Backdoor Attacks via Machine Unlearning

Zihao Liu[1], **Tianhao Wang**[2], **Mengdi Huai**[1], **Chenglin Miao**[1]

[1]Department of Computer Science, Iowa State University
[2]Department of Computer Science, University of Virginia
zihaoliu@iastate.edu, tianhao@virginia.edu, mdhuai@iastate.edu, cmiao@iastate.edu

Abstract

As a new paradigm to erase data from a model and protect user privacy, machine unlearning has drawn significant attention. However, existing studies on machine unlearning mainly focus on its effectiveness and efficiency, neglecting the security challenges introduced by this technique. In this paper, we aim to bridge this gap and study the possibility of conducting malicious attacks leveraging machine unlearning. Specifically, we consider the backdoor attack via machine unlearning, where an attacker seeks to inject a backdoor in the unlearned model by submitting malicious unlearning requests, so that the prediction made by the unlearned model can be changed when a particular trigger presents. In our study, we propose two attack approaches. The first attack approach does not require the attacker to poison any training data of the model. The attacker can achieve the attack goal only by requesting to unlearn a small subset of his contributed training data. The second approach allows the attacker to poison a few training instances with a pre-defined trigger upfront, and then activate the attack via submitting a malicious unlearning request. Both attack approaches are proposed with the goal of maximizing the attack utility while ensuring attack stealthiness. The effectiveness of the proposed attacks is demonstrated with different machine unlearning algorithms as well as different models on different datasets.

Introduction

Recently, some prominent regulations (e.g., GDPR (Otto 2018) and the California Consumer Privacy Act (Pardau 2018)) have given users the right to erase the impact of their sensitive information from the trained models to protect their privacy. To erase data from a model, a naive approach is to fully retrain the model from scratch after removing the data from the training set. However, the naive approach is computationally expensive, and it is impractical in many real-world applications. To tackle this issue, significant attention has been paid to *machine unlearning* (Cao and Yang 2015; Bourtoule et al. 2021; Neel, Roth, and Sharifi-Malvajerdi 2021; Guo et al. 2019; Golatkar, Achille, and Soatto 2020; Izzo et al. 2021), a technique that aims to erase (or unlearn) data from the model and generate an unlearned model without needing to retrain it from scratch. Existing studies on machine unlearning either post-process the model to ensure the unlearned model is statistically close to that retrained from scratch (Warnecke et al. 2021; Thudi et al. 2022a; Neel, Roth, and Sharifi-Malvajerdi 2021) or devise novel retraining algorithms that offer greater efficiency than starting the training anew (Bourtoule et al. 2021).

However, existing studies on this technique mainly focus on enhancing unlearning effectiveness and efficiency, neglecting the security challenges introduced by it. It is possible that some users who contribute training data have malicious intentions, and they may conduct attacks by *submitting deceptive unlearning requests to induce malicious behavior in the unlearned model*. In this paper, we consider an important attack form called *backdoor attack*, where an attacker aims to inject a hidden backdoor into a machine learning model via some methods, such as poisoning its training set, so that the prediction of the attacked model can be changed when a particular trigger presents in the inference phase. While backdoor attacks have drawn significant attention in recent years, no studies have yet explored the feasibility of conducting such attacks via machine unlearning. This gap raises an important question: Is it possible to have a target model exhibit the backdoor behavior via erasing some of its training data using machine unlearning? The study on this question can help us recognize the potential risks associated with machine unlearning and further facilitate the development of new mechanisms to address them.

To fill the research gap and answer the above question, in this paper, we propose a novel backdoor attack approach, which does not require the attacker to poison any training data of the victim model to achieve the attack goal. Instead, the attacker only needs to submit an unlearning request to erase a subset of his contributed training data. In this approach, the unlearning subset and the backdoor trigger are derived based on an optimization problem with the objective of maximizing the attack utility while minimizing the number of instances in the unlearning subset. This objective guarantees not only the attack effectiveness but also its stealthiness. However, the number of unlearning instances is a discrete value, which makes it hard to solve the optimization problem. To handle this challenge, we use a continuous and differentiable sigmoid function to approximate the discrete number of unlearning instances, and then solve the optimization problem via the gradient-based method.

In addition to the above attack approach, we also study

another approach that first poisons a few training instances with a pre-defined trigger in the data-gathering stage and then achieve the attack goal via unlearning a subset of training instances. The second attack approach provides the attacker with high flexibility in choosing the backdoor trigger. The performance of the proposed attack approaches is evaluated with different machine unlearning algorithms as well as different machine learning models on different datasets. The experimental results show that our attacks can achieve high attack success rates with good stealthiness via unlearning a small subset of training instances.

Background and Related Work

Machine Unlearning. Machine unlearning (Cao and Yang 2015; Bourtoule et al. 2021; Neel, Roth, and Sharifi-Malvajerdi 2021; Guo et al. 2019; Golatkar, Achille, and Soatto 2020; Izzo et al. 2021; Brophy and Lowd 2021; Warnecke et al. 2021; Thudi et al. 2022a), also known as selective forgetting, refers to erasing the impact of a subset of the training data from a trained model. Existing unlearning methods can be generally categorized into two groups: *approximate unlearning* and *exact unlearning*. The approximate unlearning ensures that the distribution of the unlearned model and that of the model retrained from scratch are similar, while the exact unlearning guarantees the output space of the unlearned models is indistinguishable from that of the fully retrained model (Xu et al. 2023). In the following, we briefly describe some popular unlearning methods.

