Learn to Forget: Machine Unlearning via Neuron Masking

Zhao Ma, Yang Liu, Ximeng Liu, Member, IEEE, Jian Liu, Jianfeng Ma, and Kui Ren, Fellow, IEEE

Abstract—Nowadays, machine learning models, especially neural networks, have become prevalent in many real-world applications. These models are trained based on a one-way trip from user data; as long as users contribute their data, there is no way to withdraw. To this end, machine unlearning becomes a popular research topic, which allows the model trainer to unlearn unexpected data from a trained machine learning model. In this article, we propose the first uniform metric called forgetting rate to measure the effectiveness of a machine unlearning method. It is based on the concept of membership inference and describes the transformation rate of the eliminated data from "memorized" to "unknown" after conducting unlearning. We also propose a novel unlearning method called Forsaken. It is superior to previous work in either utility or efficiency (when achieving the same forgetting rate). We benchmark Forsaken with eight standard datasets to evaluate its performance. The experimental results show that it can achieve more than 90% forgetting rate on average and only causeless than 5% accuracy loss.

Index Terms—Machine unlearning, neuron masking, neural network

1 INTRODUCTION

NOWADAYS, machine learning models, especially neural networks, become prevalent in many real-world applications including medical diagnosis, credit-risk assessment, autopilot, and so on. Recent works, like federated learning [1], [2] and cryptography-based machine learning [3], [4], enable the training process to be done in a way without seeing the training data. However, these training data can contain some unexpected data like out-of-distribution data [5] and poison data [6]. Current training methods leave no way for the model trainer to withdraw these unexpected data.

To this end, machine unlearning becomes a popular research topic. It allows users to eliminate the memorization of their private data from a trained machine learning model. Existing unlearning methods can be roughly classified into two categories: exact unlearning [7], [8] or approximate unlearning [9], [10]. As its name implies, exact unlearning indicates an entity completely removes the impact of specific data points on the model by retraining from scratch. In contrast, approximate unlearning result in a model close to the model obtained by exact unlearning while needing less computation resources.

In more detail, retraining-based exact unlearning retrains the model after removing the samples to be forgotten [11]. Given that retraining from scratch incurs an overhead that is usually unaffordable (it may take several days to retrain a model), Bourtoule et al. [7] propose SISA, which divides the training set into slices and trains a model via incrementally learning, s.t. each intermediate model (after one slice is added) is recorded. To forget a sample, retraining starts from the first intermediate model that contains the contribution of that sample. However, this method is simply trading storage (for recording the intermediate models) for retraining time, instead of truly reducing the overhead of retraining.

Our Contributions. In this paper, we propose a novel unlearning method called Forsaken. Compared with retraining-based unlearning, Forsaken has a much more efficient unlearning phase in both storage and time usage.

The core idea of Forsaken is neuron masking. We introduce a mask gradient generator that can iteratively generate mask gradients to “stimulate” neural neurons to unlearn the memorization of given samples. Based on the state of the updated model, the mask gradient generator adjusts the magnitude of the mask gradients to avoid over-unlearning. Especially, instead of removing a specific training instance completely, Forsaken focuses more on the original intention of machine unlearning. That is, the model trainer usually wants to conduct machine unlearning when some unexpected training data can cause negative consequences, e.g., privacy leakage caused by out-of-distribution (OOD) data [5] or model utility degradation caused by poison data [6]. From this perspective, Forsaken is a more straightforward but effective method than prior works to achieve the goal of machine unlearning. Also, as being freed from full-retraining or storing intermediate models, Forsaken is a more lightweight method to be applied in practice.
Furthermore, we propose a uniform metric called forgetting rate to evaluate the effectiveness of an unlearning method. It is based on membership inference [12], [13], which is to infer whether a given sample is a member of the training set. Roughly speaking, the forgetting rate describes the transformation rate of the unlearned data from “memorized” to “unknown” after conducting machine unlearning. The indicator can be directly used for machine unlearning evaluation with membership inference algorithms [12], [14].

The contributions of this paper are summarized below.

- We propose a uniform metric called forgetting rate that can measure the effectiveness of a machine unlearning method.
- We propose a novel machine unlearning method called Forsaken. It is superior to previous work in either utility or efficiency for removing different kinds of unexpected data.
- We benchmark Forsaken with eight standard datasets to evaluate its performance. The experimental results show that it can achieve more than 90% forgetting rate on average and only cause less than 5% accuracy loss.

2 BACKGROUND

This section briefly introduces the necessary technical background for understanding this paper.

2.1 Machine Unlearning

For machine unlearning, prior work mainly explores two classic ways, summation-based machine unlearning [15] and retraining-based machine unlearning [7], [8]. The idea of the former one is from statistical query learning [16]. Denote the update of training samples at the $i_{th}$ iteration as $G_i = \sum_{x \in D} g_i(x)$, where $g_i(x)$ is the transformation of a specific sample $x$, $D$ is the training set. According to summation-based machine unlearning, a model trained for $k$ iterations can be expressed as: $f_k = \text{Learn}(G_1, G_2, \ldots, G_k)$. When there are samples $D'$ required to be unlearned, summation-based machine unlearning computes the equation below:

$$f'_k = \text{Learn}(G_1 - G'_1, G_2 - G'_2, \ldots, G_k - G'_k),$$

where $f'_k$ is the updated model whose memorization about $D'$ is forgotten. Summation-based machine unlearning works well for non-adaptive machine learning models. As for adaptive neural networks, a slight parameter change at one iteration can make all the subsequent training to be deviated, which can further cause a considerable decrease in model performance.

The latter introduces the ensemble learning technique to implement machine unlearning. Specifically, retraining-based machine unlearning divides the whole training set into multiple shards, and each shard is further partitioned into multiple slices, which are used to train a constituent model slice by slice. During the training process, every intermediate model updated after one slice is added is recorded. When machine unlearning is invoked, retraining is applied to the first intermediate model trained without the slice containing the forgotten data. Although such a method is more efficient than native full retraining, its ensemble learning mechanism still needs massive computation and storage resources. Also, due to the partition of the training set, the performance of constituent models cannot be guaranteed.

Inspired by the above illustration, a formal definition to describe machine unlearning is given as follows.

**Definition 1 (Machine Unlearning).** Given a model $f$, an ideal membership inference $\psi$, a training set $D$ and an unlearning set $D' \subset D$, machine unlearning is perfectly performed if there exists a function $\phi$ that can output $f' = \phi(f)$ where for each $x \in D'$, we have $\psi(f(x)) = \text{true}$ and $\psi(f'(x)) = \text{false}$, and meanwhile, for each $x \in D/D'$, $f'(x) = f(x)$.

Moreover, given a learning task, the training data can always be divided into two types of data, namely OOD data and ID data. Correspondingly, machine unlearning also involves OOD data unlearning and ID data unlearning. Here, we define these two types of data in Definition 2 to make it easier to understand our work. Especially, a wide range of methods have been proposed to identify them in applications, e.g., uncertainty estimation [18], [19].

**Definition 2 (OOD and ID Data[20]).** Given a learning task, assume the desired distribution of the task to be $\chi$. For the samples $x \sim \chi$, we say that $x$ is in-distribution (ID), otherwise, it is out-of-distribution (OOD).

