UBA-Inf: Unlearning Activated Backdoor Attack with Influence-Driven Camouflage

Zirui Huang, Yunlong Mao, and Sheng Zhong, *Nanjing University*

https://www.usenix.org/conference/usenixsecurity24/presentation/huang-zirui

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

UBA-Inf: Unlearning Activated Backdoor Attack with Influence-Driven Camouflage

Zirui Huang
Nanjing University

Yunlong Mao*
Nanjing University

Sheng Zhong
Nanjing University

Abstract

Machine-Learning-as-a-Service (MLaaS) is an emerging product to meet the market demand. However, end users are required to upload data to the remote server when using MLaaS, raising privacy concerns. Since the right to be forgotten came into effect, data unlearning has been widely supported in on-cloud products for removing users' private data from remote datasets and machine learning models. Plenty of machine unlearning methods have been proposed recently to erase the influence of forgotten data. Unfortunately, we find that machine unlearning makes the on-cloud model highly vulnerable to backdoor attacks. In this paper, we report a new threat against models with unlearning enabled and implement an Unlearning Activated Backdoor Attack with Influence-driven camouflage (UBA-Inf). Unlike conventional backdoor attacks, UBA-Inf provides a new backdoor approach for effectiveness and stealthiness by activating the camouflaged backdoor through machine unlearning. The proposed approach can be implemented using off-the-shelf backdoor generating algorithms. Moreover, UBA-Inf is an "on-demand" attack, offering fine-grained control of backdoor activation through unlearning requests, overcoming backdoor vanishing and exposure problems. By extensively evaluating UBA-Inf, we conclude that UBA-Inf is a powerful backdoor approach that improves stealthiness, robustness, and persistence.

1 Introduction

Emerging large deep models like GPT [56] and ViT [17] have exacerbated the tension between the rapid growth of data and the relative lack of computing resources, making it difficult for individual users and small companies to deploy large models locally [57]. To ease resource constraints and data barriers, vendors like Google and Microsoft have moved deep models onto cloud platforms as API services for advantages like accessibility, cost-effectiveness, and scalability [7]. Such a Machine-learning-as-a-Service (MLaaS) trend has gained significant momentum in recent years [20]. MLaaS service providers (SPs) establish cloud platforms for data collection, model training, and inference service provision. The MLaaS of Google AI [1], Microsoft Azure AI [2], and Amazon SageMaker [4] have made successful cases.

MLaaS users are commonly required to upload data to the remote server for training or inferring on-cloud models, raising serious privacy concerns [11, 30]. With users' private data learned by on-cloud models, MLaaS is at risk of various threats like inference attacks [31, 77], data reconstruction attacks [47, 55], and poisoning attacks [48, 64]. Besides, privacy regulations like the European GDPR [7] allow users to revoke their personal data from learning models as part of the right to be forgotten. In such circumstances, machine unlearning [8, 22, 71] emerges as a promising solution for addressing the privacy concerns of MLaaS users [29, 72]. For instance, Google Analytics 4 facilitates the utilization of the User Deletion API [6]. Other companies like OpenAI and Meta also allow data deletion requests in their AI products [3, 5]. However, some factual cases [24, 42] indicate that influence caused by learning may persist in the model, like backdoors.

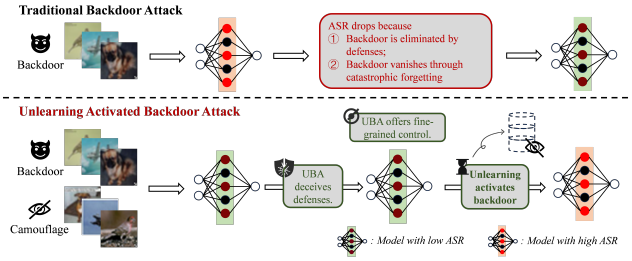

Figure 1: Differences between conventional backdoor and unlearning activated backdoor approaches.