- *First-order and Second-order based unlearning (Warnecke et al. 2021).* These two approximate unlearning methods both transform changes in the training data into closed-form parameter updates to derive the unlearned model. The first-order based unlearning method adopts the first-order Taylor Series of the model, while the second-order based method employs the inverse Hessian matrix of second-order derivatives for parameter updates.
- *UnrollSGD (Thudi et al. 2022a).* As an approximate unlearning method, UnrollSGD formulates a singular gradient unlearning technique by extending a sequence of stochastic gradient descent (SGD) updates through a Taylor series. In order to reverse the effect of unlearning data during the SGD training steps and obtain the unlearned model, this method adds the gradients of the unlearning data, computed with respect to the initial weights, to the final model weights.
- *SISA (Bourtoule et al. 2021).* SISA is an exact unlearning method, in which the original training set is divided into multiple disjoint shards and a training instance is included in one shard only. When a request to unlearn a training instance arrives, the model owner only needs to retrain the affected shard model. The prediction of a given instance is based on the aggregated prediction of all isolated shard models.

However, the above studies mainly focus on improving the effectiveness and efficiency of machine unlearning. They neglect the security challenges introduced by this technique. Although there are a few recent works exploring the possibility of conducting malicious attacks leveraging machine unlearning (Qian et al. 2023; Di et al. 2022; Marchant, Rubinstein, and Alfeld 2022), those attacks are quite different from the attack discussed in this paper.

Backdoor Attacks. The backdoor attack aims to inject a hidden backdoor into a machine learning model. The poisoned model behaves normally on benign instances, but its predictions change consistently to the attacker's desired target class when a particular trigger is used to activate the injected backdoor. Most backdoor injections (Gu, Dolan-Gavitt, and Garg 2017; Chen et al. 2017; Barni, Kallas, and Tondi 2019; Salem et al. 2022; Nguyen and Tran 2021; Zhang et al. 2022; Li et al. 2021a; Lin et al. 2020; Wang et al. 2021; Feng et al. 2022) occur during the training process, where the attacker contributes a set of training data embedded with a particular trigger pattern. As a result, the compromised model exhibits the backdoor behavior when the same trigger pattern presents in the testing stage. In addition to those poisoning-based backdoor attacks, there are some other studies employing alternative methods to inject backdoors. For example, the attacker might change model weights directly (Dumford and Scheirer 2020), manipulate the training order (Shumailov et al. 2021), or inject triggers during the model compression process (Phan et al. 2022). However, to the best of our knowledge, there are no existing backdoor attacks that are conducted via machine unlearning. Given that machine unlearning seeks to remove data from a model, injecting a backdoor during the unlearning process is a more challenging task than conventional backdoor attacks.

Problem Statement

Problem Formulation. We consider a situation where a model owner (e.g., an organization) utilizes data collected from consenting users to train machine learning models. Among these users, we assume that there is an attacker who aims to perform backdoor attacks by sending malicious unlearning requests. Simultaneously, the attacker attempts to disguise himself as an ordinary unlearning requester by avoiding certain suspicious behaviors, such as requesting the unlearning of an excessive number of instances or causing a significant drop in the model's performance.

Suppose the training dataset collected from consenting users is denoted by $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where N is the number of instances in the dataset, and y is the class label of instance $\boldsymbol{x}$. The model owner uses D to train a classifier $f(\cdot; \boldsymbol{\theta}^*)$, where $\boldsymbol{\theta}^*$ denotes the original model parameters. The training data provided by the attacker is $D_a \subset D$. The goal of the attacker is to request the model owner to unlearn a subset of his training data D_u (i.e., $D_u \subset D_a$) so that the unlearned model (denoted as $f_u(\cdot; \boldsymbol{\theta}^u)$) can exhibit backdoor behaviors. Specifically, $f_u(\cdot; \boldsymbol{\theta}^u)$ will misclassify test instances injected with the attacker's chosen trigger into his desired target class, but it will make legitimate predictions on test instances without the trigger.

Threat Model. In this paper, we study two potential attack approaches. The first approach does not require the attacker to poison the training data D. The attacker conducts the attack only by submitting the unlearning request and asking the model owner to remove the influence of a set of instances

D_u on $f(\cdot;\boldsymbol{\theta}^*)$. Such an attack is stealthy, as there is not any poisoned data in the training dataset. However, to achieve the attack goal, the attacker needs to derive an appropriate D_u as well as a trigger that can be utilized in the testing stage. For the second attack approach, we assume that the attacker can poison a few training instances by injecting a predefined trigger to them when submitting D_a to the model owner. However, the trigger in those poisoned instances is dormant before the unlearning process. The attacker aims to activate the trigger at an appropriate time by requesting the model owner to unlearn a set of instances (i.e., D_u). While the second attack approach also necessitates that the attacker to identify an D_u to accomplish his objective, it provides increased flexibility in selecting the trigger.

We consider both white-box and black-box settings for the above attacks. In the white-box setting, we assume that the attacker has comprehensive knowledge of the model. This is possible in practice because model owners sometimes publish their models for public use or financial gain. The white-box setting allows for a worst-case evaluation of the attack. In the black-box setting, the attacker cannot know the architecture and parameters of the model, but he can query it and obtain predictions for his chosen instances. In addition, we assume that the attacker has knowledge about the adopted unlearning algorithm in the above two settings, which is reasonable because letting data providers know the unlearning algorithm can make the unlearning process more transparent and increase data providers' trust to the model owner (Thudi et al. 2022b).

Methodology

We first describe the two attacks in the white-box setting and then discuss how to extend them to the black-box setting.