2.2 Membership Inference

A membership inference attack answers queries of whether a certain sample is in the training set of a machine learning model. It can be implemented based on one or multiple shadow models [12], [13], [21]. More specifically, an attacker collects a shadow dataset that is in the same distribution as the training set of the target model; and trains a shadow model based on the shadow dataset; then, the attacker feeds some members and non-members of the shadow model’s training set to the shadow model and gets a set of confidence vectors. Based on these confidence vectors, the attacker trains a binary classifier that can decide whether a confidence vector output by a model is from an input that belongs to the model’s training set. Then, given a sample’s confidence vector output by the target model, the binary classifier can be used to infer whether this sample is a member of the target model’s training set.
In this paper, we utilize membership inference as a tool to implement membership oracle to evaluate the effectiveness of a machine unlearning method. Moreover, existing membership inferences can only be used in classification tasks. Therefore, the machine learning models mentioned in this paper are all classifiers by default.

### 3 Forgetting Rate

In this section, we propose the first uniform metric called forgetting rate that can measure the effectiveness of a machine unlearning method. Given a set of samples $D$ to be forgotten, the forgetting rate (FR) of a machine unlearning method is defined as follows:

$$FR = \frac{AF - BF}{BT}$$  \hspace{1cm} (2)

- $AF$: the number of samples in $D$ that are predicted to be false by a membership oracle after machine unlearning.
- $BF$: the number of samples in $D$ that are predicted to be false by a membership oracle before machine unlearning.
- $BT$: the number of samples in $D$ that are predicted to be true by a membership oracle before machine unlearning.

In this way, FR gives an intuitive measurement of the rate of samples that are changed from member to non-member after unlearning. An effective unlearning method should at least achieve $AF > BF$ on the condition that $BT > 0$, otherwise, unlearning is reversed or meaningless. When $FR = 100\%$, $AF = BT + BF = |D|$, which means that all memorized samples have been successfully unlearned. On the other hand, when $FR = 0\%$, $AF = BF$, which means no memorized samples have been unlearned. We further remark that the membership oracle can be any membership inference algorithm. Note that to achieve a higher recall rate, current membership inference methods usually suffer from a slightly higher false positive rate, but a lower false negative rate (FNR). Thus, FR can be slightly lower than reality sometimes. However, in common unlearning scenarios, both BT and BF can be calculated based on the same target model (or directly derived from training settings). Hence, we argue that the membership inference based FR is still an effective metric to compare the effects of different unlearning methods.

### 4 Unlearning With Forsaken

In this section, we formally introduce Forsaken, a novel machine unlearning method that is superior to previous work in either utility or efficiency. We first give an intuition for designing Forsaken and then provide more details.

#### 4.1 Intuition

The core idea of Forsaken is masking the neurons of the target model with gradients (called mask gradients) that are trained to eliminate the memorization of some training samples from the target model. This idea was inspired by the theory of “active forgetting” [22] in Neurology. Different from the hysteresis of natural forgetting (a.k.a “passive forgetting”), active forgetting is vigorous and it can eliminate all traces and engram cells for a given memory. To achieve active forgetting, the forgetting cells produce a kind of special dopamine, which serves as the forgetting signal that stimulates remodeling of the engram cells and accelerates memorization elimination. Based on the feedback of the engram cells, the forgetting cells can adjust dopamine production to avoid unexpected forgetting.

In Forsaken, a mask gradient generator $G$ plays the role of forgetting cells. To eliminate the memorization of some training samples, Forsaken continuously invokes $G$ to produce mask gradients, which is similar to dopamine. The mask gradients are then applied to the neurons of the neural network and stimulate them to eliminate the memorization of the given samples. Based on the state of the updated model, $G$ adjusts the magnitude of the mask gradients to avoid catastrophic forgetting. Fig. 1 visualizes this process.

**Improvements.** Compared with the existing methods, the mask gradient based Forsaken mainly achieves the following improvements:

1) **Forsaken** can be applied to all kinds of the existing neural networks, no matter deep or shallow architecture, and meanwhile, does not introduce obvious accuracy loss on the original model.

2) **Forsaken** does not break the standard model training procedure (avoid retraining and uploading data in a predefined order) or change the original model architecture.

#### 4.2 Forsaken in Detail

Algorithm 1 shows the pseudocode for Forsaken and Table 1 lists the notations. At the $t$-th iteration, we first generate the mask gradients $\mu_i$ with the generator $G$ (line 3), and then, evaluate the unlearning effect as the mask gradient is $\mu_i$ (line 4). Next, we feed each sample to be forgotten into the updated model $f_\theta$ and obtain the prediction results:

$$T = \{y_i | y_i \leftarrow f_\theta(x_i), x_i \in D\}$$  \hspace{1cm} (line 5). For a given task to
classify any input into one of $p$ classes, the output $y_i \in \mathcal{Y}$ is limited to be a confidence vector with fixed $p$ dimensions.

In the end of the $t$-th iteration, we optimize $\mathcal{G}$ with $\mathcal{Y}$ by minimizing $L_{\text{forgetting}}$, which is the distance between $\mathcal{Y}$ (i.e., the posteriors of $D$) and $\mathcal{P}$ (i.e., the posterior distribution of non-member data) (line 6). During the optimization procedure, the original parameters of the target model are fixed. There are two challenges for optimizing $\mathcal{G}$: 1) defining its loss function $L_{\text{forgetting}}$; 2) determining a proper posterior distribution $\mathcal{P}$ for non-member data. Next, we explain how we address these two challenges in greater detail.

**Algorithm 1. Forsaken in a Nutshell**

**Input:** A target model $f_\theta$, and its trainable parameters $\theta$; A set of samples $D$ to be forgotten; $T$ is the maximum number of training iterations.

**Output:** A final model $f_{\theta_T}$.

1: Initialize the mask gradient $\mu_0$.
2: for $t \leftarrow 1$ to $T$ do
3:     $\mu_t \leftarrow \mathcal{G}(\theta_{t-1}, \mathcal{Y})$.
4:     $\theta_t \leftarrow \theta_{t-1} - \xi \cdot \mu_t$.
5:     $\mathcal{Y} = \{y_i | y_i \leftarrow f_\theta(x_i), \forall x_i \in D\}$.
6:     Optimize $\mathcal{G}$ by minimizing $L_{\text{forgetting}}$.
7: end for
8: Delete $\mathcal{G}$, and $D$ from the original dataset.
9: Return $f_{\theta_T}$.

**4.2.1 Loss Function of $\mathcal{G}$**

The goal of the mask gradient generator $\mathcal{G}$, defined in Eq. (3), is to find a particular perturbation over $\theta$ to minimize the distance between $\mathcal{Y}$ (i.e., the posteriors of $D$) and $\mathcal{P}$ (i.e., the posterior distribution of non-member data). The function of the perturbation is opposite to the descent gradient in model training, and more likely to be a kind of ascent gradients [23]

$$L(\mathcal{Y}, \mathcal{P}).$$

(3)

To resolve this optimization problem, we need to first quantify the distance between member and non-member posteriors. We borrow experience from another trendy topic of machine learning security, adversarial example (AE) [24]. To generate an AE, one needs to estimate the distance between the posteriors of an AE and a specific legal output for a given target model. A common distance function used in AE generation is Kullback-Leibler divergence (KL-divergence) [24]. We adopt this function in $\mathcal{G}$.