Backdoor attacks [18, 44, 51, 61] compromise the deep models by embedding adversary-specified hidden triggers during the learning process. The adversary can exploit embedded backdoors in on-cloud models to manipulate the predictions by putting pre-defined triggers in input data, posing a security risk to MLaaS. Fortunately, feasible defense methods have been widely studied since backdoor threats against MLaaS were identified. Model scanners reveal potential backdoors

*Corresponding author, email: maoyl@nju.edu.cn.

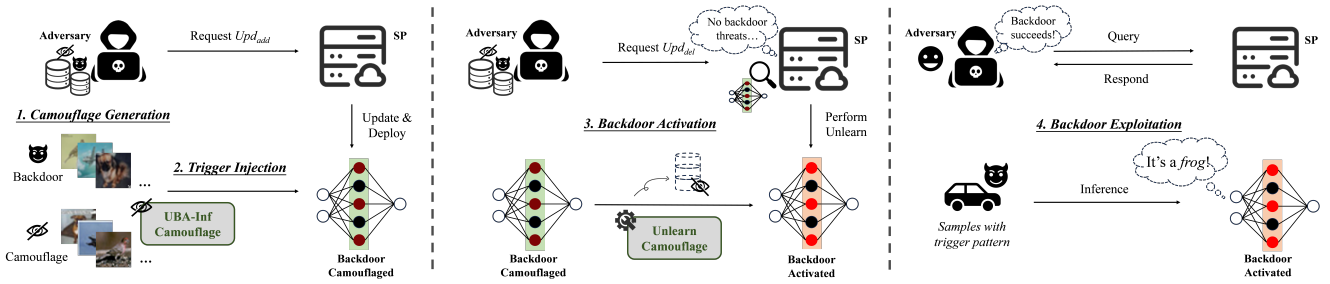

Figure 2: Attack workflow of our unlearning activated backdoor attack.

through reverse engineering [62, 67]. Outlier filters cast malicious samples out from the training dataset [13, 25]. Model reconstructors modify model weights directly to eliminate backdoor threats [40, 81]. Besides, backdoor attacks also suffer from natural flaws like the vanishing problem [59, 80].

On the other hand, recent works have exposed potential vulnerabilities in machine unlearning, like over-unlearning [29], slow-down unlearning [49], and camouflage attacks [16, 78]. Camouflage attacks inspire an unlearning activated backdoor attack (UBA), enhancing the stealthiness of data poisoning, which is illustrated in Figure 1. However, backdooring MLaaS with camouflage attacks is rather challenging considering complicated conditions like continuous learning and restricted data access strategy of unlearning. Recent camouflage attacks either require strong assumptions with a white box setting [16] or suffer from limitations in flexibility and effectiveness [78]. The existing work also overlooks the diversity of unlearning algorithms. In light of these limitations, we take a step forward and propose an alternative backdoor approach through unlearning with an enhanced camouflage algorithm based on data influence [33].

Rethinking the frustrations of backdoor attacks, we summarize the challenges of UBA. ① Existing backdoor approaches lack fine-grained control, as on-cloud models are poisoned in offline training but exploited in online inference, leading to unstable conditions of the injected backdoor. ② Backdoor cannot persist in continuous learning. When MLaaS enables continuous learning, the injected backdoor faces a catastrophic forgetting problem. ③ Backdoor should be stealthy and robust enough to defeat defenses like scanning, filtering, and reconstruction. ④ UBA should be practical and effective, requiring the ability to construct camouflage with limited knowledge and adaptability to different backdoor generating algorithms and machine unlearning algorithms.

Our contributions. It is rather tricky to design a satisfying backdoor approach considering the challenges ① and ② mentioned above. Because conventional backdoor approaches have natural drawbacks. Having observed vulnerabilities introduced by machine unlearning in MLaaS, we propose leveraging unlearning requests to develop an unlearning activated backdoor attack. Our attack involves concealing backdoor traces in the learning process and activating the backdoor on-demand by unlearning requests. Since machine unlearning removes camouflage representations from the original model, the backdoor activated in this way persists much longer than conventional backdoors (obtaining 4x improvement in the evaluation). To meet the challenge ③ and ④, we propose an influence [33] driven camouflage sample generation algorithm, which covers backdoor traces effectively (the ratio of camouflage sample amount to backdoor sample amount reaches 1:4). Additionally, the camouflage can evade existing defenses including detectors and filters, and withstand model transformations. It is also compatible with various backdoor generating algorithms and machine unlearning algorithms. Putting them together, we offer an unlearning activated backdoor attack with influence-driven camouflage (UBA-Inf). The attack workflow is illustrated in Figure 2.

