SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2024, Vol. 67, Iss. 3, 132104:1–132104:17
https://doi.org/10.1007/s11432-022-3671-9

Mitigate noisy data for smart IoT via GAN based machine unlearning

Zhuo MA[1], Yilong YANG[1*], Yang LIU[1*], Xinjing LIU[1] & Jianfeng MA[1,2]

[1]*School of Cyber Engineering, Xidian University, Xi'an* 710071*, China;*
[2]*State Key Laboratory of Integrated Services Networks* (*ISN*)*, Xi'an* 710071*, China*

Received 27 April 2022/Revised 31 October 2022/Accepted 29 December 2022/Published online 2 February 2024

Abstract With the development of IoT applications, machine learning dramatically improves the utility of variable IoT systems such as autonomous driving. Although the pretrain-finetune framework can cope well with data heterogeneity in complex IoT scenarios, the data collected by sensors often contain unexpected noisy data, e.g., out-of-distribution (OOD) data, which leads to the reduced performance of fine-tuned models. To resolve the problem, this paper proposes MuGAN, a method that can mitigate the side-effect of OOD data via the generative adversarial network (GAN)-based machine unlearning. MuGAN follows a straightforward but effective idea to mitigate the performance loss caused by OOD data, i.e., "flashbacking" the model to the condition where OOD data are excluded from model training. To achieve the goal, we design an adversarial game, where a discriminator is trained to identify whether a sample belongs to the training set by observing the confidence score. Meanwhile, a generator (i.e., the target model) is updated to fool the discriminator into believing that the OOD data are not included in the training set but others do. The experimental results show that benefiting from the high unlearning rate (more than 90%) and retention rate (99%), MuGAN succeeds in lowering the model performance degradation caused by OOD data from 5.88% to 0.8%.

Keywords machine unlearning, generative adversarial network, out of distribution data, Internet of Thing, neural network

1 Introduction

Nowadays, the application of machine learning in Internet of Things (IoT) systems provides conveniences to all walks of life, such as smart homes, the Internet of vehicles, and smart surveillance [1, 2]. In these systems, training a machine learning model in a single point (e.g., an IoT device) is usually impractical because the data collected by a single point is not enough to obtain a well-performance model [3–5]. Therefore, the pretrain-finetune framework is proposed and ubiquitously applied in state-of-the-art machine learning systems [6]. In the framework, a generic model is first pre-trained in a central control point that owns massive data collected in the whole system. Then, the pre-trained model is distributed and finetuned by each other device to further improve the performance and adaptability of the model to satisfy the personal requirement of different devices.

The vast majority of existing machine learning methods work under the closed-world assumption, where the data is drawn from the same distribution, called in-distribution (ID) data. However, in the open-world system, the data collected by the IoT sensors are often mixed with noisy data, called out-of-distribution (OOD) data [7]. As mentioned in [8,9], the introduction of these OOD data while finetuning the pre-trained model may lead to a decrease in model performance and even catastrophic consequences due to the overconfidence problem. Intuitively, the solution to the above problem is to mitigate the negative impact of these OOD data while finetuning the model, restoring the model to the state where no OOD data are encountered.

At a first glance, complete retraining with the training data excluding OOD samples is a straightforward way to remove the impact of OOD data. However, the intensive computational overhead for retraining is usually too high to be acceptable for smart IoT applications. Therefore, Bourtoule et al. [10] proposed

* Corresponding author (email: yangyilong@stu.xidian.edu.cn, bcds2018@foxmail.com)

sharded, isolated, sliced, and aggregated training (SISA) based on the retraining mechanism. SISA previously divides the training set into many blocks in sequence, incrementally trains them in turn, and records all the intermediate models. To forget certain data blocks of OOD data with side effects, SISA only has to continue incremental training from the previous intermediate model of this block instead of training from scratch. However, the optimization of SISA is based on the idea of space-for-time, introducing additional severe storage space because of intermediate models. Except for the methods derived from retraining, forcible unlearning (FU) proposed by Cao et al. [11] can also forget specific OOD training data by directly manipulating the gradients. Since FU needs to convert the machine learning algorithm into a summation form, the transformation function varies with model types. As pointed out by [12], for adaptive models such as deep neural networks, the forcible manipulation always leads to catastrophic forgetting, which can cause a dramatic performance loss of training data.

The above two approaches to removing noisy data run counter to the limited computing and storage sources and model performance requirements of smart IoT devices due to severe storage, computational overload, and catastrophic forgetting of the model [13]. Further, with the powerful extraction capability, neural networks (NN) are widely used in various complex IoT tasks, such as face recognition and autonomous driving [13]. Therefore, how to remove the negative impact of OOD data in smart IoT from neural networks effectively and efficiently needs to be solved urgently.

In this paper, we propose a machine unlearning approach based on a generative adversarial network (GAN), called MuGAN, which can restore the degraded model performance caused by noisy data. Specifically, machine unlearning in MuGAN aims to remove the negative influence of OOD data in the neural network by changing the state of OOD data from involved (overconfidence) to non-involved (underconfidence) in the training dataset. In other words, for OOD data, the target model behaves as if it outputs the confidence vector with characteristics the data never encountered. In terms of the reality of smart IoT, MuGAN accomplishes the above goal through an adversarial game with two tasks, maintaining performance and unlearning noisy data. In the game, a neural network serves as a discriminator into identify whether a sample is unlearned by observing the confidence vector. The other model, i.e., the target model, is constantly finetuned to "persuade" the discriminator to believe that the OOD data are unlearned. Intuitively, the above idea is consistent with GAN [14]. Compared with the previous work, MuGAN no longer needs to record additional data or intermediate models to accomplish the forgetting of OOD data with the lower computational load.

We summarize our work into three contributions.

• We propose a novel method, called MuGAN, to mitigate the negative effect on the model performance of noisy data via machine unlearning. MuGAN can serve as an opt-in component inside the existing pretrain-finetune framework without changing the original architecture.

• We design a novel adversarial game to implement machine unlearning by converting the confidence vector distribution of the target model on OOD data to non-member data.

• In order to remove the effects of the initial state before machine unlearning, we normalize two evaluation metrics, unlearning rate and retention rate, to evaluate the performance more effectively. The experimental results show that the accuracy of personalized data is improved by 5.08%. Moreover, MuGAN can achieve more than 90% unlearning rate on the basis of a retention rate of 99% with lower computational and storage resources while finetuning the pre-trained model.

The remaining part of this paper is arranged as follows: In Section 2, we briefly introduce some background knowledge and related work about machine unlearning. Section 3 formulates the definition of machine learning from the perspective of model functionality similarity. In Section 4, we illustrate the motivation of machine unlearning and then mention the overview and goals of MuGAN. In Section 5, we introduce MuGAN in detail and deploy it in the application. Section 6 performs comprehensive experiments and the advantages of MuGAN are illustrated by comparing with two classic schemes.

2 Background

This section briefly introduces the background knowledge of this paper.

Machine unlearning. As we all know, machine learning is a technique that can extract and learn general high-dimensional features of training data, building a model. During the process of machine learning, the training data all have a certain impact on the model. In contrast, machine unlearning is the

process of removing the impact of unlearned data. The model after machine unlearning should perform as if it has never met the unlearned data. For example, a model $G_{\mathrm{A+B}}$ is trained with data A and B, while somehow B is asked for the removal from the training dataset, and ideally, we can exploit machine unlearning to reach the status in which only A is used for training: $G_{\mathrm{A+B}} \to G_{\mathrm{A}}$.

We then discuss what is contained in so-called "somehow" through some cases in reality: (1) Privacy data. Legislation such as the "Right to be forgotten" in the General Data Protection Regulation (GDPR) expressly stipulates and requires that individuals have the right to delete or revoke their private data, e.g., browser history and purchase records, at any time. (2) Poisoned data. Considering a situation in which an adversary injects some poisoned data into the training dataset or uploads a local model trained with poisoned data in federated learning, the model owner wants to erase the impact of poisoned data from the model. (3) Negative data. In practical applications, negative data, such as OOD data in smart IoT, will inevitably be collected and trained into the model, affecting the performance and reliability of the model. When a model is detected to have negative data, the model owner wants to get rid of those data and their impact on the model to sustainably maintain the model performance.

State-of-the-art machine unlearning studies. Cao et al. [11] first proposed the concept of machine unlearning, presented an unlearning approach, FU, by converting the learning algorithm into a summation form, and presented two important goals of unlearning, i.e., completeness and timeliness. However, FU only performs better on non-adaptive models that can be converted into a summation form, such as Naive Bayes. Bourtoule et al. [10] designed SISA, a retraining-based unlearning method for constituent models by dividing the dataset into shards and slices. However, SISA is more suitable for scenarios where the unlearned data is concentrated in a shard or slice. Lately, the potential of machine unlearning in applications attracts the interest of the research field and is applied to many different machine learning models, e.g., recommendation systems [15, 16], nearest neighbors [15], linear classifier [17, 18], random forest [19], Bayesian model [20, 21], graph learning [22]. However, except for the retraining-based approach [23], there is little work for deep neural network models.

