adversarial nets. *Advances in neural information processing systems* 27
19. Ishaan G, Faruk A, Martin A, et al (2017) Improved training of wasserstein gans. In: Proceedings of the 31st international conference on neural information processing systems, pp 5769–5779
20. Jiale C, Jiale Z, Yanchao Z, et al (2020) Beyond model-level membership privacy leakage: an adversarial approach in federated learning. In: 2020 29th international conference on computer communications and networks (ICCCN), IEEE, pp 1–9
21. Jingwen Z, Jiale Z, Junjun C, et al (2020) Gan enhanced membership inference: A passive local attack in federated learning. In: ICC 2020-2020 IEEE international conference on communications (ICC), IEEE, pp 1–6
22. Jonathan B, Daniel L (2021) Machine unlearning for random forests. In: International conference on machine learning, PMLR, pp 1092–1104
23. Kim T, Jerath K (2022) Congestion-aware cooperative adaptive cruise control for mitigation of self-organized traffic jams. *IEEE Trans Intell Transp Syst* 23(7):6621–6632
24. Li Y, Yan H, Huang T, et al (2022) Model architecture level privacy leakage in neural networks. *Science China Information Sciences*
25. Liu J, Zhang Q, Mo K et al (2022) An efficient adversarial example generation algorithm based on an accelerated gradient iterative fast gradient. *Comput Stand Interfaces* 82(103):612
26. Liwei S, Prateek M (2021) Systematic evaluation of privacy risks of machine learning models. In: 30th USENIX Security symposium (USENIX Security 21), pp 2615–2632

27. Lucas B, Varun C, A CCC, et al (2021) Machine unlearning. In: 2021 IEEE Symposium on Security and Privacy (SP), IEEE, pp 141–159
28. Martin A, Leon B (2017) Towards principled methods for training generative adversarial networks. arXiv preprint [arXiv:1701.04862](https://arxiv.org/abs/1701.04862)
29. Martin A, Soumith C, Leon B (2017) Wasserstein generative adversarial networks. In: International conference on machine learning, PMLR, pp 214–223
30. Matt F, Somesh J, Thomas R (2015) Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp 1322–1333
31. Milad N, Reza S, Amir H (2019) Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: 2019 IEEE symposium on security and privacy (SP), IEEE, pp 739–753
32. Nicholas C, Chang L, Ulfar E, et al (2019) The secret sharer: Evaluating and testing unintended memorization in neural networks. In: 28th {USENIX} Security Symposium ({USENIX} Security 19), pp 267–284
33. Paul V, dem Bussche Axel V (2017) The eu general data protection regulation (gdpr). A Practical Guide, 1st Ed, Cham: Springer International Publishing 10:3152,676
34. Reza S, Marco S, Congzheng S, et al (2017) Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, pp 3–18
35. Samuel Y, Irene G, Matt F, et al (2018) Privacy risk in machine learning: Analyzing the connection to overfitting. In: 2018 IEEE 31st computer security foundations symposium (CSF), IEEE, pp 268–282
36. TONY G, MELODY G, GREG V, et al (2019) Making ai forget you: Data deletion in machine learning. *Advances in Neural Information Processing Systems*
37. Vitaly F (2020) Does learning require memorization? a short tale about a long tail. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, pp 954–959
38. Wang Y, Chen K, an Tan Y, et al (2022) Stealthy and flexible trojan in deep learning framework. *IEEE Transactions on Dependable and Secure Computing*
39. Yinzhi C, Junfeng Y (2015) Towards making systems forget with machine unlearning. In: 2015 IEEE Symposium on Security and Privacy, IEEE, pp 463–480
40. Yuval N, Tao W, Adam C, et al (2011) Reading digits in natural images with unsupervised feature learning
41. Zachary I, Anne SM, Kamalika C, et al (2021) Approximate data deletion from machine learning models. In: International Conference on Artificial Intelligence and Statistics, PMLR, pp 2008–2016
42. Zhao B, Mopuri KR, Bilen H (2020) idlg: Improved deep leakage from gradients. *CoRR abs/2001.02610*
43. Zhu E, Zhang J, Yan J et al (2022) N-gram malgan: Evading machine learning detection via feature n-gram. *Digital Communications and Networks* 8(4):485–491
44. Zhu L, Liu Z, Han S (2019) Deep leakage from gradients. In: Wallach HM, Larochelle H, Beygelzimer A, et al. (eds) *Advances in neural information processing systems 32: annual conference on neural information processing systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada*, pp 14747–14756

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.