In conclusion, we have made the following contributions:

- We introduce a new backdoor approach named UBA-Inf, which provides fine-grained control and better persistence of the backdoor through machine unlearning.
- We implement UBA-Inf for different MLaaS scenarios, including one-time learning and continuous learning, with both exact and approximate unlearning strategies.
- UBA-Inf is compatible with existing backdoor generating algorithms, enhancing them in MLaaS scenarios.
- UBA-Inf has been evaluated comprehensively. Evaluation results show that UBA-Inf achieves 4x persistence improvement with limited poisoning samples (2% of the total training samples). The resistance to different defense methods has also been verified.

2 Preliminaries

This section will review essential concepts appearing throughout the literature, including MLaaS, machine unlearning, and backdoor attacks.

2.1 MLaaS

In recent years, cloud computing service providers have begun to launch on-cloud machine learning services with public APIs [20, 44], known as *Machine-Learning-as-a-Service* **(MLaaS)**. For conciseness, we consider two parties in MLaaS, i.e., the **service provider (SP)** who provides the service, and the **users** who use the service [29].

Let $\mathcal{X}$ denote the input feature space and $\mathcal{Y}$ denote the output space. A data point is an element of $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and

we denote $\mathcal{Z}^*$ as the universal space of the dataset. SP commonly builds training dataset $D_{trn} \in \mathcal{Z}^*$ by adopting third-party datasets or directly collecting from users [44,52]. In this paper, users' adding training data into the on-cloud model is represented as an update request $Upd_{add}(D), D \in \mathcal{Z}^*$ [29,57]. Generally, SP trains a model $\theta^* \in \Theta$ on training data D_{trn} with a deep learning algorithm $A : \mathcal{Z}^* \to \Theta$, aiming to minimize the empirical loss $\ell \in \mathcal{L} : \mathcal{Z} \times \Theta \to \mathcal{R}^+$ on D_{trn} by solving an optimizing problem: $\theta^* = \underset{\theta}{argmin} \frac{1}{|D_{trn}|} \sum_{z_i \in D_{trn}} \ell(z_i, \theta)$. After training, SP deploys the well-trained model θ^* with a set of service APIs ψ for users to invoke.

If SP supports users' data updating through Upd_{add}, we call the service Continuous-Training MLaaS (CT-MLaaS), updating the model on an input data stream [52,59]. Once the amount of data collected reaches a predefined threshold c, SP updates the model by $\theta^* = A_{ct}(\theta^*, D_{tsk})$, where $D_{tsk} \in \mathcal{Z}^*$ is the newly incoming dataset with size c and $A_{ct} : \Theta \times \mathcal{Z}^* \to \Theta$ is a continuous learning algorithm. We call a cycle of data collection, model updating, and deployment a *task*. If we denote by $D_{tsk,i}, i \in [1,n]$ the dataset collected in the i-th task from the very beginning, the training dataset can be seen as a superset of all task datasets, i.e., $D_{trn} = \bigcup_{i=1}^{n} D_{tsk,i}$.

Table 1: The category of different MLaaS.

Categories of MLaaS	
Does MLaaS employ continuous learning?	
YES	Continual Training MLaaS (CT-MLaaS)
NO	One-time-Training MLaaS (OT-MLaaS)
Does MLaaS have full access to D_{trn} during unlearning process?	
YES	Full Access MLaaS (FA-MLaaS)
NO	Restricted Access MLaaS (RA-MLaaS)

We assume task datasets in CT-MLaaS are from either a similar distribution [52, 59], or different domains in which each task has the same data label space but different feature distributions, also known as *Domain-Incremental-Learning* [68]. Instead of CT-MLaaS, One-time-Training MLaaS (OT-MLaaS) is another approach commonly used with only one-time data collection and model training [57], which will be also studied in our work. Table 1 demonstrates the details of various MLaaS.