We also need to avoid “over-unlearning”, which could potentially lower the accuracy of the target model. To this end, we adopt a penalty item, $L_1$ norm regularization item [25], which is also commonly used in normal neural network training to avoid overtraining. The reason for choosing $L_1$ norm, not $L_2$ norm, $p \geq 2$, is that $L_1$ norm is smoother and the penalty of $L_1$ norm can block the convergence of Forsaken.

Put it altogether, we can rewrite Eq. (3) as follows:

$$L_{\text{KL}}(\mathcal{Y}, \mathcal{P}) + \lambda \cdot ||\mu||_1,$$

(4)

where $L_{\text{KL}}$ represents KL divergence loss function, $\lambda$ is the penalty coefficient, $||\mu||_1$ is the $L_1$ norm regularization item and $\mu$ is the mask gradient. We say that the above optimization problem can always work. If the KL-divergence of $\mathcal{P}$ and the posteriors of $D$ are already close (i.e., have a small KL-divergence), that means the target model does not remember $D$ in the very beginning. Then, there is no need of unlearning $D$. Otherwise, any optimizer can be used to solve the above problem. Moreover, we do not use the common first-order gradient optimizer as $\mathcal{G}$, like SGD and adaptive moment estimation (Adam) [26], but the second-order gradient optimizer, L-BFGS [27], which is slower than the first-order optimizer but converge more steadily.

Further, in our evaluation, we find that the above loss function still suffers from a dilemma of balancing the forgetting rate and accuracy drops. In particular, the traditional penalty item treats all parameter changes equally. Sometimes, such a penalty mechanism blocks some updates that can cause dramatic performance drops, while over-blocking some other changes that are useful to “forgetting” but hardly affect model performance. To resolve the problem, we propose a dynamic penalty mechanism as follows:

$$L_{\text{KL}}(\mathcal{Y}, \mathcal{P}) + \lambda \cdot \omega \cdot ||\mu||_1,$$

(5)

where $\omega$ is designed to maintain a subtle bound of penalty over parameter changes. When $\omega = 1$, Eqs. (5) and (4) are identical. $\omega$ can be computed based on Eq. (6)

$$\omega = \frac{1}{|D_0|} \sum_{x \in D_0} |\frac{\partial L_{\text{cross}}(x)}{\partial \theta(d)}|,$$

(6)

where $\theta(d)$ specifies the $d_{\text{th}}$ dimension of parameters and $D_0$ is a very small set of training data (less than 1%) that should not be forgotten. It can be observed that $\omega$ increases positively with the partial gradients of $D_0$. Note that the partial gradients potentially indicate the performance degradation level of the target model on normal training data. Therefore, if the change of $\theta(d)$ results in higher performance degradation, we assign more penalties to mitigate such catastrophic forgetting, and vice versa. In Section 5.3, we implement both Eqs. (4) and (5) to justify the improvement of above modification.

**4.2.2 Non-Member Data**

Recall that the goal of the mask gradient generator $\mathcal{G}$ is to minimize the distance between $\mathcal{Y}$ (i.e., the posteriors of $D$) and $\mathcal{P}$ (i.e., the posterior distribution of non-member data). $\mathcal{Y}$ can be obtained from the prediction results after feeding the samples to the model. We still need to estimate $\mathcal{P}$.

Roughly, the training data can be classified into two categories, OOD data and ID data. Here, we first discuss how to derive the desired posterior distribution of non-member OOD data, and then, expand it to ID data.

The intuition for estimating non-member OOD data is that the model no longer strongly predicts a non-member OOD sample to be in any specific category. Then, based on Shannon entropy theory [28], we define an ideal non-member OOD posterior distribution as follows.

**Definition 3 (Ideal Non-Member OOD Posterior).** Based on Shannon's entropy theory [29], for a given neural network $f_\theta$ with $p$-dimensions output, we say that an OOD sample $x$ is
ideally non-member when it reaches maximum entropy, i.e., its posterior $y_1, \ldots, y_p) = f_0(x)$ satisfying $y_1 = y_2 = \ldots = y_p = \frac{1}{p}$.

However, in real-world scenarios, such ideal posterior distribution is rarely seen. Even for the randomly generated meaningless data that has never occurred in the model training process, their posteriors are always biased towards some specific dimensions due to the generality of neural networks [30]. Therefore, instead of using the ideally uniform posterior distribution, we sample the posterior distribution of some real-world OOD data to serve as $\mathcal{P}$, defined in Definition 4. Since the target distribution tends to be similar to the normal real-world non-member OOD samples, the difficulty of the adversary to identify the forgotten data is increased, and the catastrophic forgetting can be mitigated.

**Definition 4 (Real-World Non-Member OOD).** For a neural network $f_0$ with $p$-dimensions output, we say that an OOD sample $x$ is real-world non-member if its posteriors $(y_1, y_2, \ldots, y_p) = f_0(x)$ tends to be $(z_1, z_2, \ldots, z_p) = f_0(X_n)$, where $X_n$ is a set of non-member OOD samples and $z_i$ is the posterior distribution of $X_n$ over dimension $i$.

In more detail, the sampling of real-world non-member OOD distribution can be implemented by leveraging some publicly available data that are irrelevant to the learning task. For example, assume the learning task is 0-9 digital number recognition. We can first collect digital images about a-z characters to serve as the non-member OOD data, and label them by querying the target model. Then, the posteriors of the non-member OOD data with the same class are averaged to form $\mathcal{P}$. Inspired by the above discussion, we can simply expand Forsaken to the ID data unlearning scenario by replacing the non-member OOD data with testing data.

**Definition 5 (Real-World Non-Member ID).** For a neural network $f_0$ with $p$-dimensions output, we say that an ID sample $x$ is real-world non-member if its posteriors $(y_1, y_2, \ldots, y_p) = f_0(x)$ tends to be $(z_1, z_2, \ldots, z_p) = f_0(X_n)$, where $X_n$ is a set of non-member ID samples (testing samples) and $z_i$ is the posterior distribution of $X_n$ over dimension $i$.

Furthermore, compared to OOD data, ID data are more highly related to the learning task, and thus, the non-member ID posterior is usually closer to its corresponding member posterior. The fact always leads to more accuracy reduction and slower convergence of ID data unlearning than OOD data unlearning. In Section 5, comprehensive experiments are conducted to validate such a statement.

### 4.3 Implementation of Forsaken

Given the design details of Forsaken, we present Algorithm 2 to explain how to deploy in applications.

Like prior works [7], we assume the unlearning process is invoked before the target model has been deployed on applications. At this time, if the data provider, or the model trainer itself, finds some unexpected data $D_0$ needs to be unlearned, the following procedure is conducted. First, the model trainer randomly selects a small set of data uninvolved in the training set. Based on these selected samples, the non-member posterior distribution $\mathcal{P}$ can be determined based on Definition 4. Denote the target model to be $f_0$.

Then, the model trainer can invoke Forsaken to compute mask gradients for $f_0$ with the unlearning set $D_0$ and $\mathcal{P}$. Note that during the process, the optimization object is the gradients but not the original model parameters $\theta$. Thus, after the iteration, if the unlearning effect is evaluated to be not satisfying, the current iteration can be abandoned directly. Finally, as proper mask gradients are obtained, they are accumulated to the target model to complete the unlearning process.

**Algorithm 2. Machine Unlearning With Forsaken**

**Input:** The unlearning set $D_0$ with size $n_0$; the current neural network $f_0$; the learning rate $\eta$; the loss function $\mathcal{L}(\cdot)$; the maximum iteration number $T$.