Generative adversarial network. GAN was first proposed by Goodfellow et al. [14] to estimate generative models via an adversarial learning process between two models. The common GAN comprises a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$. Let $x \sim \mathcal{X}_r$ denote the real data distribution and $x_f \sim \mathcal{X}_g$ denote the fake results provided by $\mathcal{G}$. $\mathcal{G}$ and $\mathcal{D}$ play the following min-max game:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathop{\mathbb{E}}_{x\sim\mathcal{X}_r}[\log(\mathcal{D}(x))] + \mathop{\mathbb{E}}_{x_f\sim\mathcal{X}_g}[\log(1-\mathcal{D}(x_f))]. \tag{1}$$

Followed by Goodfellow, a series of GANs with different characteristics were developed, e.g., DCGAN [24], CycleGAN [25]. In MuGAN, we treat the target model as the initialized $\mathcal{G}$ and randomly initialize a neural network as $\mathcal{D}$ to implement machine unlearning and mitigate the negative effect of OOD data during the finetuning process.

Out of distribution data. In an ideal setting, the training dataset of a machine learning model is independent and identically distributed, called ID data. However, the data collected from real-world applications hardly meet such a condition and are usually mixed with the data from out of distribution. Oliver et al. [26] proposed a class-distribution-mismatch concept to distinguish ID and OOD data, as shown in Definition 1. As referenced in [27], the machine learning model tends to make unexpected high confidence predictions when encountering OOD data, which sometimes results in performance drops and excessive-high false-positive rate.

Definition 1 (Out-of-distribution data)**.** In a $\mathcal{C}$-class classification task, the training data has the feature space $\mathcal{X}$ and label space $\mathcal{Y}$. Assuming there is a sample $x \in \mathcal{X}$ and can be labeled as $y_{\mathcal{C}} \in \mathcal{Y}$. We refer to such samples as ID data, while the other data belongs to OOD data.

3 Formulation of machine unlearning

For a better understanding of the design for MuGAN, we begin by formalizing the definition of machine unlearning used in the context of MuGAN. The meanings of notations used in the paper are summarized in Table 1. Considering machine unlearning, the general goal is to make the unlearned model equivalent to the one retrained with the original training data that excludes the forgotten data, called exact unlearning.

Definition 2 (Exact unlearning)**.** Given a neural network model $f_{t-1}(\mathcal{D}, \theta_{t-1})$ trained with dataset $\mathcal{D}$ and parameters θ_{t-1} in time $t-1$, and a forgetting set $\mathcal{D}' \subset \mathcal{D}$, exact unlearning is perfectly performed

Table 1 Notations

Notation	Description
$\mathcal{D}', \mathcal{D}, \mathcal{D}_{\text{non}}$	OOD data to be forgotten, the whole training dataset and the non-member dataset
λ	The weight coefficient
y_i	The ground truth
θ_0, θ_t	The t-th trainable parameters of the target model
$\mathcal{P}_{\text{in}}$, $\mathcal{P}_{\text{out}}$	The posterior probability distribution of member and non-member data
$\mathcal{P}$	The posterior probability distribution of ID data
Δ_{in}, Δ_{non}	The confidence vector set of member and non-member data
Δ	The confidence vector set of ID data

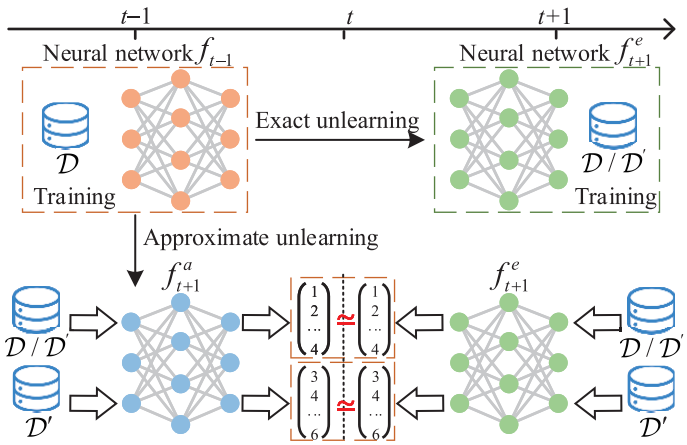

Figure 1 (Color online) The exact and approximate unlearning approach at time t. In exact unlearning, neural network f_{t-1} and f^e_{t+1} are regarded as being trained by dataset $\mathcal{D}$ and $\mathcal{D}/\mathcal{D}'$, respectively. In approximate unlearning, the model f^a_{t+1} keeps the inference results of dataset $\mathcal{D}$ unchanged.

if there exits a function φ that can output $f_t(\mathcal{D}/\mathcal{D}', \theta_t) = \varphi(f_{t-1}(\mathcal{D}, \theta_{t-1}))$.

However, due to the high complexity of inner architecture, it is struggling to make the unlearning of non-adaptive learning models such as neural networks meet the above strict definition unless conducting full-retraining whose price is almost equivalent to training a new model. Moreover, the condition for implementing retraining is to keep all the raw data available. Clearly, the above computational and storage resource requirements cannot be acceptable for smart IoT devices. As a consequence, we weaken the condition and define a new concept of machine unlearning called approximate unlearning from the perspective of model functionality similarity [28]. In terms of input data and prediction results, the model obtained by approximate unlearning has a relatively similar function to the retraining one [29].

Definition 3 (Approximate unlearning)**.** Given a neural network model $f_{t-1}(\mathcal{D}, \theta_{t-1})$ training with dataset $\mathcal{D}$ and parameters θ_{t-1} in time $t-1$, an exact unlearning function φ, and a forgetting set $\mathcal{D}' \subset \mathcal{D}$, approximate unlearning is perfectly performed if there exists a function ψ that can output $K(\varphi(f_{t-1}(\mathcal{D}, \theta_{t-1})), \psi(f_{t-1}(\mathcal{D}, \theta_{t-1})), x) < \epsilon$, where $x \in \mathcal{D}$, $K(f_1, f_2, x)$ represents the distance between the contribution of sample x in models f_1 and f_2, and $\epsilon > 0$ is the tolerant distance.

As shown in Figure 1, the model f^e_{t+1} applied with exact unlearning can be understood as being equivalent to the model trained using dataset $\mathcal{D}/\mathcal{D}'$, as if dataset $\mathcal{D}'$ has not been encountered ever. Compared with exact unlearning, approximate unlearning only requires enough similarity of the unlearned model to the ideal condition. To some extent, whether the samples have contributed to the model can be reflected by whether the model has the right results on those data [29]. For the decision results of samples belonging to dataset $\mathcal{D}/\mathcal{D}'$, the model f^a_{t+1} should be as similar as possible to the model f^e_{t+1}.

Since it is resource-intensive to obtain the retraining model to implement Definition 3, which is basically unrealistic in smart IoT. Therefore, we propose to make a direct comparison of the model function similarity before and after the execution of approximate unlearning. As shown in Figure 2, model f_{t-1} is able to infer the correct results because the training dataset includes both ID and OOD data. With the accuracy of the ID data remaining unchanged, model f^a_{t+1} gets the wrong results for the OOD data, i.e.,

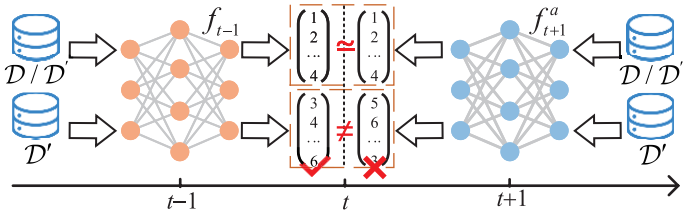

Figure 2 (Color online) The ideal approximate unlearning approach at the time t before and after t achieves the model $f(\cdot)$ keeps the inference results of data $\mathcal{D}/\mathcal{D}'$ unchanged, while the result for data $\mathcal{D}'$ is completely different and wrong.

forgetting the effect of the same distribution as OOD data. In this paper, we discuss the performance of machine unlearning between f_{t-1} and f^a_{t+1}. We say that approximate unlearning is enough for smart IoT scenarios because the goal of MuGAN focuses on making the recovery of performance drop caused by OOD data to be practical to real-world applications. Comprehensively, considering the balance of practicality and performance improvement, approximate unlearning can better satisfy our requirement and can be used to rule the adversarial game in MuGAN to achieve data unlearning.

4 Motivation & intuition of MuGAN

In this section, we first illustrate the motivation for unlearning the OOD data for smart IoT systems. Then, we overview the design intuition and goals of MuGAN.

4.1 Motivation

MuGAN is mainly motivated by the practical requirement of neural network applications in smart IoT. Referring to autonomous driving, most autopilot providers are currently obtaining models using simulation data that are focused on all domains [30]. In fact, different characteristics of the user's long-term driving environment will cause the model perception to focus on different inference capabilities [31, 32]. For example, northern China is usually covered by heavy snow, and a central model trained on data collected from all over the nation will lead to low accuracy in this specific scenario [31]. To improve the performance, the sensors collect data during driving and then conduct finetune functional upgrades based on the central model to obtain local models.