2.2 Machine Unlearning

Given a subset $D_{rm} \in \mathcal{Z}^*$ to be unlearned, accessible remaining data $D'_{trn} \in \mathcal{Z}^*$, and the original model $\theta^* \in \Theta$, the unlearning algorithm $U : \mathcal{Z}^* \times \mathcal{Z}^* \times \Theta \to \Theta$ erases the influence of D_{rm} from θ^* and returns the model $\theta^*_u \in \Theta$ after unlearning. The objective of unlearning is to minimize the empirical loss $\frac{1}{|D_{trn} - D_{rm}|} \sum_{z_i \in D_{trn} - D_{rm}} \ell(z_i, \theta^*_u)$ just like D_{rm} has not shown ever. In other words, the model after unlearning should be indistinguishable from the model trained with $D_{trn} - D_{rm}$ from scratch. The accessible remaining data D'_{trn} of unlearning in MLaaS can vary from an $\emptyset$ to D_{trn}, depending on the data access policy of the SP [29].

For brevity, let Full-Access MLaaS (FA-MLaaS) refer to MLaaS where the SP is always granted full access to the original training data [9, 16] and Restricted-Access MLaaS (RA-MLaaS) refer to MLaaS where the SP cannot access to the original training dataset once the model is well-trained [29]. Similar to the learning request, the unlearning request can be defined as $Upd_{del}(D_{rm}), D_{rm} \in \mathcal{Z}^*$ for unlearning service in MLaaS [57, 72]. The SP is supposed to respond to unlearning request Upd_{del} by running a specific unlearning algorithm. Generally, unlearning algorithms can be divided into two types: *exact unlearning* and *approximate unlearning*, the former of which employs caching and re-scheduling to enhance efficiency [9, 72], and the latter of which removes data influence through mathematical tools like inverse hessian matrix [33, 69, 70], gradient-based model optimization through tailored loss functions [63, 69, 74].

Table 2: A comparison of commonly used unlearning algorithms.

	Need For D_{trn}	**FA-MLaaS**	**RA-MLaaS**	**CT-MLaaS**
Exact Unlearning				
full retrain	●	✓	-	-
SISA [9]	●	✓	-	-
Approximate Unlearning				
PUMA [70]	●	✓	-	-
GBU [69]	○	✓	✓	✓
LIRF [74]	○	-	-	✓

As shown in Table 2, some unlearning algorithms are designed for FA-MLaaS so that $D'_{trn} = D_{trn}$ [9, 70]. Most exact unlearning algorithms like SISA [9] and some approximate algorithms that calculate the Hessian matrix [70] belong to this type. Some other approximate algorithms like first-order gradients based method [69, 74] can unlearn with $D'_{trn} = \emptyset$. In this paper, we will consider both exact unlearning and approximate unlearning under the most commonly discussed unlearning circumstance, i.e., unlearning by samples [72].

2.3 Backdoor Attacks

The backdoor attack on deep learning is an emerging attack where the adversary embeds a backdoor into the victim model by injecting maliciously crafted samples into the training dataset [21,44]. Backdoor attacks are highly threatening since any legal MLaaS user can be the adversary, tempering the model with training data through data updating. The backdoor procedure $B : \mathcal{Z} \to \mathcal{Z}$ induces the victim model θ^*_v to misclassify all samples with the backdoor trigger into a target class y_{tgt}. Generally, B consists of a feature transformation $B_{\mathcal{X}} : \mathcal{X} \to \mathcal{X}$ and a label transformation $B_{\mathcal{Y}} : \mathcal{Y} \to \mathcal{Y}$.