**Output:** The updated model $f_D$.

1. The model trainer does the follow steps.
2. Sample the posterior distribution $\mathcal{P}$.
3. Continue a new iteration process to compute the mask gradients $u = \text{Forsaken}(f_0, \theta, D_0, T, \mathcal{P})$.
4. Accumulate the mask gradients to the current model.
5. Return the updated model $f_D$.

### 5 EVALUATION

In this section, comprehensive experiments are conducted to evaluate the effectiveness and efficiency of Forsaken.

#### 5.1 Experimental Setup

**Dataset.** We mainly implement Forsaken on eight standard datasets, namely CIFAR10, CIFAR-100, IMDB, News, Reuters, STL-10, TinyImage, Customer. The datasets cover three different types of learning tasks, including image classification, sentiment analysis, and text categorization. The unlearning ID data are randomly selected from the training set. Besides, we randomly select 1000 samples from the testing set to compute the desired non-member ID distribution defined in Definition 5. We leverage three common ways [31] to simulate OOD data.

1. C10.S. The OOD data are from the same learning task but mislabeled by users, e.g., CIFAR10 and STL-10. We randomly select parts of data from STL-10 and mislabel them with another label to serve as OOD data.
2. C10.T., C100.T. and I.C. The OOD data are from completely different learning tasks. For example, we can collect skirt images from TinyImages, label them as birds and then insert them into CIFAR10, which does not contain skirt images, to serve as OOD data.
3. News (15-5) and Reuters (35-16). In this case, the training set is split into two parts based on labels. One is used for target model training. The other is used as the OOD set. For instance, we can select samples from News whose labels are between 0-14 as the target set. The others are deployed as OOD data.

Note that all unlearning OOD data are randomly selected with a mix of labels in our evaluation. The default unlearning size is 200. Moreover, 1000 samples are randomly chosen from the remaining unselected OOD samples to
Table 2 summarizes the usage of the datasets in the experiments.

Membership Inference. To compute FR, we train a black-box membership inference for each dataset according to the shadow model based method [13]. Concretely, we randomly partition each dataset into two halves, which are used to train the target and the shadow model, respectively. Two models have the same architecture and hyperparameters. Then, we train an XGBoost [32] model (the best performance model in our experiments) to serve as the membership inference. In Section 5.5, we further discuss the performance of Forsaken with the white-box inference oracle.

Neural Networks. Four different neural networks are involved in processing the target datasets. The detailed neural network architectures for IMDB, Reuters and News are given in Figs. 2, 3 and 4, respectively. For CIFAR-10 and CIFAR100, we use a previously proposed deep network architecture VGG-16 [33]. Figs. 2, 3 and 4 show the neural network architectures used to process IMDB, Reuters and News, respectively. The detailed setting of the model training is shown in Table 3.

Other Setting. The experiments are simulated on a workstation equipped with Tesla V100 GPU and 16GB VRAM. The experiment results are averaged over ten trials and processed in a single thread.

5.2 Performance Comparison

To evaluate the performance of machine unlearning, three indicators should be considered: 1) the number of samples that are memorized during training, i.e., BT in Eq. (2); 2) the rate of samples that are successfully unlearned, i.e., FR; 3) the performance drops compared with the original model. Indicator 1 suggests whether the memorization of unlearning data is formed. Indicator 2 evaluates the effectiveness of machine unlearning algorithms. Indicator 3 indicates the side-effect of machine unlearning.

Tables 4 and 5 summarize the former two indicators and compare Forsaken with other four machine unlearning methods on both OOD and ID data unlearning. Here, native full retraining without the unlearning data serves as the basic baseline. To implement summation-based machine unlearning [15] (MU), we record all gradients of the unlearning data in each epoch, and then, conduct one more epoch of training to subtract them from the target model. As for the retraining-based machine unlearning proposed in [7] (SISA), we set the number of shards to be 10, a very low level of partition in applications (ten users in federated learning). Each shard is then divided into 100 slices, which are used to train a constituent model slice by slice. The output of SISA is decided by majority voting. The posterior of SISA is the mean value of the posteriors for the constituent models that are winners in the voting process. The hyperparameters of each constituent model in SISA are the same as Forsaken. For fairness, the membership inference used

---

**Table 2**

| Abbreviation | Target Dataset | OOD Dataset |
|--------------|----------------|-------------|
| C10.S.       | CIFAR10        | STL-10      |
| C10.T.       | CIFAR10        | TinyImage   |
| C100.T.      | CIFAR100       | TinyImage   |
| LC.          | IMDB           | Customer Review |
| Reuters (35-11) | Reuters (35)  | Reuters (11) |
| News (15-5)  | News (15)      | News (5)    |

1) Embedding: Input word $1 \times 1000$, the output word embedding is $1 \times 100$.
2) Convolution: Windows size $3 \times 100$, number of output channel 100.
3) Convolution: Windows size $4 \times 100$, number of output channel 100.
4) Convolution: Windows size $5 \times 100$, number of output channel 100.
5) MaxPooling: Window size $1 \times 35 \times 35$.
6) Fully Connected Layer: Fully connected the incoming 300 nodes to the outgoing 2 nodes.

**Table 3**

| Name          | No. of Instances | Features | Classes | Epochs |
|---------------|------------------|----------|---------|--------|
| CIFAR-10      | 60000 (10k for testing) | 32 x 32  | 10      | 20     |
| CIFAR-100     | 60000 (10k for testing) | 32 x 32  | 100     | 15     |
| IMDB          | 50000 (25k for testing) | 200      | 2       | 30     |
| Reuters       | 11228 (2246 for testing) | 10000    | 46      | 40     |
| News          | 11314 (2262 for testing) | 1000     | 20      | 40     |
| STL10         | 13000            | 32 x 32  | 10      | OOD    |
| TinyImage     | 100000           | 32 x 32  | 200     | OOD    |
| Customer      | 2775             | 200      | 2       | OOD    |
for SISA is the same as \textit{Forsaken} and MU, not trained by the ensemble of multiple shadow constituent models. Such a configuration can also show that roughly applying ensemble learning to machine unlearning may cause unexpected performance loss, such as weaker memorization on training data, lower accuracy and more training time. SISA-DP [8] is implemented similar to SISA. The only difference is that SISA-DP introduces more tricks to ensure model performance and includes a DP based model publishing function. The differential noise budget $\epsilon$ is set to be 16 as suggested by [34], which is often used in applications and can maintain a good balance between accuracy drops and data privacy.

From the results, the first observation is that although OOD data are irrelevant to the original learning task, most of them are still memorized by the target model. Besides, most of the involved OOD data are correctly classified into the correct classes and do not obviously affect the performance of the target model (shown in Table 6). The experiments prove that unintended memorization caused by OOD data is a common phenomenon in real applications. Further, we present more detailed experiment analysis as follows.