However, in an open world, the images collected by the visual sensors usually include OOD data, such as pure black or blurred visual data due to a leaf or a drop of water [8, 33]. Although these OOD data represent only a fraction of the local data, the direct use of those data results in reduced performance [9] (detailed in Subsection 6.3) and reliability of the model [34]. For example, the model remembers an OOD sample containing a raindrop picture and its corresponding operation of spinning the wheel sharply. When the sample is encountered again, the model will make an incorrect decision of spinning the wheel with overconfidence. Can a model trained in a dataset with OOD data be rolled back to a state where no OOD data is encountered? As discussed in Section 3, machine unlearning is able to achieve this goal. In other words, after performing machine unlearning, i.e., removing the impact of the OOD data, the model can recover lost performance due to OOD data and reject OOD samples or hand this over to human users for safety [33]. Therefore, mitigating the side effects of OOD data is necessary for the autonomous driving system. The existing MU algorithms focus on the effect of forgetting OOD data, while in smart IoT scenarios, the performance of the model should be guaranteed first and foremost. Therefore, from a practical perspective, effective and economical MuGAN is just proposed to mitigate the side effects by forgetting specific OOD data for NN architecture.

Shown as in Figure 3, the pretrain-finetune framework consists of three main steps: first of all, a pre-trained model is trained by the existing data and then the model parameters (until the cut layer) are encrypted and distributed to each IoT device. Encryption can be done using any method that supports forward inference, e.g., homomorphic encryption. Since encryption is not our main contribution, we will not go into details here. Then, all devices finetune the unencrypted layers with a continuous stream of data, constantly updating their local models f_R. The OOD data $\mathcal{D}'$ mixed in this process have negative effects on the model. Finally, IoT devices actively mitigate the side-effects of OOD data locally and recover the model f_R to f_I which can be regarded as trained by $\mathcal{D}/\mathcal{D}'$.

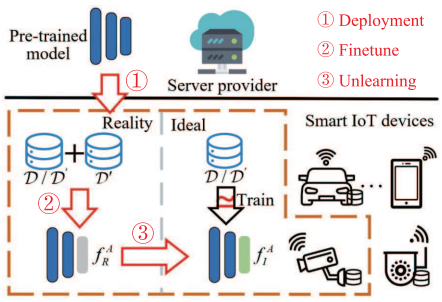

Figure 3 (Color online) Workflow of MuGAN in smart IoT scenario. Specifically, (1) the pre-trained model is trained from a certain number of samples collected by the service provider. (2) In reality, IoT devices get local model $f_R^{\cdot}$ by finetuning the pre-trained model with the dataset mixed in OOD data D'. (3) IoT devices invoke MuGAN to make the model recover to the ideal state $f_I^{\cdot}$, as if the OOD data were never encountered ever.

4.2 Overview of MuGAN

Referring to the prior studies of machine unlearning, e.g., FU [11] and SISA [10], the gradient is a powerful tool to change the memorization of a neural network. Therefore, MuGAN also operates the gradients to implement machine unlearning. Theoretically, if the direction of the model update, i.e., the gradients, can be oriented, the forgetting of OOD data can be effectively achieved. From the previous studies [35, 36], it can be found that the model has overconfidence in OOD data existing in the training set, while the posterior distribution of the non-member data is relatively uniform. If the posterior probability vector distribution of the OOD data can be similar enough to that of the non-member data, Definition 3 can be achieved. The core intuition of MuGAN comes from the adversarial learning idea in GAN. Unlike the original GAN's random initialization of the model, the generator of MuGAN is the target model. Meanwhile, MuGAN newly designs two tasks to guarantee the continuous usability and unlearning of the target model. The gradients used in MuGAN are a combination of the original optimization of the target model and transmitted back by the discriminator.

Specifically, we set the target model as a generator and introduce a neural network capable of identifying overconfidence vectors of the target model output as a discriminator. The target model tries to remove the influence of the OOD data, that is, the output of the OOD data no longer contains high confidence prediction results. In the end, the final output vectors of data set D' and D_{non} are similar enough to satisfy the above requirements, as shown in Figure 4.

4.3 Goals

Specifically, in order to implement Definition 3, MuGAN is designed to accomplish the following goals:

(1) Usability. To avoid catastrophic forgetting, the first goal of MuGAN is to guarantee that after machine unlearning, the performance of the target model should be retained as much as possible to ensure the usability of the model. Given a NN model f_{t-1}, the machine unlearning is to obtain a model f_{t+1} that makes $|A_{\mathcal{D}/\mathcal{D}'}(f_{t-1}) - A_{\mathcal{D}/\mathcal{D}'}(f_{t+1})|$ as small as possible, where $A_D(f)$ represents the accuracy of model f on data D.

(2) Unlearning. The unlearning set (i.e., OOD data for this paper) should be unlearned as defined by Definition 3 to mitigate its side-effect on the target model (reflected in the target model performance and unlearning rate as discussed in Section 6). Given a NN model f_{t-1}, the machine unlearning is to obtain a model f_{t+1} that makes $|A_{\mathcal{D}'}(f_{t-1}) - A_{\mathcal{D}'}(f_{t+1})|$ as big as possible, where $A_D(f)$ represents the accuracy of model f on data D.

(3) Practicality. Considering practicality, MuGAN should be capable of being realized efficiently in the pretrain-finetune framework of edge computing in smart IoT (less running time and storage space consumption).

5 MuGAN

In this section, we first illustrate the approach in detail and then discuss how to deploy MuGAN in IoT applications.

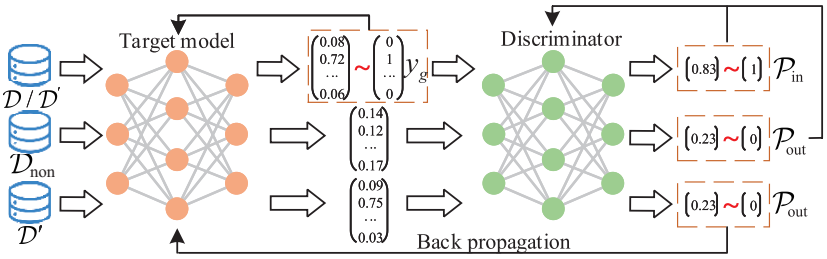

Figure 4 (Color online) Approach of MuGAN. Given the OOD dataset $\mathcal{D}'$, ID dataset $\mathcal{D}/\mathcal{D}'$, non-member dataset $\mathcal{D}_{\text{non}}$, (1) we first construct $\mathcal{P}_{\text{in}}$ for ID dataset $\mathcal{D}/\mathcal{D}'$ and $\mathcal{P}_{\text{out}}$ for non-member dataset $\mathcal{D}_{\text{non}}$ and train the discriminator. (2) Then the target model is updated with the ID data samples, its ground truth y_g, and OOD dataset $\mathcal{D}'$. After multiple updates, the confidence vectors of $\mathcal{D}'$ output from the target model are identified by the discriminator as $\mathcal{P}_{\text{out}}$.

5.1 Approach of MuGAN

Algorithm 1 outlines the steps of MuGAN to implement OOD data unlearning. Let f_{θ_0} denote the target neural network model, where θ_0 is the original model parameters, as shown in Figure 4. For Goal 1, MuGAN newly designs a penalty term to ensure that the target model keeps the memory of the ID data features. For Goal 2, to remove the influence of OOD data, MuGAN aims to make the posterior probability distribution of the OOD data similar enough to that of the non-member data. Simultaneously, we build a discriminator D_{θ_0} whose parameters θ_0 are randomly generated. The role of the discriminator is to extract features from the confidence vector output of the target model, ideally to directly determine whether a sample is in the training set of the target model or not. To obtain a usable discriminator, we select a small number of member and non-member samples to represent whether the sample belongs to the training set or not, respectively. The member data comes from the ID data $\mathcal{D}/\mathcal{D}'$, and the non-member data are samples from other domains that the model has not encountered.