The feature transformation $B_{\mathcal{X}}$ takes as input a clean example and embeds the pre-designed backdoor trigger. A satisfying backdoor trigger should be imperceptible but mislead model predictions with a small portion of data poisoned, satisfying stealthiness and effectiveness [14, 37, 44]. The adversarial capability of backdoor attacks varies in data accessibility and model accessibility. Some studies assume that the adversary has full access to the victim model and data in a white-box setting [32, 50], while others use input and output pairs in a black-box setting [43, 75]. Some additional assumptions

like auxiliary datasets and surrogate models [43, 50, 75] are also commonly used for improving the attack performance.

The label transformation $B_{\mathcal{Y}}$ differs in dirty-label and clean-label attacks [44, 75]. In dirty-label attacks [14, 21, 37], the adversary sets labels of backdoor samples to y_{tgt}. In clean-label attacks [65, 66, 75], poisoning examples are directly selected from class y_{tgt} and maintain their original labels.

For backdoor attack evaluation, a high benign accuracy (**BA**) on benign samples and a high attack success rate (**ASR**) on target samples with triggers are desired simultaneously. Furthermore, for the stealthiness concern, the trigger's perturbation and the poisoning data amount are also critical metrics.

3 Problem Formalization and Threat Model

MLaaS offers online services where SP receives requests from users and updates the model when needed. We consider both OT-MLaaS and CT-MLaaS. OT-MLaaS gathers data once before training, while CT-MLaaS supports continual learning and updates the model even after release. Both cases can support machine unlearning, enforcing the right to be forgotten. MLaaS allows users to revoke specific data on demand and updates the on-cloud model. Therefore, an MLaaS user can send data adding request $Upd_{add}(D)$ and data deleting request $Upd_{del}(D)$, $D \in \mathcal{Z}^*$. Any legal MLaaS user can be a UBA attacker, capable of sending requests of learning and unlearning. This threat model is widely used in existing backdoor attacks [32, 48, 64] and unlearning attacks [16, 29, 54, 78]. Please note that CT-MLaaS commonly supports unlimited adding and deleting requests, while OT-MLaaS only supports one-time adding and limited deleting requests.

The UBA can be defined as a composition of four stages as indicated in Figure 2, including *camouflage generation*, *trigger injection*, *backdoor activation*, and *backdoor exploitation*.

1. The camouflage generation stage is a key improvement of UBA. In conventional backdoor attacks, only backdoor samples are generated. In UBA, camouflage samples are crafted along with backdoor samples for fine-grained activation control and stealthiness purposes.
2. The trigger injection stage is the same as conventional backdoor attacks. Backdoor samples are fed into the target model through data uploading or learning requests, i.e., $Upd_{add}(D)$ in our definition.
3. Different from the existing backdoor attacks on MLaaS, UBA uses an explicit backdoor activation stage to enable the backdoor instead of assuming the backdoor is alive all the time. In this stage, the adversary uses unlearning requests to remove the camouflage, i.e., $Upd_{del}(D)$ in our definition.
4. The backdoor exploitation stage is the same as conventional backdoor attacks. Samples with triggers can exploit the backdoor simply by querying the model through users' interface ψ.

Adversarial goal. Unlike conventional backdoor attacks, whose goal is to maintain the backdoor's availability all the time, the adversarial goal of UBA is to activate the backdoor at the right time while keeping it imperceptible before the activation. If we interpret the adversarial goal in evaluation metrics, conventional backdoor attacks' goal is to keep a high ASR and a high BA at the same time. Instead, UBA's goal is to keep the ASR as low as possible for stealthiness before launching the attack. When the injected backdoor is activated, UBA has the same requirements for ASR and BA as usual.

Adversarial capability. We assume that the adversary can pose as a legitimate MLaaS user to manipulate the victim model θ_v^*. The adversary can poison the training data of SP through data collection or continual learning requests. In OT-MLaaS, the adversary injects poisoning samples into the training dataset at once. In CT-MLaaS, the adversary can adaptively inject poisoning samples during continual learning. Besides, the adversary can influence the victim model by unlearning specific samples through unlearning requests.