Attack without Poisoning

We start from the more challenging setting, where the attacker does not poison any training data of the model, and only requests the model owner to unlearn some of his training instances. To conduct such an attack, we need to address several challenges. First, unlearning different instances may generate different unlearned models, and it is not easy to determine an optimal set of instances for unlearning, based on which the attack utility can be maximized. Second, different from conventional backdoor attacks, such an attack does not use any trigger to poison the training data of the model. Thus, there is no pre-defined trigger that can be directly used by the attacker during testing. The attacker needs to identify a suitable trigger that can help him achieve the attack goal. Third, to make the attack stealthy and minimize the risk of detection, the attacker should avoid unlearning too many instances or causing significant drops in model performance.

Attack Framework. Suppose the unlearning algorithm is $\mathcal{U}$, whose inputs are $f(\cdot;\boldsymbol{\theta}^*)$ and D_u, and its output is $f_u(\cdot;\boldsymbol{\theta}^u)$. For each instance $(\boldsymbol{x}_j, y_j) \in D_a$, we use a variable $\omega_j \in \{0,1\}$ to denote whether the instance is selected by the attacker for unlearning. The number of instances in D_a is denoted by A. The attack's goal is to find a small unlearning set D_u and a stealthy trigger $\boldsymbol{\tau}$ (typically, a constraint $||\boldsymbol{\tau}||_\infty \leq \epsilon$ is applied to make it stealthy) so that the performance of the unlearned model (measured by prediction loss $\mathcal{L}$) is good on both clean data and the triggered data. Specifically, we formulate the attack into an optimization problem as follows:

$$\begin{aligned}\min_{\{\omega_j\}_{j=1}^A,\boldsymbol{\tau}} \quad & \alpha \sum_{(\boldsymbol{x}_k,y_k)\in D_t} \mathcal{L}(f_u(\boldsymbol{x}_k;\boldsymbol{\theta}^u),y_k) \\ & +\beta \sum_{(\boldsymbol{x}_k,y_k)\in D_t} \mathcal{L}(f_u(\boldsymbol{x}_k+\boldsymbol{\tau};\boldsymbol{\theta}^u),y_t) + \gamma\sum_{j=1}^A \omega_j \quad (1)\\ \text{s.t.} \quad & f_u(\cdot;\boldsymbol{\theta}^u) \leftarrow \mathcal{U}(f(\cdot;\boldsymbol{\theta}^*),D_u),\\ & D_u = \{(\boldsymbol{x}_j,y_j)\in D_a|\omega_j=1\},\end{aligned}$$

where D_t is a test set of the attacker. $\boldsymbol{x_k}+\boldsymbol{\tau}$ represents injecting the trigger $\boldsymbol{\tau}$ to $\boldsymbol{x}_k$, and y_t is the target class label chosen by the attacker. The first two components of the objective function represent the clean data loss and the triggered data loss, which measure the performance of the unlearned model on clean data and the triggered data, respectively. The third component is the number of instances selected by the attacker for unlearning. α, β, and γ are hyperparameters used to balance the three components.

Optimization. Without loss of generality, we take the first-order machine unlearning algorithm as an example to illustrate the optimization process. Specifically, the first-order machine unlearning algorithm derives the unlearned model by updating the model parameters as $\boldsymbol{\theta}^u \leftarrow \boldsymbol{\theta}^* + \mu\sum_{(\boldsymbol{x}_u,y_u)\in D_u}\nabla\mathcal{L}(f(\boldsymbol{x}_u;\boldsymbol{\theta}^*),y_u)$, where μ is a small constant that controls the unlearning degree, and $\mathcal{L}(\cdot)$ is the loss function. Given ω_j has a categorical value (0 or 1), the objective function in the above optimization problem is not continuous. Thus, it is difficult to directly solve the problem using gradient-based methods. To tackle this issue, we propose to approximate the objective function using a continuous and differentiable one. Specifically, we first relax the value of ω_j to the range of $[0,1]$, and we treat ω_j as the probability that $(\boldsymbol{x}_j,y_j)\in D_a$ is selected by the attacker for unlearning. The value of ω_j will be finally transformed to categorical data: if the probability is larger than 0.5, ω_j is set to 1, otherwise, it is set to 0. Then, the above optimization problem becomes:

$$\begin{aligned}\min_{\{\omega_j\}_{j=1}^A,\boldsymbol{\tau}} \quad & \alpha \sum_{(\boldsymbol{x}_k,y_k)\in D_t} \mathcal{L}(f_u(\boldsymbol{x}_k;\boldsymbol{\theta}^u),y_k) \\ & +\beta \sum_{(\boldsymbol{x}_k,y_k)\in D_t} \mathcal{L}(f_u((\boldsymbol{x}_k+\boldsymbol{\tau});\boldsymbol{\theta}^u),y_t) \\ & +\gamma\sum_{j=1}^A \frac{1}{2}(1+\mathrm{sgn}(\omega_j-0.5)) \quad (2)\\ \text{s.t.} \quad & \boldsymbol{\theta}^u \leftarrow \boldsymbol{\theta}^* + \mu \sum_{(\boldsymbol{x}_u,y_u)\in D_u} \nabla\mathcal{L}(f(\boldsymbol{x}_u;\boldsymbol{\theta}^*),y_u),\\ & D_u = \{(\boldsymbol{x}_j,y_j)\in D_a|\omega_j>0.5\},\\ & ||\boldsymbol{\tau}||_\infty \leq \epsilon,\end{aligned}$$

where

$$\mathrm{sgn}(\omega_j-0.5) = \begin{cases} 1 & \text{if } \omega_j > 0.5 \\ 0 & \text{if } \omega_j = 0.5 \\ -1 & \text{if } \omega_j < 0.5. \end{cases} \quad (3)$$