Algorithm 1 Machine unlearning with GAN (MuGAN)

Input: The target model f_{θ_0} and its parameters θ_0; the unlearning set $\mathcal{D}' \subset \mathcal{D}$; the non-member OOD data $\mathcal{D}_{\text{non}}$; the maximum adversarial iterations T; the weight coefficient λ; the posterior distribution of member/non-member data $\mathcal{P}_{\text{in}}$ and $\mathcal{P}_{\text{out}}$; the posterior distribution of ID data $\mathcal{P}$;
Output: The target model f_{θ_T};
After initializing the discriminator D_{θ_0}, the target model f_{θ_0} and the discriminator $\mathcal{D}_{\theta_0}$ do the following iteration;
for $t \leftarrow 1$ to T **do**
  Freeze the target model f_{θ_t};
  Compute $\Delta_{\text{in}} = \{y_i | y_i \leftarrow D_{\theta_t}(f_{\theta_t}(x_i)), x_i \in \mathcal{D}/\mathcal{D}'\}$;
  Compute $\Delta_{\text{non}} = \{y_{\text{non}} | y_{\text{non}} \leftarrow D_{\theta_t}(f_{\theta_t}(x_{\text{non}})), x_{\text{non}} \in \mathcal{D}_{\text{non}}\}$;
  Optimize $\mathcal{D}_{\theta_t}$ by minimizing $\mathcal{L} = \mathcal{L}_{\text{BCE}}(\Delta_{\text{in}}, \mathcal{P}_{\text{in}}) + \mathcal{L}_{\text{BCE}}(\Delta_{\text{non}}, \mathcal{P}_{\text{out}})$;
  Freeze the discriminator $\mathcal{D}_{\theta_t}$;
  Compute $\Delta = \{y_i | y_i \leftarrow f_{\theta_t}(x_i), x_i \in \mathcal{D}/\mathcal{D}'\}$;
  Compute $\Delta_o = \{y_i | y_i \leftarrow D_{\theta_t}(f_{\theta_t}(x_i)), x_i \in \mathcal{D}'\}$;
  Optimize f_{θ_t} by minimizing $\mathcal{L} = \lambda_1 \cdot \mathcal{L}_{\text{CE}}(\Delta, \mathcal{P}) + \lambda_2 \cdot \mathcal{L}_{\text{BCE}}(\Delta_o, \mathcal{P}_{\text{out}})$;
end for
return The target model f_{θ_T}.

Specifically, at the tth iteration of training, we train f_{θ_t} and D_{θ_t} separately, that is, only perform forward inference operations on the other. For the member dataset, we input each sample $x_i \in \mathcal{D}/\mathcal{D}'$ into f_{θ_t}, and immediately input the result into D_{θ_t} to get $\Delta_{\text{in}} = \{y_i | y_i \leftarrow D_{\theta_t}(f_{\theta_t}(x_i)), x_i \in \mathcal{D}/\mathcal{D}'\}$. $f_\theta(x_{\text{o}})$ and $D_\theta(x_{\text{o}})$ represent the output confidence vectors of the target model and discriminator on the member samples, respectively, which reveals how well the target model learns the sample. Then we label the member samples as positive, which means that the sample is "in" the training dataset. Meanwhile, for the non-member data, the same operation is performed except for different label values (negative and "not in"). The objectives are expressed as the following formulas:

$$\mathcal{L}_i(D_\theta) = \mathbb{E}_{x_i \sim \mathbb{P}_{\text{in}}}[\text{BCE}(D_\theta(f_\theta(x_i)), \mathcal{P}_{\text{in}})], \tag{2}$$

$$\mathcal{L}_n(D_\theta) = \mathbb{E}_{x_{\text{non}} \sim \mathbb{P}_{\text{non}}}[\text{BCE}(D_\theta(f_\theta(x_{\text{non}})), \mathcal{P}_{\text{out}})], \tag{3}$$

where $\mathcal{P}_{\text{in}}$ and $\mathcal{P}_{\text{out}}$ denote the posterior distribution of member and non-member, BCE$(\cdot)$ denotes binary cross-entropy function, $f_\theta(x)$ represents the confidence vector obtained from the input of x into $f_\theta(\cdot)$.

To accomplish the identification of member and non-member samples, $\mathcal{D}_{\theta_t}$ updates its parameters by minimizing the following equation:

$$\mathcal{L}(D_\theta) = \mathcal{L}_i(\Delta_i, \mathcal{P}_{\text{in}}) + \mathcal{L}_n(\Delta_{\text{non}}, \mathcal{P}_{\text{out}}). \tag{4}$$

Corresponding to Goals 1 and 2, the target model is required to accomplish two tasks: (1) eliminate the negative impact of OOD data, (2) keep the ID data accuracy retained as much as possible. To meet the above tasks, two loss functions need to be calculated as the following formulas:

$$\mathcal{L}_{\text{unl}}(f_\theta) = \mathbb{E}_{x_{\text{o}} \sim \mathbb{P}_{\text{out}}}[\text{BCE}(D_\theta(f_\theta(x_{\text{o}})), \mathcal{P}_{\text{out}})], \tag{5}$$

$$\mathcal{L}_{\text{cat}}(f_\theta) = \mathbb{E}_{(x_i, y_i) \sim \mathbb{P}_{\text{in}}}[\text{CE}(f_\theta(x_i), y_i)], \tag{6}$$

where $\text{CE}(\cdot)$ represents cross-entropy function, y_i is the ground truth of ID sample. Note that the dimension of $\mathcal{P}_{\text{in}}$ and $\mathcal{P}_{\text{out}}$ is consistent with the corresponding function output. For the above two completely separate tasks, f_{θ_t} is performed in a differentiated degree. Overall, f_{θ_t} updates its parameters by minimizing

$$\mathcal{L}(f_\theta) = \lambda_1 \cdot \mathcal{L}_{\text{unl}}(\Delta_o, \mathcal{P}_{\text{out}}) + \lambda_2 \cdot \mathcal{L}_{\text{cat}}(\Delta, \mathcal{P}), \tag{7}$$

where λ represents weight coefficient, $\mathcal{P}$ denotes the posterior probability distribution of ID data. Δ is computed for each sample of the unlearning set $x \in \mathcal{D}'$, and Δ_o is calculated for each normal sample of $x \in \mathcal{D}/\mathcal{D}'$. Recall Definition 3, the discriminator plays the role of measuring the distance between the contribution of samples. The implementation of Definition 3 is equivalent to that the output of the target model to all data can fool the discriminator. In MuGAN, a small amount of ID, OOD, and non-member data is needed to maintain individual probability distribution heterogeneity, which can be stored in advance or dynamically updated during the finetuning process. For ID data, users can update them by selecting the sample with the highest confidence vector.

5.2 How to apply MuGAN in applications

Given the design details of MuGAN, we give the specific algorithm in pretrain-finetune framework mentioned in Subsection 4.1, shown as Algorithm 2. In MuPFF, we draw on the idea in [37] and view the learning model as base + personalization layers, which respectively represent the layers from bottom to the cut layer c and others, denoted by $-c$ and $c-$. Since many finetune methods have been proposed and are not our contribution, we do not discuss them anymore. The server provider gets semi-encrypted model $f^p_{\theta_0^{c-}}$ by selecting the cut layer and encrypting base layers, with the most valuable part of the model preserved. Devices receive the pre-trained model and finetune the personalization layers with data $\mathcal{D}^i_{\epsilon_1}$ that contains OOD data to obtain the local model. When the system detects OOD data or the user has an active operation, the device calls MuGAN to mitigate the side-effects of OOD data using $\mathcal{D}^i_{\epsilon_2}$, which is kept in the same distribution with $\mathcal{D}^i_{\epsilon_1}$. Note that $\mathcal{D}^i_{\epsilon_2}$ has a fewer samples than $\mathcal{D}^i_{\epsilon_1}$ but one more type of non-member data. From line 7 in Algorithm 2, it can be found that MuGAN achieves its design goal for practicality (Goal 3). As an expansion, the service provider can use an aggregation strategy (e.g., FedAvg) to obtain a stronger model.

Algorithm 2 Machine unlearning in the pretrain-finetune framework (MuPFF)

Input: Device $u_i \in U = \{u_1, u_2, \ldots, u_N\}$ and its local data $\mathcal{D}^i_\epsilon$; the posterior probability distribution of member and non-member data $\mathcal{P}_{\text{in}}$ and $\mathcal{P}_{\text{out}}$; the cut layer c; the loss function $\mathcal{L}$;
Output: The local model $f^{u_i}_{\theta_T}$;
1: Preset the posterior distribution of member and non-member data $\mathcal{P}_{\text{in}}$ and $\mathcal{P}_{\text{out}}$ in each device;
2: Build a pre-trained model $f^p_{\theta_0}$ using existing data;
3: Select the cut layer c, and encrypt base layers, then distribute semi-encrypted model $f^p_{\theta_0^{c-}}$ to all devices;
4: **for** $i \leftarrow 1$ to N **do**
5: **for** $t \leftarrow 1$ to T **do**
6: u_i gets $f^{u_i}_{\theta_t^{c-}}$ by finetuning $f^p_{\theta_t^{c-}}$ using local data $\mathcal{D}^i_{\epsilon_1}$;
7: u_i initializes a neural network D_i, and then optimizes $f^{u_i}_{\theta_t^{c-}}$ by minimizing $\mathcal{L}_i$ using data $\mathcal{D}^i_{\epsilon_2}$;
8: **end for**
9: **end for**
10: **return** The local models $f^{u_i}_{\theta_T}$.

Table 2 Experiment datasets

Abbreviation	ID data	OOD data
C10.T	CIFAR10	TinyImage
C10.S	CIFAR10	STL-10
C10.L	CIFAR10	LSUN(resize)
C10.i	CIFAR10	iSUN
GTS.T	GTSRB	TinyImage
GTS.L	GTSRB	LSUN(resize)

6 Experiment

In this section, we normalize two indicators from the perspective of accuracy to evaluate the effectiveness of machine unlearning more effectively. Then we conduct comprehensive experiments to demonstrate the performance of MuGAN, including comparing it with the two existing studies, FU [11] and SISA [10] under multiple network architectures and datasets.