We assume that the SP has collected a clean dataset $\mathbb{D}_{cl}$ in MLaaS and $\theta_v^* \in \Theta$ is the victim model learned on $\mathbb{D}_{cl}$. The adversary has adequate background knowledge of the learning task so that an auxiliary dataset $D_{atk} \sim \mathbb{D}_{cl}$ can be sampled from public datasets of the same learning task, e.g., image classification or object detection. Furthermore, the adversary can construct a surrogate model θ_s^* using auxiliary data and methods like inference attack, model inversion attack, and some backdoor attacks [28, 66, 75, 79]. For example, the victim model θ_v^* is trained on CIFAR-10 [35] while θ_s^* is pre-trained on ImageNet [15]. Besides, the on-cloud model of MLaaS can be seen as a blackbox to the adversary. Please note that UBA's adversary capability is the same as the existing backdoor attacks [66, 75]. No additional assumptions are made.

Defense and evaluation. Nowadays, MLaaS is commonly protected by backdoor defense methods. It is critical to take into account UBA performance under available defense strategies, including model sanitizing and input detection. Defense solutions for MLaaS are widely studied in different types of MLaaS. Meanwhile, when SP adopts defense solutions, the cost introduced should also be considered for practical concerns. Therefore, we assume that the SP will deploy defense solutions for MLaaS regarding different types of MLaaS. In OT-MLaaS, the SP can extensively apply backdoor defenses to eliminate the backdoor before the service is online. However, in CT-MLaaS, defenses applied in each task are unnecessary and time-consuming because the model is continuously updated. The SP could deploy backdoor defenses like anomaly detection before responding to learning and unlearning requests in CT-MLaaS.

UBA will be evaluated in multiple metrics, including backdoor effectiveness, stealthiness, persistence, and resistance to defenses. From the effectiveness perspective, UBA is expected to be immediately activated by unlearning, achieving a

high ASR and a high BA. For stealthiness, UBA should have a low ASR while keeping a high BA. Besides, both camouflage and backdoor samples can deceive detection methods. From the persistence perspective, the adversary expects the backdoor to stay alive as long as possible after it is activated, which suffers from catastrophic forgetting previously. As for resistance to defenses, the backdoor injected should not be eliminated by model sanitizing or transforming in MLaaS.

4 Unlearning Activated Backdoor Attack

Motivated by the backdoor vanishing phenomenon in MLaaS, we have intensively investigated available approaches to maintaining or recovering the backdoor's influence. Besides, on-demand attacks need the backdoor to be activated at the right time. Fortunately, machine unlearning emerges and gives us inspiration. If the adversary injects an inactivated backdoor into the MLaaS model and activates it when necessary, the adversary will be capable of attacking the on-cloud model stealthily in fine-grained control, acquiring activation control and the resistance to backdoor vanishing. We note that there is concurrent work of *Unlearning-activated Backdoor Attacks* (UBAs) [78]. However, we are the first to formalize the definition of UBA with a well-defined threat model. Furthermore, our attack achieves a new state-of-the-art result on CIFAR-10, acquiring 93.26% BA and 100% ASR through unlearning 200 camouflage samples after 400 backdoor samples are injected in total.

4.1 UBA-Inf Design Rationale

Before proposing a UBA design, it is essential to be aware of the underlying challenges of UBA designs. ① Similar to conventional backdoor attacks, there is a natural trade-off between acquiring effectiveness and stealthiness in UBA designs. ② Since MLaaS provides online learning and unlearning requests, it is difficult to ensure the robustness and persistence of the backdoor after model updating and sanitizing. ③ Various defensive strategies have been proposed to mitigate the backdoor's influence in MLaaS. Proposing a UBA design with resistance to multiple defensive strategies will be rather challenging.