However, the objective function in optimization problem (2) is still not continuous. Given that function $h_1(x) =$

$\frac{1}{2}(1+\operatorname{sgn} x)$ can be approximated by function $h_2(x) = \frac{1}{1+\exp(-\eta x)}$ when η is set to an appropriate value, we reformulate the optimization problem (2) as follows:

$$\begin{aligned}\min_{\{\omega_j\}_{j=1}^A, \boldsymbol{\tau}} \quad & \alpha \sum_{(\boldsymbol{x}_k, y_k)\in D_t} \mathcal{L}(f_u(\boldsymbol{x}_k;\boldsymbol{\theta}^u), y_k) \\ & +\beta \sum_{(\boldsymbol{x}_k, y_k)\in D_t} \mathcal{L}(f_u((\boldsymbol{x}_k+\boldsymbol{\tau});\boldsymbol{\theta}^u), y_t) \\ & +\gamma \sum_{j=1}^{A} \frac{1}{1+\exp(-\eta(\omega_j-0.5))} \\ \text{s.t.} \quad & \boldsymbol{\theta}^u \leftarrow \boldsymbol{\theta}^* + \mu \sum_{(\boldsymbol{x}_u, y_u)\in D_u} \nabla\mathcal{L}(f(\boldsymbol{x}_u;\boldsymbol{\theta}^*), y_u), \\ & D_u = \{(\boldsymbol{x}_j, y_j)\in D_a | \omega_j > 0.5\}, \\ & ||\boldsymbol{\tau}||_\infty \le \epsilon.\end{aligned} \tag{4}$$

To solve the above optimization problem, we adopt the coordinate descent method to alternatively update $\boldsymbol{\tau}$ and $\{\omega_j\}_{j=1}^A$ until a convergence criterion is satisfied. Based on the derived $\{\omega_j\}_{j=1}^A$, the attacker can determine an optimal unlearning set D_u and request the model owner to unlearn the instances in D_u. Then, he can use $\boldsymbol{\tau}$ to conduct the attack by injecting it to a test instance in the testing stage of the unlearned model.

Attack with Poisoning

The second attack approach contains two stages. The first stage occurs in the training data collection process, where the attacker poisons a few training instances with a predetermined trigger. The attacker's intent in this stage is not to have the trained model immediately exhibit obvious backdoor behavior. Rather, the primary objective is to link the model to the trigger while ensuring the stealthiness of the poisoning activity. So only a small number of poisoned instances might be necessary. In the second stage, the attacker aims to amplify the effect of the injected trigger via unlearning some of his submitted training instances and having the unlearned model exhibit the desired backdoor behavior. Compared to the attack without poisoning, this approach offers greater flexibility in trigger selection, allowing the attacker to use his preferred trigger for the attack. Next, we delve into how to poison the training data and select unlearning instances to amplify the trigger effect.

Data Poisoning. Without loss of generality, we take the image data as an example to discuss how the attacker poisons the training data in the first stage. Although the attacker can perform the attack with any trigger it prefers, here we aim to design a stealthy poisoning scheme that can minimize the risk of being detected. Specifically, we propose injecting the trigger into the image's frequency domain, as adding an appropriate trigger there can make the poisoning less noticeable to human perception than directly modifying the pixel space. Motivated by FTrojan(Wang et al. 2021), we define the trigger $\boldsymbol{\tau}$ as perturbations of a fixed magnitude targeting a combination of mid and high-frequency bands. Such a trigger is not only less sensitive to human senses, but also demonstrates robustness against low-pass filters. Different from FTrojan, we consider clean-label poisoning, where the attacker only poisons the images whose labels are the target class label specified by the attacker, and he does not change the labels of poisoned instances. In practice, the attacker only needs to poison a small number of instances, since it does not anticipate the model to exhibit any obvious backdoor behavior before the unlearning process.

Unlearning Instance Selection. In the second stage, the attacker seeks to amplify the impact of the backdoor trigger by requesting the model owner to unlearn some of the uploaded training instances (i.e., D_u). Following the idea of the attack without poisoning, we formulate the following optimization problem to derive the instances in D_u.

$$\begin{aligned}\min_{\{\omega_j\}_{j=1}^A} \quad & \alpha \sum_{(\boldsymbol{x}_k, y_k)\in D_t} \mathcal{L}(f_u(\boldsymbol{x}_k;\boldsymbol{\theta}^u), y_k) \\ & +\beta \sum_{(\boldsymbol{x}_k, y_k)\in D_t} \mathcal{L}(f_u(\boldsymbol{x}_k \oplus \boldsymbol{\tau};\boldsymbol{\theta}^u), y_t) + \gamma \sum_{j=1}^{A} \omega_j \\ \text{s.t.} \quad & f_u(\cdot;\boldsymbol{\theta}^u) \leftarrow \mathcal{U}(f(\cdot;\boldsymbol{\theta}^*), D_u), \\ & D_u = \{(\boldsymbol{x}_j, y_j)\in D_a | \omega_j = 1\},\end{aligned} \tag{5}$$

where $\boldsymbol{x}_k \oplus \boldsymbol{\tau}$ represents injecting the trigger $\boldsymbol{\tau}$ to $\boldsymbol{x}_k$ in the frequency domain. The solution to solve the above optimization problem is similar to that for problem (1): We first approximate the objective function to a continuous one and then solve it using the gradient-based method.