6.1 Experiment setup

Dataset. We implement MuGAN on dataset CIFAR10, GTSRB [38], TinyImage, STL-10[1)], LSUN[2)], and iSUN [39]. German traffic sign recognition benchmark (GTSRB), a traffic sign dataset with 43 classes, is widely used to simulate the model performance of autonomous driving in smart IoT. Especially, TinyImage, STL-10, LSUN, and iSUN are used as OOD data, where LSUN and GTSRB are resized as 32×32 by downsampling. Table 2 summarizes the usage and abbreviation of the dataset. Similar to [12], we simulate two realistic situations that would be encountered. (1) The ID data and OOD data belong entirely to different learning tasks. For example, CIFAR10 has ten classes (e.g., airplane, automobile, bird). TinyImage contains 200 classes (e.g., goldfish, bullfrog, trilobite). We randomly select one or more classes and label them as one or more classes in CIFAR10, for example, label goldfish as airplane. (2) The ID data and OOD data belong to the same learning task. For example, both CIFAR10 and STL-10 contain ten classes (e.g., airplane, car, bird). We randomly select samples of a particular class and label them as other classes, for example, label airplane as car, both of which belong to the ID data.

Neural networks. For the target model, we adopt three network structures to process the above datasets, namely VGG16 [40], ResNet18, and ResNet34 (-V, -R18, and -R34 for short), whose parameter size are 14.73, 11.17, and 21.28 M, respectively. At finetune stage, we use the configuration in [37] for reference, and the personalization layer is set to 2 blocks + classifier. For discriminator, we use three layers fully connected network structure to ensure that the gradients of the unlearning data can be transmitted back to the target network.

Other setting. To better simulate the data heterogeneity in smart IoT, we limit the local data of each particular to include up to 60% of all classes [41, 42], where a certain class data accounts for most of them, called personalized data. The default ID data include 2000 samples of personalized data and 500 samples of other classes. For GTSRB, simple data augmentation, such as rotation and mirroring, is performed to satisfy this. Note that the mixed OOD data replaces the above data proportionally. The OOD data size for each class is 450, and each class is labeled with the same label. For example, an OOD number of 900 represents two classes, 450 in each class. The amount of data used to execute MuGAN (we call them adversarial sample) is by default half of the training data, and non-member unlearning samples are equal to member samples. The optimizer is stochastic gradient descent (SGD), learning rate = 0.01, batch size = 128. The experiments are performed on a workstation equipped with NVIDIA GeForce RTX 2080 Graphics Card and 32 GB RAM in a single thread.

6.2 Indicators

To better evaluate the performance of unlearning, we investigate current papers and algorithms related to machine unlearning, and the results show that they generally use the drop in test accuracy as the evaluation criterion [10, 43, 44]. Indeed, for the unlearning of a sample, the most direct criterion is whether the neural network remembers it. If the neural network classifies this sample incorrectly, it

1) https://cs.stanford.edu/acoates/stl10/.
2) https://www.yf.io/p/lsun.

Table 3 Indicator notations

Notation	Description
AO	The test accuracy of $\mathcal{D}'$ AFTER conducting unlearning method
BO	The test accuracy of $\mathcal{D}'$ BEFORE conducting unlearning method
AI	The test accuracy of $\mathcal{D}/\mathcal{D}'$ AFTER conducting unlearning method
BI	The test accuracy of $\mathcal{D}/\mathcal{D}'$ BEFORE conducting unlearning method

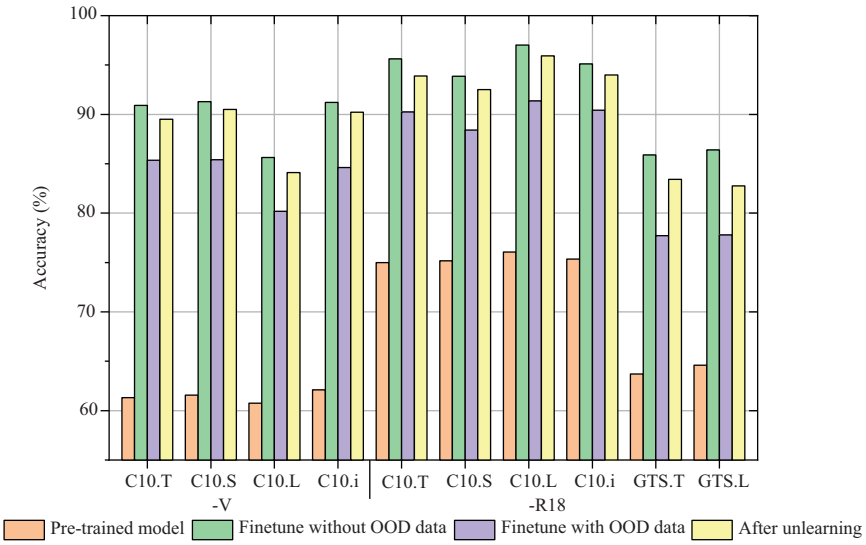

Figure 5 (Color online) The accuracy of personalized data in four stages.

intuitively means that the neural network is no longer sensitive to this sample and can be regarded as finishing unlearning. However, because the initial values of the specific model are variable and not reproducible, the presentation of experimental data is often less intuitive and does not allow for accurate and valid comparisons. For example, before and after conducting machine unlearning, the number of samples remembered by the model changes from 100 to 50 and from 70 to 20. Although the change is 50 for both, it is clear that the unlearning effect is different, with the latter being more effective. Inspired by [12], we redefine unlearning rate (UR) of unlearning data $\mathcal{D}'$ and retention rate (RR) of ID data $\mathcal{D}/\mathcal{D}'$ based on the accuracy standard. Specifically, UR represents the sensitivity of the neural network to unlearning data before and after the execution of machine unlearning, i.e., the accuracy changes of classification result of unlearning data from right to wrong, which can be formulated as

$$\mathrm{UR} = \frac{\mathrm{BO} - \mathrm{AO}}{\mathrm{BO}}. \tag{8}$$

Obviously, the parameters of the neural network cover the impact of every sample in the training dataset, including unlearning data and ID data. Up to now, machine unlearning also has a negative impact on ID data, which in turn affects the classification accuracy of the model on $\mathcal{D}/\mathcal{D}'$. Therefore, to measure the retention of ID data, RR is introduced:

$$\mathrm{RR} = 1 - \frac{\mathrm{BI} - \mathrm{AI}}{\mathrm{BI}}. \tag{9}$$

UR and RR can be utilized to evaluate the percentage of corresponding unlearning or retention levels after conducting unlearning. The symbols used in (8) and (9) are shown in Table 3. The above are two indicators for evaluating the unlearning method from the perspective of accuracy. Consider the ideal scenario where the ID data accuracy is kept constant while the unlearning data accuracy is reduced to 0%. In summary, one that meets high UR and RR is an effective unlearning method, that is, the goal of the model is to maintain the accuracy of ID data and reduce the accuracy of unlearning data.

6.3 Changes in performance

Accuracy of four stages. Shown in Figure 5, we illustrate the accuracy changes for testing personalized data at four stages: original pre-trained model, finetune without OOD data, finetune with OOD data, and

Table 4 The RR and UR of MuGAN, FU, SISA, and DOB (%)

Dataset	MuGAN		FU [11]		SISA [10]		DOB [23]	
	RR ↑	UR ↑	RR ↑	UR ↑	RR ↑	UR ↑	RR ↑	UR ↑
C10.T-V	**99.87**	**91.13**	34.87	85.06	82.44	90.63	78.96	89.4
C10.S-V	**99.95**	87.73	23.76	80.9	80.7	89.45	78.4	**90.62**
C10.L-V	**98.3**	90.28	30.81	86.49	79.11	**91.67**	68.54	90.99
C10.i-V	**98.4**	**92.7**	32.93	88.08	74.27	91.97	67.43	90.87
C10.T-R18	**95.71**	**81.04**	40.35	79.2	76.89	75.82	73.46	77.37
C10.S-R18	**94.8**	83.06	35.06	71.53	81.36	**86.04**	76.47	85.84
C10.L-R18	**95.06**	**83.33**	36.49	78.57	77.17	82.38	68.87	80.9
C10.i-R18	**94.87**	**86.1**	38.84	74.12	78.09	83.16	74.3	82.35
GTS.T-R34	**98.22**	97.29	52.16	91.06	92.12	97.2	86.26	**97.39**
GTS.L-R34	**94.74**	96.13	36.5	88.32	88.07	**97.1**	84.06	96.12

after conducting MuGAN on two network architectures. The pre-trained model is obtained from all users by the FedAvg algorithm [45]. The results show that in scenarios where data heterogeneity on different devices exists, local finetuning has significant performance improvement in all datasets and network architectures, greatly learning fine-grained information on specific devices. Furthermore, a comparison of without OOD data and with OOD data clearly shows that the OOD data have a negative impact on the model performance. Moreover, although the parameter size of VGG16 and ResNet18 is relatively similar, ResNet18 obtains more competitively robust results than VGG16 due to the residual structure to optimize the degradation problem, especially when finetuning only the personalization layers. Besides, the experimental results show that MuGAN successfully reduces the performance drop off personalized data from 5.88% to 0.8%, which can be considered to be basically restored to the state where only finetune without OOD data.