### TABLE 6

| Dataset | Acc with OOD Data | Acc without OOD data |
|---------|------------------|----------------------|
|         | Acc.Train | Acc.Test | Acc.Train | Acc.Test | C.B.OOD |
| CIFAR10 | 96.15%    | 72.72%    | 99.99%    | 77.98%    | 120     |
| CIFAR100| 99.99%    | 76.92%    | 99.99%    | 77.98%    | 181     |
| IMDB    | 90.71%    | 85.34%    | 91.68%    | 85.7%     | 185     |
| Reuters | 90.62%    | 77.89%    | 91.96%    | 79.91%    | 146     |
| News    | 99.93%    | 61.57%    | 99.99%    | 64.93%    | 154     |

### TABLE 4

Forgetting Rate of \textit{Forsaken} on Different Datasets for OOD Data Unlearning

| Dataset | Parameter Size | BT (BF) | Forgetting Rate (Average / Variance) |
|---------|----------------|---------|-------------------------------------|
|         |                |         | Full Retraining | Forsaken | MU [15] | SISA [7] | SISA-DP [8] |
| C10.S.  | 14.77M         | 168 (32)| 93.45 / 0.34%    | 97.62 / 0.44%    | 86.91 / 4.89%    | 91.67 / 0.79%    | 94.05 / 1.21% |
| C10.T.  | 14.77M         | 176 (24)| 94.89 / 0.28%    | 98.29 / 0.69%    | 93.75 / 5.94%    | 96.03 / 0.63%    | 97.16 / 0.73% |
| C100.T. | 14.77M         | 117 (83)| 96.58 / 0.41%    | 95.73 / 1.32%    | 71.79 / 4.13%    | 86.32 / 2.89%    | 89.74 / 1.59% |
| I.C.    | 2.62M          | 195 (5) | 94.36 / 0.16%    | 98.46 / 0.79%    | 43.59 / 6.33%    | 95.89 / 0.41%    | 96.41 / 1.14% |
| Reuters | 5.26M          | 192 (6) | 94.79 / 0.13%    | 96.35 / 0.82%    | 81.25 / 2.51%    | 95.32 / 0.76%    | 93.75 / 0.87% |
| News (15-5) | 0.25M  | 162 (38)| 95.06 / 0.23%    | 97.53 / 0.59%    | 86.93 / 4.47%    | 92.26 / 1.09%    | 94.44 / 1.81% |

Forgetting Rate. For both OOD and ID data, \textit{Forsaken} generally achieves close to, or even higher, $FR$ than full retraining, and the size of the training set or the mask gradient does not strongly affect the performance of \textit{Forsaken}. Relatively, MU has the worst performance due to its non-directional forcible unlearning mechanism. Intuitively, without consideration of practicality, the most effective method to implement machine unlearning is full retraining. However, full retraining (including SISA) fails to achieve 100\% $FR$ in our experiments. Two reasons lead to the phenomenon. First, the generality of neural networks makes the retrained model be still capable of identifying some unlearning data with high confidence, especially in the ID data scenario. Second, for SISA, its design changes the architecture of the original neural network thereby lowering the learning ability of the target model. Taking the original network as the baseline, the model trained with SISA fails to memorize the unlearned data from the training stage. Thus, SISA suffers from a lower FR (referring to Eq. (2)). As for SISA-DP, due to the differential noises, it achieves similar $FR$ to SISA but suffers from more accuracy drops. Here, another interesting discovery is that on the model protected with a low level of noise, membership inference attains similar performance to the non-defensive model, which corresponds to the statement of [34]. Moreover, compared to OOD data unlearning, ID data unlearning achieves lower $FR$. This is because the high relevance of ID data to the learning task makes ID data harder to unlearn and easier to cause accuracy drops.

### TABLE 5

Forgetting Rate of \textit{Forsaken} on Different Datasets for ID Data Unlearning

| Dataset | Parameter Size | BT (BF) | Forgetting Rate (Average / Variance) |
|---------|----------------|---------|-------------------------------------|
|         |                |         | Full Retraining | Forsaken | MU [15] | SISA [7] | SISA-DP [8] |
| CIFAR10 | 14.77M         | 167 (33)| 85.32 / 0.09%    | 88.63 / 2.29%    | 85.03 / 5.77%    | 86.23 / 0.41%    | 88.02 / 0.43% |
| CIFAR100| 14.77M         | 191 (9) | 94.24 / 0.04%    | 87.96 / 1.63%    | 79.06 / 5.05%    | 84.29 / 1.52%    | 88.48 / 1.46% |
| IMDB    | 2.62M          | 151 (49)| 84.77 / 0.11%    | 94.04 / 2.19%    | 62.25 / 6.13%    | 87.42 / 0.71%    | 90.72 / 0.81% |
| Reuters | 5.26M          | 160 (40)| 83.13 / 0.09%    | 86.25 / 1.29%    | 69.38 / 5.33%    | 81.25 / 0.29%    | 84.38 / 0.76% |
| News    | 0.25M          | 172 (28)| 85.47 / 0.71%    | 90.12 / 1.84%    | 86.86 / 6.67%    | 84.31 / 0.37%    | 87.21 / 0.84% |

Performance Drops. Then, we focus on the model performance change of the target model before and after conducting OOD and ID data unlearning, shown in Figs. 5 and 6, respectively. From the results, the model performances after conducting SISA and SISA-DP are much lower than the original normal model, which reflects its weaker memorization over the training set caused by data partition.
Conversely, except full retraining, the performance drop of Forsaken is the lowest due to its subtle remodeling of the target model. Moreover, as mentioned before, the performance drop caused by ID data unlearning is higher than OOD data for Forsaken.

Efficiency. Except for performance drops, efficiency also counts a lot for applying machine unlearning. Since the computation complexities of OOD and ID data unlearning are identical, we only show the running time comparison result of OOD data unlearning in Table 7. It can be observed that the running time of SISA and SISA-DP is similar to full retraining, because SISA degrades to the performance of retraining the whole model as the percentage of unlearning data is more than 0.075% (detailedly discussed in [7]), and has to retrain all 10 shards. Compared with the retraining based methods, Forsaken can save hundreds of times overheads. Even for MU, which only needs one epoch of training, its running time is several times more than Forsaken because the involved training data of Forsaken is less than MU. To further understand the efficiency comparison results, we theoretically analyze the time complexity as follows.

Take the sizes of the training set and unlearning set as $N$ and $n_0$, respectively. Suppose the time required to update the target model with one sample is $O(1)$. The time complexity of full retraining is $O(KN)$ where $K$ is the number of training epochs. The time complexity of Forsaken is $O(tn_0)$, where $t$ is the training iterations of mask gradient generator. The time complexity of both SISA and SISA-DP is $O(K(N - n_0))$. MU only needs one normal epoch of training used to subtract recorded gradients. Thus, the time complexity of MU is $O(N - n_0)$. Due to $n_0 \ll N$, the running time of Forsaken is less than MU.

Comparison With Different Unlearning Sizes. Overall, Forsaken attains the best performance among all baselines. To further evaluate the stability of Forsaken, we compare it with other methods with varying unlearning sizes (100 to 300) over two types of datasets, C10.T and IMDB. Fig. 7 plots the experiment results for OOD data unlearning. Test. Acc means the testing accuracy.

### Table 7

| Dataset      | Running Time (s) | Retraining | Forsaken | MU         | SISA       | SISA-DP   |
|--------------|------------------|------------|----------|------------|------------|-----------|
| C10.S.       | 1087.6 / 2.78    | 15.09 / 0.18 |          | 45.7 / 0.21 | 965.59 / 1.73 | 971.61 / 1.63 |
| C10.T.       | 1089.4 / 2.56    | 16.12 / 0.07 |          | 47.05 / 0.06 | 960.25 / 1.85 | 972.31 / 2.01 |
| C100.T.      | 925.65 / 1.73    | 7.69 / 0.12  |          | 61.05 / 0.11 | 851.59 / 2.12 | 852.29 / 1.32 |
| I.C.         | 372.3 / 0.72     | 2.35 / 0.07  |          | 10.5 / 0.07  | 282.52 / 0.22 | 285.45 / 1.06 |
| Reuters (35-11) | 121.6 / 0.67    | 1.68 / 0.02  |          | 3.35 / 0.03  | 89.87 / 0.14  | 91.12 / 1.32  |
| News (15-5)  | 174 / 0.73       | 2.57 / 0.09  |          | 3.91 / 0.05  | 110.08 / 0.26 | 117.72 / 0.49  |

Fig. 5. Performance change of the target model over the testing set for OOD data unlearning.