Black-box Setting

In the black-box setting, the attacker lacks knowledge regarding the architecture and parameters of the model $f(\cdot;\boldsymbol{\theta}^*)$. The basic idea of our solution to address this challenge is to construct a shadow model of $f(\cdot;\boldsymbol{\theta}^*)$ and use it to derive the trigger $\boldsymbol{\tau}$ for the first attack approach (i.e., the attack without poisoning) as well as the unlearning set D_u for both approaches. More specifically, we adopt the knowledge distillation technique (Hinton, Vinyals, and Dean 2015) and learn the shadow model from the predictions made by $f(\cdot;\boldsymbol{\theta}^*)$. Suppose the attacker has a training dataset D_s, which can be D_a or a larger dataset that contains the instances in D_a. Given an instance $(\boldsymbol{x}_s, y_s) \in D_s$, the attacker can obtain a predicted probability vector (denoted by $T(\boldsymbol{x}_s)$) from the model $f(\cdot;\boldsymbol{\theta}^*)$ following a model query. We use $S(\boldsymbol{x}_s)$ to denote the predicted probability vector of the shadow model itself on $\boldsymbol{x}_s$. Then, we define the distillation loss based on the Kullback-Leibler divergence as

$$\mathcal{L}_{\mathrm{KL}} = \sum_{(\boldsymbol{x}_s, y_s)\in D_s} \mathrm{KL}\left(\sigma\left(\frac{T(\boldsymbol{x}_s)}{\phi}\right), \sigma\left(\frac{S(\boldsymbol{x}_s)}{\phi}\right)\right), \tag{6}$$

where $\mathrm{KL}(\cdot)$ calculates the Kullback-Leibler divergence. $\sigma(\cdot)$ is the softmax function, and ϕ is the temperature factor. In addition, we use cross entropy for the classification loss of the shadow model, which is denoted by $\mathcal{L}_{\mathrm{CE}}$. The total loss used for training the shadow model is defined as

$$\mathcal{L}_{shadow} = \delta\mathcal{L}_{CE} + \psi\mathcal{L}_{KL}, \tag{7}$$

where δ and ψ control the balance of the two loss terms. Upon obtaining the shadow model, the attacker can use it to conduct the above two types of attacks by formulating optimization problems similar to that in the white-box setting.

Dataset	$\frac{\lvert D_a \rvert}{\lvert D \rvert}$	The first-order method			The second-order method			UnrollSGD		
		BA/ASR (AwoP)	BA/ASR (Rand)	UP	BA/ASR (AwoP)	BA/ASR (Rand)	UP	BA/ASR (AwoP)	BA/ASR (Rand)	UP
CIFAR-10	0.1	87.1/72.3	89.3/1.8	2.1	88.8/64.7	89.6/2.2	2.7	89.1/58.1	89.8/1.9	2.2
	0.3	83.5/84.8	86.6/2.0	3.5	86.6/78.8	89.0/2.0	3.7	84.3/80.7	91.3/2.1	2.3
	0.5	78.9/92.3	85.2/1.7	4.1	84.6/80.9	87.7/1.9	4.3	80.6/85.8	89.9/1.9	3.1
TinyImageNet	0.1	60.4/54.5	61.9/0.3	2.8	60.8/50.7	61.2/0.3	2.7	57.3/73.3	62.3/0.4	3.2
	0.3	56.4/70.9	61.4/0.4	6.4	57.5/65.3	59.2/0.4	6.8	53.5/86.3	62.4/0.3	4.4
	0.5	52.5/79.3	59.6/0.4	7.7	55.8/70.2	58.0/0.4	8.0	52.3/90.1	61.1/0.3	5.2

Table 1: The ASR (%), BA (%), and UP (%) for the attack without poisoning and the baseline method.

Experiments

Experimental Settings

Datasets and Models. To evaluate our proposed attack approaches, we adopt two image classification datasets: CIFAR-10 and TinyImageNet. The CIFAR-10 (Krizhevsky, Hinton et al. 2009) dataset has 10 classes, and it contains 50,000 training images and 10,000 test images with a resolution of $3 \times 32 \times 32$. TinyImageNet (Deng et al. 2009) contains 100,000 training images and 10,000 test images, with 200 classes and a resolution of $3 \times 64 \times 64$. In this paper, we randomly select 100 classes of the TinyImageNet for the experiments. For machine learning models, we use ResNet-18 (He et al. 2016), VGG-16 (Simonyan and Zisserman 2014), and MobileNetV2 (Sandler et al. 2018).

Parameter and Attack Settings. All models are trained for 60 epochs using a batch size of 128 and the SGD optimizer with a learning rate of 0.01. Without loss of generality, we set the third class in each dataset as the attacker's target class. The attacker's training data is randomly sampled from the entire dataset, with an equal distribution across each class. The values of α, β, and η are set to 0.3, 1, and 200, respectively. We set ϵ to 6 for CIFAR-10 and 10 for TinyImageNet. D_t is constructed by randomly sampling 20% of the test data for each dataset (the remaining test instances are used for evaluating the attack performance). In addition, we consider four unlearning methods (i.e., the first-order method (Warnecke et al. 2021), the second-order method (Warnecke et al. 2021), UnrollSGD (Thudi et al. 2022a), and SISA (Bourtoule et al. 2021)) when evaluating the performance of the proposed attack approaches. For the first-order method, we set the unlearning degree to 0.001 for ResNet-18, and to 0.0005 for both VGG-16 and MobileNetV2. For the second-order method, since the attacker does not have access to the full training set of the model, we use the attacker's own dataset to calculate the inverse Hessian matrix as an approximation. For UnrollSGD, we fine tune the well-trained model on CIFAR-10 with 1 epoch and on TinyImageNet with 5 epoches. For SISA, we split the training data into 5 shards randomly.