6.4 Comparison with prior work

Unlearning rate and retention rate. Shown in Table 4, we compare the two indicators proposed in Subsection 6.1 of ID and OOD data on MuGAN with the existing three studies, FU [11], SISA [10] and (DeepOBliviate) DOB [23], on two datasets and three network architectures. The best results are bolded. Specifically, for FU, we record all the OOD data to be forgotten. When the device invokes FU, the gradients generated by OOD data are directly subtracted from a normal iteration and the target model is updated. For SISA, referring to [46], we set shards to 10, and each shard trains a constituent model by orderly and incrementally letting 10 slices of the shard participate in the model training. For DOB, we set $\varepsilon = 0.04$ to achieve a higher consistent and accurate unlearning as discussed in [23]. The aggregation algorithm adopts the posterior average, which is the mean value of all constituent models. The remaining hyperparameters of the constituent models, like learning rate, are the same as MuGAN. Note that the order of all samples for SISA is fixed, i.e., the original data is used for the whole process, while the data for MuGAN are randomly selected from a large pool of samples that maintained the characteristics of the same distribution.

From the comparison results, it can be observed that FU has the worst performance in retention rate which is because the directly forced subtraction of the gradients reflected by OOD data innocently and strongly affects the performance of ID data, causing catastrophic forgetting. On the other hand, the unlearning rate of FU is not much worse, which corresponds to our statement in Subsection 4.2 that to some extent, the gradients represent the samples. Overall, MuGAN performs better than SISA in most of the dataset, except for C10.S, because this dataset belongs to the same classification task that has the same distribution. The gradients generated by OOD data have a high similarity to the gradients of the ID data, causing interference with the target model, which conversely leads to a higher retention rate of MuGAN. The reason that SISA and DOB both have good UR is that they remove the OOD data from the training data level and completely retrain. The UR is higher on the GTSRB dataset than on CIFAR10 because the 43 classes make the AO smaller. For the usability of the model (Goal 1), the results show that MuGAN has better performance than others. This is because MuGAN is optimized for the characteristics in the smart IoT scenario, where usability is in the first place. While the remaining machine unlearning methods focus on eliminating data forgetting, which leads to severe accuracy drops in normal data.

Table 5 The accuracy drop of ID and OOD data (%)

Dataset	MuGAN		FU [11]		SISA [10]		DOB [23]	
	ID ↓	OOD ↑	ID ↓	OOD ↑	ID ↓	OOD ↑	ID ↓	OOD ↑
C10.T-V	**0.13**	**65.46**	54.25	61.17	14.63	65.17	17.53	64.22
C10.S-V	**0.04**	71.85	63.31	66.26	16.03	73.26	17.93	**74.22**
C10.L-V	**1.37**	65.37	53.72	62.27	16.81	**66.74**	24.1	65.51
C10.i-V	**1.34**	**63.51**	56.21	60.34	21.62	63.15	27.3	62.25
C10.T-R18	**3.87**	**48.41**	53.87	47.32	20.87	45.30	23.97	46.23
C10.S-R18	**4.57**	60.33	57.08	51.93	16.38	**62.47**	20.68	62.33
C10.L-R18	**4.46**	**52.58**	57.38	49.50	20.63	51.94	28.13	50.97
C10.i-R18	**4.61**	**58.55**	54.96	50.47	19.69	56.55	23.09	56.71
GTS.T-R34	**1.45**	76.96	39.16	72.03	6.45	76.88	11.25	**77.04**
GTS.L-R34	**4.72**	73.84	57.06	67.84	10.72	**74.58**	14.32	73.83

Accuracy drop in ID and OOD accuracy. As discussed in Subsection 6.2, the drop in test accuracy is a straightforward way to evaluate the performance of machine unlearning methods. We also demonstrate the drops in ID and OOD accuracy before and after conducting machine unlearning, as shown in Table 5. Overall, the drop trends of ID and OOD data are consistent with the results of UR and RR. The reason why the accuracy of SISA decreases so much is that the less amount of data available after removing OOD data has a greater impact on the accuracy of continual learning. However, as discussed in Subsection 6.2, the drop in test accuracy is not intuitive enough. For example, MuGAN drops 63.51% accuracy at C10.i-V. This number is not intuitively high, but MuGAN has forgotten 92.7% of OOD data compared with the initial state.

Storage. To better understand the storage space required for MuGAN, FU, and SISA, we perform a theoretical analysis of the space complexity. Take the sizes of training data and eliminated data as N and n, respectively. Given one sample storage space s and one model storage space m, the space complexity of MuGAN is $O(n_s \cdot s)$, where n_s represents the size of samples used to execute MuGAN. FU and SISA need to record original data or intermediate model points; therefore, the storage complexity of FU and SISA are $O(n \cdot s)$ and $O(N \cdot s + m \cdot K)$, where K is the number of intermediate models needed to be recorded (i.e., calculated by shard $\cdot$ slice). The storage space of DOB remains essentially the same as that of SISA. Since $n_s < n$, $n_s \ll N$, and n_s take values independent of N, the storage resources of MuGAN are significantly less than the above prior studies. To some extent, MuGAN is memory-free because there is no need to remember any data but only some of the preset data, whilst the other two have to constantly keep allocating storage space over time.

Efficiency. Similar to storage, efficiency also plays a significant role in applying machine unlearning. Table 6 summarizes the running time of four methods on the CIFAR10 dataset. Take the sizes of the training data and unlearning data as N and n, respectively. Given the training data size n_t and the unlearning data size n_u when conducting MuGAN, the time complexity of MuGAN is $O((n_t + n_u) \cdot t + n_u \cdot t_d)$, where t and t_d represent the time to update the target model and discriminator. Since $t_d \ll t$, the time complexity can be approximately equivalent to $O((n_t + n_u) \cdot t)$. Commonly, from the experimental results mentioned in Subsection 6.5, good performance is obtained when $n_t = \frac{1}{2}N$ is satisfied. In the worst case, SISA retrains all samples except unlearning data from scratch; therefore, the time complexity is $O(K \cdot (N - n) \cdot t)$, where K represents the empirically selected training epochs. To make matters worse, DOB additionally needs to calculate residual memory to determine the affected areas. Therefore, DOB is even inferior to SISA when the amount of forgotten data is large. FU conducts one update iteration on the target model to directly subtract the contribution of the unlearning dataset, which needs to compute the gradients of all training data, so its time complexity is $O(N \cdot t)$. From the results, it can be observed that MuGAN is more effective, which is consistent with the results of the theoretical analysis.

6.5 Further understanding of MuGAN

Performance at different iteration. Shown in Figure 6, we illustrate the performance change (the accuracy of ID and OOD data) of MuGAN under different iteration rounds. Iteration 0 in the figure represents the initial performance of the target model before MuGAN is invoked. From the experimental results, in the first 5 iterations, the accuracy of ID data experiences a sharp drop and then rebounds, and finally basically reaches the performance before conducting MuGAN. This is because the discriminator

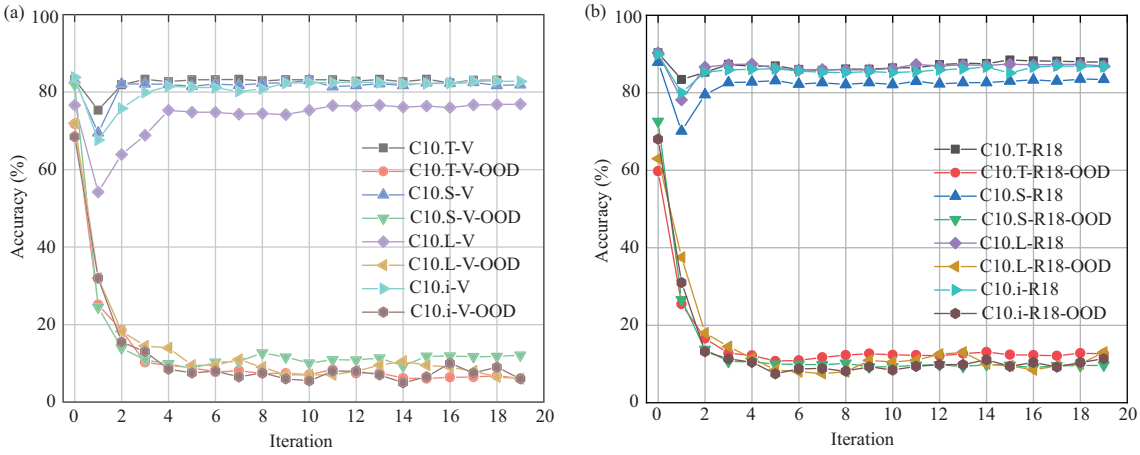

Figure 6 (Color online) The detailed performance of MuGAN at each iteration on (a) VGG16 and (b) ResNet18. -OOD means the accuracy of OOD data. Iteration 0 represents the initial performance before conducting MuGAN.