Fig. 6. Performance change of the target model over the testing set for ID data unlearning.
Fig. 8. The increased unlearning data size only leads to a modest reduction of \( FR \) and accuracy for Forsaken. Besides, the performance of Forsaken is always close to full retraining and outperforms other methods with different unlearning sizes. The phenomenon validates that Forsaken can perform as stably as retraining-based methods.

**Elaborate Comparison With SISA.** As the retraining-based machine unlearning is the most competitive method among all baselines, Table 8 further elaborates comparison of Forsaken with SISA and SISA-DP. In more detail, we show the performance change of the three methods with different numbers of shards for OOD data unlearning. From the experimental results, SISA and SISA-DP attain similar \( FR \) to Forsaken but cause more accuracy drops as the number of shards increases from 1 to 4. The phenomenon illustrates that although the memorization of OOD data can be forgotten by SISA, the target model loses its utility due to the partition of datasets. Relatively, Forsaken does not suffer from the problem.

**5.3 Effect Factors**

In this part, we conduct extensive experiments to analyze the key factors that affect Forsaken performance. Three different types of datasets are mainly involved, including C10.T., I.C. and Reuters (35-11).

**Training Iteration of Generator.** We experiment with the performance change of Forsaken under different iteration rounds (shown in Fig. 9). Commonly, Forsaken reaches its best performance after tens of iterations. Once the best point is reached, more training iterations hardly affect \( FR \) but cause more performance degradation of the target model (mainly reflected in accuracy decreasing). The reason is that according to our design, overtraining of Forsaken can cause over-unlearning of unrelated memorization, which lowers the performance of the target model. Moreover, the efficiency of Forsaken is also strongly affected by the number of training iterations. Table 9 shows the running time of Forsaken to accomplish machine unlearning of 200 samples with different iteration numbers. The results show that the running time linearly increases along with the training iterations. However, even for the
neural network with 14.09M parameters, the mask gradient generation can complete 30 iterations in seconds, which is much faster than other retraining based methods.

Penalty. In Forsaken, another important factor is the penalty method used for alleviating catastrophic forgetting. Table 10 illustrates the performance of Forsaken under different penalty settings. It can be found that when the penalty is canceled ($\lambda = 0$), Forsaken achieves more than 97% FR but suffers from about 9% accuracy drop. As the weighted penalty mechanism is introduced, accuracy drop is reduced to less than 4% while maintaining the same level of FR. Moreover, a common phenomenon for the two penalty mechanisms is that both FR and accuracy drop are decreased with the increasing penalty coefficient $\lambda$. When $\lambda$ is increased to 100, the unlearning process will be blocked, however, the accuracy drop can be well controlled. Combined with experiments in Section 5.7, it is still enough to defend against the data leakage caused by unintended memorization even FR is as low as 80%. Thus, the choice of high penalty is recommended in applications if there is not a strict restriction on FR.

Overtraining. According to [5], [13], [35], overtraining is also a factor that tightly relates to machine learning memorization. Loosely speaking, overtraining brings deeper memorization of the trained model towards the training set. Fig. 10 shows a typical example of overtraining with C10.T. At the first few epochs, the testing loss drops rapidly until reaching the best point. After this point, the trend of testing loss is reversed, which means the model begins to be overtrained. To evaluate Forsaken versus overtraining, we record the machine unlearning results at different stages of model training, including before and after overtraining.

Table 10 Changing of Forsaken Performance With Different Penalty Strategies

| Dataset        | without penalty weight $\omega$ | with penalty weight $\omega$ |
|----------------|---------------------------------|------------------------------|
|                | $\lambda = 0$ | $\lambda = 0.1$ | $\lambda = 1.0$ | $\lambda = 10$ | $\lambda = 100$ |
| C10.T.         | FR 97.72% | 97.72% | 87.75% | 98.29% | 78.97% |
|                | Diff.Acc 8.54% | 4.96% | 3.5% | 3.34% | 1.13% |
| I.C.           | FR 98.46% | 94.43% | 88.72% | 98.46% | 89.74% |
|                | Diff.Acc 6.21% | 4.14% | 2.32% | 2.66% | 0.67% |
| Reuters(35-11) | FR 97.39% | 93.39% | 85.54% | 96.35% | 86.98% |
|                | Diff.Acc 7.85% | 5.59% | 2.64% | 4.61% | 1.19% |

Here, Diff.Acc specifies the testing accuracy drops.

Every time we measure the performance of Forsaken in the experiment, a new membership inference is trained (training shadow model with the same epochs as the target model). This is because, unlike the previous experiments, the target model successively changes in the overtraining experiment. It can be discovered that Forsaken performs a little worse when the degree of overtraining is weaker. The reason is that the deeper memorization of the overtrained model leaves more space for Forsaken to fine-tune the target model for machine unlearning.

5.4 Visualization

To intuitively evaluate the effectiveness of Forsaken, we visualize the change of data attribution before and after performing machine unlearning. The visualization is based on t-distributed Stochastic Neighbor Embedding (t-SNE), a feature dimension reduction method [36]. Here, we only focus on the analysis of the OOD data unlearning scenario because Forsaken achieves similar performance on both OOD and ID data unlearning. Two types of learning tasks are picked to conduct the visualization experiment, namely C10.T. and I.C. Figs. 11 and 12 demonstrate the visualization results.

Three kinds of data are involved in the experiments, namely 100 member samples from the training set, 100 non-member samples from the testing set and 100 unlearning samples from the OOD data set. Using t-SNE, we reflect the
The top three posteriors of all samples into 2-dimension data points [13]. In the graphs, the colors of these samples are based on their attributions, which means a sample from the testing set is marked as a green point, the training sample is the blue point and the eliminated sample is the brown point. As the dimension embedding of t-SNE is related to the random state and input values, the locations of the training and testing samples before and after machine unlearning are not fixed. From the results, most of the eliminated samples are close to the group of training samples before machine unlearning. After machine unlearning, the eliminated samples tend to move to the non-member testing group. In other words, Forsaken successfully makes the eliminated samples change from “memorized” to “unknown”. Such a phenomenon is in accordance with Definition 1, which proves the effectiveness of our methodology to achieve machine unlearning.

### 5.5 White-Box Membership Inference

Considering the hardness to get access to the whole training set in many application scenarios (e.g., federated learning), the above all experiments are conducted with shadow model based black-box membership inference (BMI). To evaluate our method more comprehensively, we also experiment with the white-box membership inference (WMI) as the evaluation tool. In these experiments, WMI oracles are trained according to the method proposed in [37].