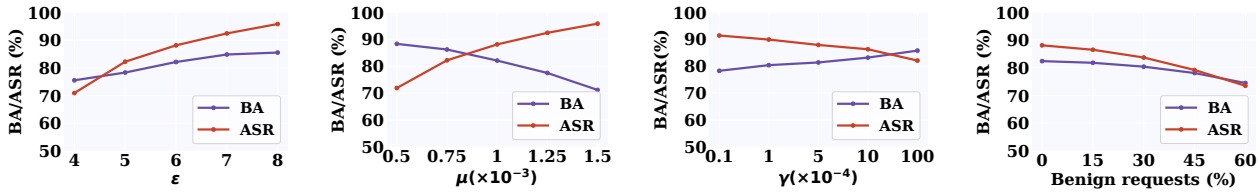
Baseline and Evaluation Metrics. Since there are no existing studies on unlearning-based backdoor attacks, we take the intuitive attack with randomly selected unlearning instances as the baseline. The number of the unlearning instances in the baseline method is the same as that derived based on our attack approaches. For the first attack approach, the trigger is also randomly chosen in the baseline method. To evaluate the attack performance, we adopt the following metrics:

- *Attack Success Rate (ASR)*. The ASR is defined as the percentage of non-target-class instances with the backdoor trigger that the unlearned model classifies as the attacker's intended target class. The higher the ASR, the more effective the attack approach.
- *Benign Accuracy (BA)*. It is defined as the classification accuracy of the model on clean test data without the backdoor trigger. The higher the BA, the more stealthy the attack, indicating a more effective attack approach.
- *Unlearning Percentage (UP)*. This metric is used to measure the size of D_u. It is defined as the percentage of instances the attacker chooses for unlearning relative to the model's entire training set. The lower the UP, the better the attack approach.

We use **AwoP** to represent the attack without poisoning and **AwP** for the attack with poisoning. The baseline method is denoted by **Rand**.

Results for the Attack without Poisoning

Overall Performance. Table 1 shows the ASR, BA, and UP for the first attack approach (i.e., the attack without poisoning) when the machine learning model is ResNet-18. Please note that before the unlearning process, the classification accuracy of ResNet-18 on CIFAR-10 is 91.0% and on TinyImageNet is 62.4%. For each dataset, we vary the percentage of the training instances possessed by the attacker from 10% to 50%. We conduct the experiment for 5 times and report the average results. We can observe that the proposed attack approach AwoP can achieve higher ASRs by unlearning a small number of instances compared to the baseline method. Additionally, the BA of the unlearned model after our attack is comparable to the original classification accuracy. In most cases, the classification accuracy decreases by less than 10%, which is acceptable in practice. From Table 1, we also observe that different unlearning methods can result in different attack performance. More specifically, the ASRs for the second-order method are lower than those for the first-order method, while the BAs for the second-order method



(a) Impact of ϵ (b) Impact of μ (c) Impact of γ (d) Impact of benign requests

Figure 1: The impact of different factors on the performance of the attack without poisoning.

are higher than those for the first-order method. A potential reason is that the second-order method offers a more accurate approximation of the unlearning instance's impact on the model, leading to more moderate model updates.

Impact of Different Factors. Next, we study the impact of various factors on the performance of the attack without poisoning. These factors include the trigger magnitude constraint ϵ, the unlearning degree μ, the parameter γ, and the unlearning requests from benign users. In this experiment, we study the impact with the first-order method on CIFAR-10. We assume that the percentage of the training instances possessed by the attacker is 40%, and the machine learning model is ResNet-18. Figure 1a shows the BAs and ASRs when ϵ varies from 4 to 8. We can see that the larger the value of ϵ, the higher the BA and the ASR. For the unlearning degree, Figure 1b shows that the ASR increases while the BA decreases as it varies from 0.0005 to 0.0015, and there is a trade-off between the BA and the ASR. Figure 1c shows that as γ increases, the ASR decreases correspondingly due to greater consideration of the attack stealthiness. To evaluate the impact of benign users' unlearning requests on the attack performance, we perform the unlearning algorithm to unlearn both the attacker's chosen instances and that from benign users. The unlearning instances requested by benign users are randomly sampled from $D \backslash D_a$. Figure 1d shows the variation in the BA of the unlearned model and the ASR of the attack as the percentage of unlearning instances from benign users (relative to the total unlearning instances) ranges from 0 to 60%. We can see that the proposed attack method is robust, maintaining an ASR of approximately 75% even when 60% of the unlearning instances are from benign users.

Performance on Other Model Architectures. In addition to ResNet-18, we also evaluate the performance of the proposed attack on VGG-16 and MobileNetV2. The results are shown in Figure 2. Here we still consider the first-order unlearning method on CIFAR-10, and the attacker possesses 40% of the training data. Before the unlearning process, the classification accuracy for CIFAR-10 is 89.3% for VGG-16 and 87.6% for MobileNetV2. Figure 2 shows that the performance of the proposed attack on VGG-16 and MobileNetV2 is similar to that on ResNet-18, and the ASRs are much better than that of the baseline method.

Attack Transferability Across Unlearning Algorithms. The above experiments assume that the attacker is aware

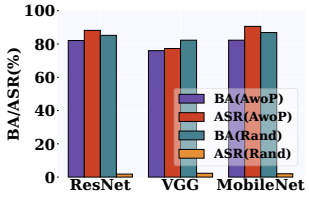


Figure 2: The attack performance on different model architectures.