Table 6 Running time (s)

Dataset	MuGAN	FU [11]	SISA [10]	DOB [23]
C10.T	6.87	11.18	55.34	59.71
C10.S	6.71	10.95	55.18	59.57
C10.L	6.88	11.13	55.47	58.73
C10.i	6.76	11.39	55.61	59.58

is initialized at the beginning, and then the gradients transmitted back are relatively invalid. Besides, the accuracy of OOD data shows a significant decreasing trend, which is in line with our expectations. Overall, after about ten iterations, MuGAN is able to reach its best performance. Further, more iterations have little impact on the ID data performance of the target model, because MuGAN cannot get effective gradients anymore from discriminator during back-propagation when the unlearning rate of OOD data has reached a threshold, but rather ensures the accuracy of ID data to a greater extent by 6. Moreover, we can find that the accuracy of OOD data is in fluctuates after reaching the best performance. The reason is that the confidence vector of the target model for the non-member data is underconfident and is approximately equal to guessing the results.

Number of adversarial samples. To minimize the performance drop in the accuracy of ID data and capture the characteristics of OOD data, we have to use a small amount of data with the same distribution as the training data. It is obvious that the amount of data used to execute MuGAN (we call them adversarial sample) has an important impact on the effectiveness. Therefore, we evaluate different adversarial sample proportions with reference to the training data at the finetune stage, where 0.5 means the adversarial sample is half of the training data, including ID and OOD data. Figure 7 shows the experimental results, where Dif.Loss indicates the differences of loss before and after performing MuGAN. From the result, it can be found that UR and RR have a significant increase with the increase of the adversarial sample proportion when the proportion is less than half, and the best performance is reached at 0.5. After that, the increase of the adversarial sample proportion hardly causes a visible effect on UR and RR, but the testing loss is still decreasing. This is because our discriminator has already reached overtraining, but the ID data are still playing a role in updating the target model, which is equivalent to simply training the target model using training data. Moreover, once the adversarial sample proportion reaches 0.5, the decreasing trend of testing loss diminishes as the proportion increases, because the target model does not have to step as fast as before to make significant adjustments to the ID data. Considering only the unlearning rate, a proportion of 0.2 is sufficient to achieve a good performance, which significantly reduces the number of OOD data preset by the service provider.

Sizes of unlearning set. We evaluate the performance change of different sizes of unlearning samples on the basis of the default adversarial sample proportion, shown in Figure 8. Note that the increased number of unlearning samples makes the increase of UR and RR, and a reduction of loss difference, which is largely caused by the changes of BI and BO. Assuming constant AO and AI, the higher the number of unlearning samples, the larger and smaller the BO and BI, respectively, leading to the results shown in

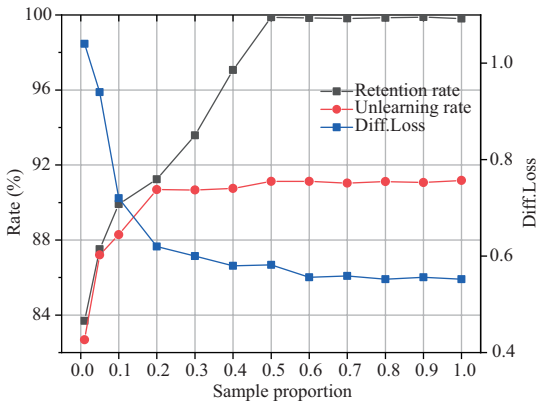

Figure 7 (Color online) The detailed performance (UR and RR) changes of target model at different adversarial sample proportions. Dif.Loss means the difference of target model loss before and after conducting MuGAN.

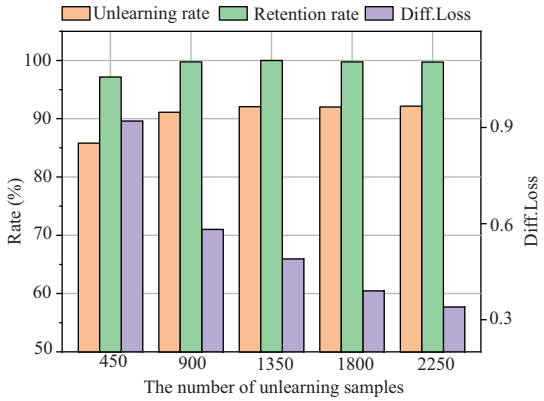

Figure 8 (Color online) The detailed performance (UR and RR) changes of target model at different sizes of unlearning samples. Dif.Loss means the difference of target model loss before and after conducting MuGAN.

Table 7 Accuracy (%) changing of the model after aggregation with and without executing MuGAN

Diff.acc	VGG16		ResNet18		ResNet34	
	ID	OOD	ID	OOD	ID	OOD
C10.T	1.86	52.6/1.81	3.94	39.6/7.1	4.25	50.11/5.89
C10.S	4.68	74.5/3.41	0.34	51.4/10.9	0.24	59.19/11.6
C10.L	6.74	43.5/2.9	0.31	38.5/6.5	1.02	37.3/9.5
C10.i	6.48	40.71/3.29	0.25	37.2/8.5	0.5	36.6/6.5

Figure 8. Besides, from UR and RR, we can observe that unlearning hardly affects the performance of the target model, even when the unlearning size reaches 2250. The phenomenon shows that MuGAN is robust to the different unlearning data sizes.

Scalability under the pretrain-finetune framework. To prove the scalability of our scheme in the pretrain-finetune framework, we evaluate the performance changes before and after conducting MuGAN after aggregation in the central server under different datasets and network architectures. We separately test the accuracy of ID and the mixed OOD data with and without conducting MuGAN for the aggregated model, i.e., the difference of accuracy (Diff.acc), shown in Table 7. Where the number behind the notation "/" represents the accuracy after execution and the number before it indicates the testing accuracy drops. From the results, the aggregated model basically drops the accuracy of OOD data to single digits, and the best reaches 1.81%, which can be equated to no memory at all of OOD data. At the same time, the ID data only caused about 0.25% of the accuracy rate drop, which can basically be ignored. Since the gradients of the residual network are directly transmitted back to the base layers, ResNet has better performance than VGG except for the final OOD accuracy. For the dataset C10.S, the final OOD accuracy with executing MuGAN is still on the high side because this network structure has a strong feature extraction capability; therefore, the OOD data belonging to the same learning task leads to more interference, which is also in line with VGG16 network.

Visualization. To show the effectiveness of MuGAN vividly, we visualize the confidence vectors of the target model before and after conducting MuGAN. We use t-distributed stochastic neighbor embedding (t-SNE) as the visualization tool, with detailed reference to [47]. On the dataset C10.T and VGG16 networks, we randomly select 150 training samples, test samples, and unlearning samples, respectively, for a total of 450 samples. The results of visualizing the top 3 confidence vectors of the target model are shown in Figure 9. Note that due to the randomization of t-SNE, the location of samples on the figure is shifted. In Figure 9(a), it is very clearly observed that the three types of samples are evenly scattered across the whole figure, implying that the target model remembers their distribution relatively similarly; i.e., the model can remember them with high confidence. In Figure 9(b) after executing MuGAN, the training and test samples are still evenly scattered together, which indicates that the target model still maintains usability and achieves Goal 1, i.e., high RR. Meanwhile, the unlearning samples are basically gathered in the upper right corner, in other words, the target model can be clearly distinguished from

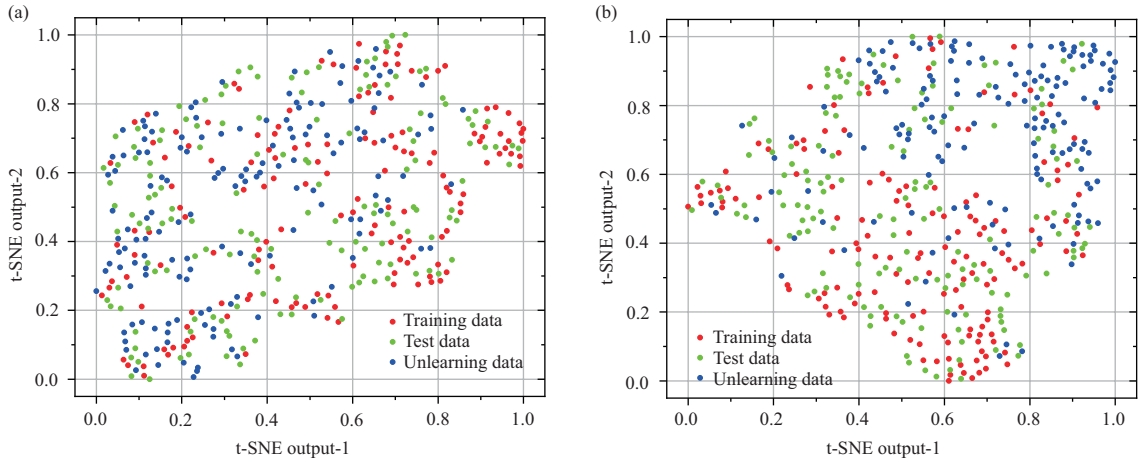

Figure 9 (Color online) Visualization of data distribution of the training data, test data, and unlearning data samples before (a) and after (b) conducting MuGAN for C10.T on VGG16 network. The figures show the dimensionality reduction results of top-3 of the confidence vector obtained from the same samples. (a) Before MuGAN; (b) after MuGAN.

the ID data, which is consistent with Goal 2, i.e., high UR.