From Table 11, it can be found that with WMI oracles, Forsaken achieves close performance to the results computed with BMI. To further validate such a statement, we conduct visualization analysis with WMI oracles, the result of which is given in Figs. 13 and 14. For WMI based visualization, we change the encoder (embedding) layer output size in [37] from 1 to 10 and apply the sigmoid function to map the embedding outputs to range $[0, 1]$. With t-SNE, the top-3 embedding outputs are further mapped to 2-dimension points for visualization. From the result, it can be observed that similar to BMI, most eliminated data points are mixed together with other member data points before machine unlearning. Then, after Forsaken is conducted, the eliminated data points are moved from the member area to the non-member area. The fact proves that our proposed indicator and Forsaken are feasible and robust with no matter BMI or WMI as the evaluation tool.

### 5.6 Poison Data

To further validate the robustness of Forsaken, we expand Forsaken to the poison data forgetting scenario proposed in [6]. Here, the goal of machine unlearning is to improve the target model’s performance by eliminating polluted memories. In the experiments, the target dataset is CIFAR10. The percentage of introduced poison data is 2%, 4%, 10%. The attack adopts the $5 \rightarrow 3$ setting in [6], which means parts of data that should be labeled as 5 are modified to be 3 during the model training process. Moreover, for OOD or ID data unlearning, non-member sample distribution $P$ is derived from non-member sample posteriors. For poison data unlearning, $P$ is formed by the posteriors of the original data without poisoned labels. In other words, the learning task of Forsaken is changed from eliminating useless memories to correcting polluted memories. From Fig. 15, it can be observed that the introduction of poison data does not significantly affect the performance of Forsaken.

---

**TABLE 11**

| Dataset     | Forgetting rate ($FR$) |
|-------------|------------------------|
|             | Black-box Oracle       | White-box Oracle      |
| C10.S       | 97.62%                 | 97.02%                |
| C10.T       | 98.29%                 | 98.29%                |
| C100.T      | 95.73%                 | 96.58%                |
| I.C.        | 98.46%                 | 97.44%                |
| Reuters (35-11) | 96.35%             | 96.88%                |
| News (15-5) | 97.53%                 | 96.29%                |

---

(a) Visualization of training samples, unknown samples and unlearned samples before machine unlearning for C10.T. (b) Visualization of training samples, unknown samples and unlearned samples after machine unlearning for C10.T.
data can cause about 1% accuracy drop and 8.4% recall rate drop compared to the original model (full retraining). MU, SISA and SISA-DP fail to achieve performance improvement because of their native drawbacks as mentioned before. Forsaken fails to improve the accuracy of the target model but successfully improves the recall rate. Since recall rate quantifies the number of correct positive predictions made out of all positive predictions, Forsaken indeed improves the performance of the target model over the polluted category. From the above, we justify the generality of Forsaken to process polluted data.

5.7 Unintended Memorization Exposure

For the generative sequence model, e.g., the LSTM [38] based language model, Carlini et al. [5] introduced an indicator to measure the degree of the unintended memorization caused by a given OOD but private sequence $s[r]$, expressed as exposure. As stated in [5], the risk of the private information to be extracted by the adversary is positively related to exposure. The phenomenon demonstrates the high data privacy risk caused by OOD data and becomes one of the key factors to make this paper focus more on OOD data unlearning. The following is the equation to compute exposure:

$$\text{exposure}_u(s[r]) = -\log_2 \int_0^{P_u(s|r)} \rho(x)dx,$$

where $P_u(s|r)$ is the log-perplexity of $s[r]$ under the machine learning model $f_2$ and $\rho(\cdot)$ is a standard skew-normal function [39] with mean $\mu$, standard derivation $\sigma^2$ and skew $\alpha$. Log-perplexity is a standard indicator to evaluate how “surprise” the language model is to see a given sequence, which can be computed based on Eq. (8)

$$P_u(s|r) = \frac{\int_{-\infty}^{\infty} \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx}{\int_{-\infty}^{\infty} \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \rho(x)dx},$$

Notably, exposure is a specifically defined indicator that can only be applied to the generative sequence model.

To evaluate the effectiveness of Forsaken to lower the risk of data privacy leakage, we use the Penn TreeBank (PTB) dataset [40] to train a language model with a two-layer LSTM. During training, we insert 100 canaries into the training set using the same method as [5]. The 100 canaries are treated as the OOD samples required to be unlearned. Similar to above-mentioned CNNs, Forsaken is utilized to generate mask gradients for the weights and biases of the LSTM in the unlearning process. Fig. 16 plots the experiment result, where we record the averaged exposure of the 100 canaries. It can be discovered that exposure decreases along with the increase of FR. According to [5], the success probability of the adversary to extract a given sequence is negligible when exposure is less than 10. Therefore, Forsaken can significantly lower the probability of the adversary to extract the user’s private information from the unintended memorization.

6 RELATED WORK

In this section, we briefly review the related works that inspire our design of Forsaken.

Machine Unlearning. Machine unlearning has been one of trendy topic about machine learning security [11], [41]. Cao et al. [15] extensively studied the possibility of deploying the data forgetting mechanism in some real-world machine learning systems. Similar to [11] and [41], the main research object of [15] was the machine learning models with a simple structure, e.g., Naive Bayes. Another strategy to achieve machine unlearning is retraining-based [42], [43], [44]. For retraining, the unavoidable problem is the high extra computation and storage cost. Moreover, some retraining methods like [7] modifies the normal training workflow of machine learning and changes the original model structure (due to the utilization of ensemble learning). Considering the evaluation to machine unlearning effect, Golatkar et al. [45] proposed mixed-linear forgetting via a first-order Taylor-series. However, such an indicator failed to provide quantitative and intuitive evaluation metric on the unlearning effect. Chen et al. [46] studied the member privacy leakage problem caused by machine unlearning and presented potential ways to mitigate the problem.

Membership Inference. The research of membership oracle is first inspired by the membership privacy problem while deploying machine learning as a service [12]. Membership oracle answers the question of whether a given sample is a member of the training set used to train the target model based on the model output or not. The principle of membership oracle inspires us a lot to define the evaluation indicator of machine unlearning for machine learning. The early method to train a membership oracle used multiple shadow models [12], which was complex and inefficient. Later,
Salem et al. [13] showed that the membership oracle trained with only one shadow model could still have a good performance for most machine learning models. Further work [47] studied why the implementation of a membership oracle is possible. To defend the attack launched by a membership oracle, Jia et al. [48] believed that the most effective way was adding differential noises to the model predictions.

7 Conclusion

In this paper, we emphatically discussed the machine unlearning problem, and inspired by the “active forgetting” mechanism of the human nervous system, proposed our “learn to forget” method, Forsaken, to achieve machine unlearning for neural networks. In the method, the user could stimulate the neurons of a machine learning model to unlearn specific memorization by training a dummy gradient generator. In particular, the mask gradient could be treated in the same way as the common procedure of machine learning, which was more practical and efficient than the prior works. For better evaluating the performance of machine unlearning, we also presented the first indicator, called forgetting rate, to measure the transformation rate of the eliminated data from “memorized” to “unknown”. Reconsidering the design of Forsaken or other machine unlearning methods, we failed to provide provable guarantee for hiding the indirect footprint of unlearned data points. With our work as the stepping stone, we hoped more future works could further dive into the machine unlearning field and provide more extensive options to protect both private user data security and data unlearning privacy.