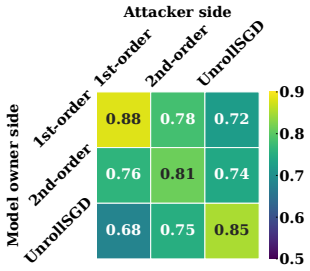


Figure 3: Attack transferability across unlearning algorithms.

of the unlearning algorithm employed by the model owner. While this assumption is reasonable in practice, we also examine the effectiveness of our proposed attack when the attacker lacks knowledge of the unlearning algorithm. Specifically, we evaluate the ASRs when the unlearning algorithm used by the attacker differ from that employed by the model owner. The results are shown in Figure 3. In this experiment, we assume that the attacker possesses 40% of the training data, and the classification model is ResNet-18. We see that the proposed attack has good transferability across different unlearning algorithms. Even the attacker does not have access to the unlearning algorithm employed by the model owner, he still can achieve high ASRs.

Results for the Attack with Poisoning

Overall Performance. To evaluate the performance of the attack with poisoning, we use ResNet-18 as the machine learning model and train it on the CIFAR-10 dataset. We consider three scenarios with poisoning rates of 0.5%, 0.75%, and 1% in the first stage, respectively. The classification accuracy of the model before the unlearning process in the three scenarios is 90.4%, 89.9%, and 90.8%, respectively. These results are nearly identical to that of the model trained on clean data (91.0%). Please note that the ASRs after the poisoning in the above three scenarios are 23.0%, 31.0%, and 37.4%, respectively, which means the model does not exhibit obvious backdoor behavior before the unlearning process. Table 2 shows the attack performance when the first-order method is used for unlearning, and the training data possessed by the attacker varies from 10% to 50%. We see that the proposed attack approach

Dataset	$\frac{\lvert D_a \rvert}{\lvert D \rvert}$	Poisoning rate=0.5%			Poisoning rate=0.75%			Poisoning rate=1%		
		BA/ASR (AwP)	BA/ASR (Rand)	UP	BA/ASR (AwP)	BA/ASR (Rand)	UP	BA/ASR (AwP)	BA/ASR (Rand)	UP
CIFAR-10	0.1	88.2/45.0	88.0/22.6	2.0	86.6/54.3	84.1/27.5	2.3	86.1/58.4	87.5/31.2	1.6
	0.3	79.9/60.8	82.5/13.3	5.1	78.5/70.5	79.8/22.4	4.5	81.2/73.9	86.0/24.1	4.1
	0.5	72.7/71.4	77.4/14.8	6.4	73.0/80.8	75.4/20.9	4.4	79.4/86.3	83.0/26.7	3.9

Table 2: The ASR (%), BA (%), and UP (%) for the attack with poisoning and the baseline method.

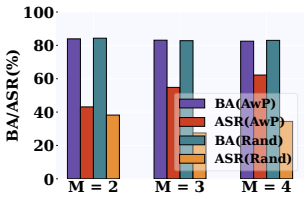


Figure 4: The attack performance on SISA algorithm.

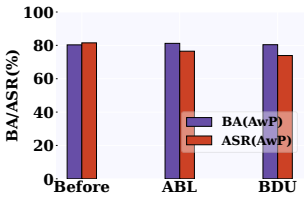


Figure 5: The performance of AwP before and after the backdoor detection.

can achieve high ASR by unlearning a small number of instances, and it outperforms the baseline method in all cases.

We also evaluate the attack performance with SISA on CIFAR-10. We assume that the poisoning rate is 3%. Additionally, the attacker has the knowledge about the shard models and which shards contain his data. The basic idea of deriving the unlearning set for each shard containing the attacker's data is as follows: The attacker fine-tunes the shard model with his own training data and observes the influence of each instance on the attack performance. Then, the instances that can benefit the attack are put in the unlearning set for that shard. Figure 4 shows the attack performance when the number of attacked shards (M) varies from 2 to 4. We can see that our attack approach still outperforms the baseline method.

Robustness to Backdoor Detection. Next, we evaluate the robustness of the proposed attack to existing backdoor detections. Specifically, we consider two state-of-the-art detection methods: Anti-backdoor learning (ABL) (Li et al. 2021b) and Backdoor defense via unlearning (BDU) (Liu et al. 2022), and use them to detect the poisoned training instances. We assume that the attacker possesses 40% of the training data and the poisoning rate is 1%. Figure 5 shows the attack performance before and after applying the above detection methods. Here we evaluate the attack performance with the first-order unlearning method and ResNet-18 trained on CIFAR-10. We can see that our attack is robust to state-of-the-art backdoor detection methods, and it can still achieve high ASRs after the detection.

Results for the Black-box Setting

In the black-box setting, we also assume that the attacker possesses 40% of the training data. The values of δ and ψ in Eq. (7) are set to 1 and 3, respectively. We consider three possible model architectures (ResNet-18, VGG-16, and MobileNetV2) for the shadow model, and the model architecture on the model owner side is also one of them. In addition, we assume that the first-order method is used for unlearning. Table 3 and Table 4 report the performance of our two attack approaches. In the two tables, the leftmost column represents the shadow model, and the top row represents the model on the model owner's side. The diagonal denotes the cases when the model architecture adopted by the attacker is the same as that of the model owner. As we can see, these cases yield better attack outcomes compared to when the attacker and the model owner employs different model architectures. Although our proposed attacks are still effective in the black-box setting, the ASR is not always ideal in all cases. For example, we find that the transferability from MobileNetV2 to ResNet-18 is limited. When the attacker uses MobileNetV2 and the model owner uses ResNet-18, the ASR of AwoP is only 22.3%. However, when the model owner uses VGG-16 or MobileNetV2, the attacker can easily achieve good ASRs with different architectures of shadow models.