7 Conclusion

In this paper, we discussed the existing problem of the side-effects of noisy data, e.g., OOD data, when the finetuning framework is deployed in smart IoT and proposed a novel machine unlearning method to mitigate the negative impact. Inspired by the adversarial idea in the generative adversarial network, we designed a new adversarial game and proposed MuGAN. In this approach, we build a discriminator capable of distinguishing the state of ID and non-member data according to the output of the target model. For the target model, the posterior probability for OOD data to be forgotten is as close as possible to the non-member data to confront the discriminator. Unlike previous studies, MuGAN accomplished the forgetting of OOD data with no need to record whole original data or intermediate models. Moreover, for the normal and eliminated data, we considered the changing of accuracy, i.e., retention rate and unlearning rate, to quantify the performance of unlearning. The results of extensive experiments proved that compared to the existing machine unlearning work, MuGAN had lower computational overhead and higher effectiveness.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2022YFB3103500), National Natural Science Foundation of China (Grant Nos. U21A20464, 61872283), Natural Science Basic Research Program of Shaanxi (Grant No. 2021JC-22), Key Research and Development Program of Shaanxi (Grant No. 2022GY-029), and China 111 Project (Grant No. B16037).

References

1 Peng B, Chi M M, Liu C. Non-IID federated learning via random exchange of local feature maps for textile IIoT secure computing. Sci China Inf Sci, 2022, 65: 170302
2 Jung J, Kim B, Cho J, et al. A secure platform model based on ARM platform security architecture for IoT devices. IEEE Internet Things J, 2022, 9: 5548–5560
3 Imteaj A, Thakker U, Wang S, et al. A survey on federated learning for resource-constrained IoT devices. IEEE Internet Things J, 2021, 9: 1–24
4 Khan L U, Saad W, Han Z, et al. Federated learning for Internet of Things: recent advances, taxonomy, and open challenges. IEEE Commun Surv Tutorials, 2021, 23: 1759–1799
5 Zhang T, Gao L, He C, et al. Federated learning for the Internet of Things: applications, challenges, and opportunities. IEEE Internet Things M, 2022, 5: 24–29
6 He T X, Liu J, Cho K, et al. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, 2021. 1121–1133
7 Krishnamurthi R, Kumar A, Gopinathan D, et al. An overview of IoT sensor data processing, fusion, and analysis techniques. Sensors, 2020, 20: 6076

8 Wu Z-F, Wei T, Jiang J W, et al. NGC: a unified framework for learning with open-world noisy data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021. 62–71

9 Wenzel F, Dittadi A, Gehler P V, et al. Assaying out-of-distribution generalization in transfer learning. 2022. ArXiv:2207.09239

10 Bourtoule L, Chandrasekaran V, Choquette-Choo C A, et al. Machine unlearning. In: Proceedings of IEEE Symposium on Security and Privacy (SP), 2021. 141–159

11 Cao Y Z, Yang J F. Towards making systems forget with machine unlearning. In: Proceedings of IEEE Symposium on Security and Privacy, 2015. 463–480

12 Ma Z, Liu Y, Liu X, et al. Learn to forget: machine unlearning via neuron masking. IEEE Trans Dependable Secure Comput, 2022. doi: 10.1109/TDSC.2022.3194884

13 Hsu T H, Wang Z H, See A R. A cloud-edge-smart IoT architecture for speeding up the deployment of neural network models with transfer learning techniques. Electronics, 2022, 11: 2255–2269

14 Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial networks. Commun ACM, 2020, 63: 139–144

15 Schelter S. "Amnesia"—machine learning models that can forget user data very fast. In: Proceedings of the 10th Conference on Innovative Data Systems Research, Amsterdam, 2020

16 Chen C, Sun F, Zhang M, et al. Recommendation unlearning. In: Proceedings of the ACM Web Conference, 2022. 2768–2777

17 Baumhauer T, Schöttle P, Zeppelzauer M. Machine unlearning: linear filtration for logit-based classifiers. Mach Learn, 2022, 111: 3203–3226

18 Izzo Z, Smart M A, Chaudhuri K, et al. Approximate data deletion from machine learning models. In: Proceedings of International Conference on Artificial Intelligence and Statistics, 2021. 2008–2016

19 Brophy J, Lowd D. Machine unlearning for Random forests. In: Proceedings of International Conference on Machine Learning, 2021. 1092–1104

20 Fu S P, He F X, Tao D C. Knowledge removal in sampling-based Bayesian inference. 2022. ArXiv:2203.12964

21 Rawat A, Requeima J, Bruinsma W, et al. Challenges and pitfalls of Bayesian unlearning. 2022. ArXiv:2207.03227

22 Chien E, Pan C, Milenkovic O. Certified graph unlearning. 2022. ArXiv:2206.09140

23 He Y Z, Meng G Z, Chen K, et al. Deepobliviate: a powerful charm for erasing data residual memory in deep neural networks. 2021. ArXiv:2105.06209

24 Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015. ArXiv:1511.06434

25 Zhu J, Park T, Isola P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision, 2017. 2223–2232

26 Oliver A, Odena A, Raffel C, et al. Realistic evaluation of deep semi-supervised learning algorithms. In: Proceedings of Advances in Neural Information Processing Systems 31, 2018

27 Morningstar W, Ham C, Gallagher A, et al. Density of states estimation for out of distribution detection. In: Proceedings of International Conference on Artificial Intelligence and Statistics, 2021. 3232–3240

28 Orekondy T, Schiele B, Fritz M. Knockoff nets: stealing functionality of black-box models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 4954–4963

29 Tramèr F, Zhang F, Juels A, et al. Stealing machine learning models via prediction APIs. In: Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), 2016. 601-618

30 Caesar H, Bankiti V, Lang A, et al. nuScenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 11621–11631

31 Pan Z Y, Emaru T, Ravankar A, et al. Applying semantic segmentation to autonomous cars in the snowy environment. 2020. ArXiv:2007.12869

32 Nakanoya M, Im J, Qiu H, et al. Personalized federated learning of driver prediction models for autonomous driving. 2021. ArXiv:2112.00956

33 Li Z, Pan M X, Zhang T, et al. Testing DNN-based autonomous driving systems under critical environmental conditions. In: Proceedings of International Conference on Machine Learning, 2021. 6471–6482

34 Li J N, Xiong C M, Hoi S C H. Learning from noisy data with robust representation learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021. 9485–9494

35 Zhang L, Goldstein M, Ranganath R. Understanding failures in out-of-distribution detection with deep generative models. In: Proceedings of International Conference on Machine Learning, 2021. 12427–12436

36 Ulmer D, Cinà G. Know your limits: uncertainty estimation with ReLU classifiers fails at reliable OOD detection. In: Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence, 2021. 1766–1776

37 Arivazhagan M G, Aggarwal V S, Aaditya K, et al. Federated learning with personalization layers. 2019. ArXiv:1912.00818

38 McMahan B, Moore E, Ramage D, et al. The German traffic sign recognition benchmark: a multi-class classification competition. In: Proceedings of the 2011 International Joint Conference on Neural Networks, 2011. 1453–1460

39 Xu P, Ehinger K A, Zhang Y. TurkerGaze: crowdsourcing saliency with webcam based eye tracking. 2015. ArXiv:1504.06755

40 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. ArXiv:1409.1556

41 Dinh T C, Tran N, Nguyen J. Personalized federated learning with Moreau envelopes. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 33: 21394–21405

42 Luo B, Xiao W L, Wang S Q, et al. Tackling system and statistical heterogeneity for federated learning with adaptive client sampling. In: Proceedings of IEEE INFOCOM 2022-IEEE Conference on Computer Communications, 2022. 1739–1748

43 Schelter S, Grafberger S, Dunning T. HedgeCut: maintaining randomised trees for low-latency machine unlearning. In: Proceedings of the 2021 International Conference on Management of Data, 2021. 1545–1557

44 Gupta V, Jung C, Neel S, et al. Adaptive machine unlearning. In: Proceedings of Advances in Neural Information Processing Systems 34 (NeurIPS 2021), 2021

45 McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of Artificial Intelligence and Statistics, 2017. 1273–1282

46 Chen M, Zhang Z K, Wang T H, et al. When machine unlearning jeopardizes privacy. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, 2021

47 van der Maaten L, Hinton G. Visualizing data using t-SNE. J Mach Learning Res, 2008, 9: 2579–2605