References

[1] K. Bonawitz et al., “Practical secure aggregation for privacy-preserving machine learning,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2017, pp. 1175–1191.
[2] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, “Federated learning over wireless networks: Optimization model design and analysis,” in Proc. IEEE Conf. Comput. Commun., 2019, pp. 1387–1395.
[3] P. Mohassel and P. Rindal, “ABY3: A mixed protocol framework for machine learning,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2018, pp. 35–52.
[4] N. Agrawal, A. S. Shamsabadi, M. J. Kusner, and A. Gascón, “QUOTIENT: Two-party secure neural network training and prediction,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2019, pp. 1231–1247.
[5] N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song, “The secret sharer: Evaluating and testing unintended memorization in neural networks,” in Proc. 28th USENIX Secur. Symp., 2019, pp. 267–284.
[6] V. Tolpegin, S. Truek, M. E. Gursoy, and L. Liu, “Data poisoning attacks against federated learning systems,” in Proc. Euro. Symp. Res. Comput. Secur., 2020, pp. 480–501.
[7] L. Bourtoule et al., “Machine unlearning,” in Proc. IEEE Symp. Secur. Privacy, 2021, pp. 141–159.
[8] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, “Descent-to-delete: Gradient-based methods for machine unlearning,” in Proc. 32nd Int. Conf. Algorithmic Learn. Theory, 2021, pp. 931–962.
[9] A. Thudi, H. Jia, S. Shumailov, and N. Papernot, “On the necessity of auditable algorithmic definitions for machine unlearning,” 2021, arXiv:2110.11891.
[10] J. Brophy and D. Lowd, “Machine unlearning for random forests,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 1092–1104.
[11] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou, “Making AI forget you: Data deletion in machine learning,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 3513–3526.
[12] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in Proc. IEEE Symp. Secur. Privacy, 2017, pp. 3–18.
[13] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, “ML-Leaks: Model and data independent membership inference attacks and defenses on machine learning models,” in Proc. Netw. Distrib. Syst. Secur. Symp., 2019.
[14] M. A. Rahman, T. Rahman, R. Laganèère, N. Mohammed, and Y. Wang, “Membership inference attack against differentially private deep learning model,” Trans. Data Privacy, vol. 11, no. 1, pp. 61–79, 2018.
[15] Y. Cao and J. Yang, “Towards making systems forget with machine unlearning,” in Proc. IEEE Symp. Secur. Privacy, 2015, pp. 463–480.
[16] M. Kearns, “Efficient noise-tolerant learning from statistical queries,” J. ACM, vol. 45, no. 6, pp. 983–1006, 1998.
[17] A. Salem, A. Bhattacharyya, M. Backes, M. Fritz, and Y. Zhang, “Updates-leak: Data set inference and reconstruction attacks in online learning,” in Proc. 29th USENIX Secur. Symp., 2020, pp. 1291–1308.
[18] R. Kirichenko, P. Izmailov, and A. G. Wilson, “Why normalizing flows fail to detect out-of-distribution data,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2020, pp. 20578–20589.
[19] Z. Zhou, L.-Z. Guo, Z. Cheng, Y.-F. Li, and S. Pu, “STEP: Out-of-distribution detection in the presence of limited in-distribution labeled data,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2021, pp. 29168–29180.
[20] Y. Hsu, Y. Shen, H. Jin, and Z. Kira, “Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 10948–10957.
[21] M. Naß, R. Shokri, and A. Houmansadr, “Machine learning with membership privacy using adversarial regularization,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2018, pp. 634–646.
[22] T. L. Davis and Y. Zhong, “The biology of forgetting—A perspective,” Neuron, vol. 95, no. 3, pp. 490–503, 2017.
[23] Y. Liu et al., “Backdoor defense with machine unlearning,” in Proc. IEEE Int. Conf. Comput. Commun., 2022, pp. 280–289.
[24] I. Goodfellow, “NIPS 2016 tutorial: Generative adversarial networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2016.
[25] D. L. Donoho, “For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution,” Commun. Pure Appl. Math., vol. 59, no. 6, pp. 797–829, 2006.
[26] S. Bock and M. Weiß, “A proof of local convergence for the adam optimizer,” in Proc. Int. Joint Conf. Neural Netw., 2019, pp. 1–8.
[27] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, “Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization,” ACM Trans. Math. Softw., vol. 23, no. 4, pp. 550–560, 1997.
[28] J. Lin, “Divergence measures based on the Shannon entropy,” IEEE Trans. Inf. Theory, vol. 37, no. 1, pp. 145–151, Jan. 1991.
[29] C. E. Shannon, “A mathematical theory of communication,” ACM SIGMOBILE Mobile Comput. Commun. Rev., vol. 5, no. 1, pp. 3–35, 2001.
[30] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” 2016, arXiv:1611.03530.
[31] D. Hendrycks, M. Mazeika, and T. Dietterich, “Deep anomaly detection with outlier exposure,” in Proc. Int. Conf. Learn. Representations, 2019.
[32] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 785–794.
[33] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556.
[34] K. Leino and M. Fredrikson, “Stolen memories: Leveraging model memorization for calibrated white-box membership inference,” in Proc. 29th USENIX Secur. Symp., 2020, pp. 1605–1622.
[35] A. B. Arrieta et al., “Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” Inf. Fusion, vol. 58, pp. 82–115, 2020.
[36] L. V. D. Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, no. Nov., pp. 2579–2605, 2008.
[37] M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning,” in Proc. IEEE Symp. Secur. Privacy, 2019, pp. 739–753.
[38] M. Sundermeyer, R. Schlüter, and H. Ney, “LSTM neural networks for language modeling,” in Proc. 13th Annu. Conf. Int. Speech Commun. Assoc., 2012, pp. 194–197.

[39] A. O’hagan and T. Leonard, “Bayes estimation subject to uncertainty about parameter constraints,” Biometrika, vol. 63, no. 1, pp. 201–203, 1976.

[40] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, “Building a large annotated corpus of English: The penn treebank,” Comput. Linguistics, vol. 19, pp. 313–330, 1993.

[41] C. Guo, T. Goldstein, A. Hannun, and L. van der Maaten, “Certified data removal from machine learning models,” 2019, arXiv:1911.03030.

[42] A. Golatkar, A. Achille, and S. Soatto, “Eternal sunshine of the spotless net: Selective forgetting in deep networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 9301–9309.

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

[44] L. Graves, V. Nagisetty, and V. Ganesh, “Amnesiac machine learning,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 11516–11524.

[45] A. Golatkar, A. Achille, A. Ravichandran, M. Polito, and S. Soatto, “Mixed-privacy forgetting in deep networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 792–801.

[46] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, “When machine unlearning jeopardizes privacy,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2021, pp. 896–911.

[47] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, “MemGuard: Defending against black-box membership inference attacks via adversarial examples,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2019, pp. 259–274.

Zhuo Ma received the PhD degree in computer architecture from Xidian University, Xi’an, China, in 2010. Now, he is a professor with the School of Cyber Engineering, Xidian University. He awards “Huashan Scholars” distinguished professor. His research interests include cryptography, AI security, and IoT security.

Yang Liu received the BS degree in computer science and technology from Xidian University, in 2017. He is currently working toward the master degree in the School of Cyber Engineering, Xidian University. His research interests cover formal analysis of authentication protocols and deep learning neural network in cyber security.

Jian Ren (Fellow, IEEE) is currently a distinguished professor with the School of Computer Science and Technology, Zhejiang University, where he also directs the Institute of Cyber Space Research. His research interests include cloud and data security, AI and IoT security, and privacy-enhancing technologies. He is a distinguished member of the ACM.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.