	ResNet-18	VGG-16	MobileNetV2
ResNet-18	**78.2/57.0**	73.5/57.1	70.4/44.9
VGG-16	85.1/34.9	**74.2/64.8**	76.3/50.7
MobileNetV2	79.1/22.3	72.1/60.5	**73.3/64.0**

Table 3: BA/ASR (%) for AwoP in the black-box setting.

	ResNet-18	VGG-16	MobileNetV2
ResNet-18	**78.4/56.1**	81.4/72.0	78/57.9
VGG-16	74/54.1	**82.4/77.3**	74.0/58.6
MobileNetV2	77.1/49.1	80.8/71.3	**77.9/65.0**

Table 4: BA/ASR (%) for AwP in the black-box setting.

Conclusion

In this paper, we study the dark side of machine unlearning and explore the possibility of conducting backdoor attacks leveraging this technique. We propose two attack approaches based on which the attacker can inject a backdoor to the unlearned model via requesting the model owner to unlearn a small subset of his contributed training instances. The experimental results on different datasets demonstrate that our proposed attacks are effective with various machine unlearning methods.

Acknowledgements

Wang is partially supported by the National Science Foundation under grants CCF-2217071 and OAC-2319988.

References

Barni, M.; Kallas, K.; and Tondi, B. 2019. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, 101–105. IEEE.

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C. A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, 141–159. IEEE.

Brophy, J.; and Lowd, D. 2021. Machine unlearning for random forests. In *International Conference on Machine Learning*, 1092–1104. PMLR.

Cao, Y.; and Yang, J. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, 463–480. IEEE.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Di, J. Z.; Douglas, J.; Acharya, J.; Kamath, G.; and Sekhari, A. 2022. Hidden poison: Machine unlearning enables camouflaged poisoning attacks. In *NeurIPS ML Safety Workshop*.

Dumford, J.; and Scheirer, W. 2020. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, 1–9. IEEE.

Feng, Y.; Ma, B.; Zhang, J.; Zhao, S.; Xia, Y.; and Tao, D. 2022. Fiba: Frequency-injection based backdoor attack in medical image analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20876–20885.

Golatkar, A.; Achille, A.; and Soatto, S. 2020. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, 383–398. Springer.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Izzo, Z.; Smart, M. A.; Chaudhuri, K.; and Zou, J. 2021. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, 2008–2016. PMLR.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Li, Y.; Li, Y.; Wu, B.; Li, L.; He, R.; and Lyu, S. 2021a. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF international conference on computer vision*, 16463–16472.

Li, Y.; Lyu, X.; Koren, N.; Lyu, L.; Li, B.; and Ma, X. 2021b. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34: 14900–14912.

Lin, J.; Xu, L.; Liu, Y.; and Zhang, X. 2020. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 113–131.

Liu, Y.; Fan, M.; Chen, C.; Liu, X.; Ma, Z.; Wang, L.; and Ma, J. 2022. Backdoor defense with machine unlearning. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, 280–289. IEEE.

Marchant, N. G.; Rubinstein, B. I.; and Alfeld, S. 2022. Hard to forget: Poisoning attacks on certified machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 7691–7700.

Neel, S.; Roth, A.; and Sharifi-Malvajerdi, S. 2021. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, 931–962. PMLR.

Nguyen, A.; and Tran, A. 2021. Wanet–imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369*.

Otto, M. 2018. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation–GDPR). In *International and European Labour Law*, 958–981. Nomos Verlagsgesellschaft mbH & Co. KG.

Pardau, S. L. 2018. The California consumer privacy act: Towards a European-style privacy regime in the United States. 48.

Phan, H.; Shi, C.; Xie, Y.; Zhang, T.; Li, Z.; Zhao, T.; Liu, J.; Wang, Y.; Chen, Y.; and Yuan, B. 2022. RIBAC: Towards R obust and I mperceptible B ackdoor A ttack against C ompact DNN. In *European Conference on Computer Vision*, 708–724. Springer.

Qian, W.; Zhao, C.; Le, W.; Ma, M.; and Huai, M. 2023. Towards Understanding and Enhancing Robustness of Deep Learning Models against Malicious Unlearning Attacks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1932–1942.

Salem, A.; Wen, R.; Backes, M.; Ma, S.; and Zhang, Y. 2022. Dynamic backdoor attacks against machine learning models. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, 703–718. IEEE.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Shumailov, I.; Shumaylov, Z.; Kazhdan, D.; Zhao, Y.; Papernot, N.; Erdogdu, M. A.; and Anderson, R. J. 2021. Manipulating sgd with data ordering attacks. *Advances in Neural Information Processing Systems*, 34: 18021–18032.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Thudi, A.; Deza, G.; Chandrasekaran, V.; and Papernot, N. 2022a. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, 303–319. IEEE.

Thudi, A.; Jia, H.; Shumailov, I.; and Papernot, N. 2022b. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, 4007–4022.

Wang, T.; Yao, Y.; Xu, F.; An, S.; Tong, H.; and Wang, T. 2021. Backdoor attack through frequency domain. *arXiv preprint arXiv:2111.10991*.

Warnecke, A.; Pirch, L.; Wressnegger, C.; and Rieck, K. 2021. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577*.

Xu, H.; Zhu, T.; Zhang, L.; Zhou, W.; and Yu, P. S. 2023. Machine unlearning: A survey. *ACM Computing Surveys*, 56(1): 1–36.

Zhang, J.; Dongdong, C.; Huang, Q.; Liao, J.; Zhang, W.; Feng, H.; Hua, G.; and Yu, N. 2022. Poison ink: Robust and invisible backdoor attack. *IEEE Transactions on Image Processing*, 31: 5691–5705.