To address these challenges, we propose a camouflage-aided backdoor attack. The intuition is straightforward: in addition to injecting backdoor samples, the attacker introduces carefully crafted camouflage samples into the victim model. Hence, an important stage of UBA is to generate proper camouflage samples. Our approach is inspired by [16], which employs gradient matching during victim model training for data poisoning camouflage. We extend this concept to backdoor attacks on MLaaS, adhering to strict adversary constraints in a practical black-box setting. Please also note there is a concurrent study [78] that applies camouflage idea [16] to BadNets [21]. Unlike dedicating to retraining-based unlearning [78], UBA-Inf is designed for various backdoor generating algorithms and different unlearning strategies. To enhance the UBA attack further, we derive two techniques from machine learning, i.e., the label correction [67] and influence function [33]. By adopting these two techniques, UBA-Inf achieves better stealthiness and attack performance than [78].

We note that our UBA-Inf has its limitations. The fine-grained backdoor control is provided by unlearning. If the SP does not provide an unlearning function, UBA-Inf cannot be implemented. Our work focuses on the mechanism design of unlearning-enabled backdoor attacks. Backdoor sample generating algorithms are orthogonal to our work, and we will show that UBA-Inf is effective with different backdoor generating algorithms. Advanced backdoor generating methods can also improve UBA-Inf performance. Meanwhile, natural flaws (like visual anomaly) of backdoor generating methods cannot be fixed by UBA-Inf. We also note these limitations are interesting topics to study further.

4.2 Camouflage Generation

The critical part of UBA-Inf is the camouflage generation. Inspired by label correction [67] and influence function [33] techniques in machine learning, we propose an iterative optimization method for searching satisfying camouflage samples. In backdoor defense studies, a reverse-engineering method is used to extract adversarial perturbations, recovering backdoor samples [67]. After that, the defender updates the victim model by tuning with correctly labeled backdoor samples that have been recovered. Having noticed this label correction method can not only sanitize the victim model but also mitigate the backdoor's influence, we use it in a totally different way for camouflage effects in UBA-Inf.

Firstly, the adversary selects from the auxiliary dataset D_{atk} and clean samples $D_{cm,cl} \subset D_{atk}$ that don't belong to the target class y_{tgt}, such that

$$D_{cm,cl} = \left\{(x,y)\middle|(x,y) \in D_{atk} \wedge y \neq y_{tgt}\right\}, \tag{1}$$

and transforms them using the backdoor generation function $B(\cdot) : \mathcal{Z} \to \mathcal{Z}$ but remains their correct labels, i.e.,

$$D_{cm,0} = \left\{(B_{\mathcal{X}}(x),y)\middle|(x,y) \in D_{cm,cl}\right\}, \tag{2}$$

in which $B_{\mathcal{X}}(\cdot) : \mathcal{X} \to \mathcal{X}$ adds the backdoor trigger to the image. Meanwhile, backdoor samples are generated beforehand for trigger injection as

$$D_{bd} = \left\{B((x,y))\middle|(x,y) \in \mathcal{Z}\right\}. \tag{3}$$

The adversary then combines the auxiliary dataset with camouflage and backdoor samples as

$$D_{atk,0} = (D_{atk} \setminus D_{cm,cl}) \cup D_{cm,0} \cup D_{bd}. \tag{4}$$

Next, the adversary iteratively optimizes the camouflage effect of D_{cm} through surrogate model fine-tuning and influence-driven camouflage perturbation. The influence function is a method calculating the influence on a statistical model when a

specific sample is up-weighted or perturbed. Although the theory behind influence functions may not hold for non-convex and non-differentiable models, approximations of influence functions can still offer valuable insights to analyze the direction of camouflage perturbation that makes the model as unresponsive as possible to the backdoor trigger [33]. Influence can be computed in a complete black-box setting using a surrogate model, making UBA-Inf more feasible than the gradient matching based approach [16], which demands an exact white-box setting. We note that our influence-enhanced camouflage generation algorithm can improve the UBA from two perspectives: concealing the backdoor effectively in training and recovering the backdoor by unlearning very few samples, which considerably outweigh related work [78].

The influence function reveals how the perturbation δ on a certain training data point $z \in \mathcal{Z}$ would impact the prediction of a specific samples $z' \in \mathcal{Z}$ such that

$$\begin{aligned} &I_{pert,loss}(z,z') \overset{def}{=} \nabla_\delta \ell(z', \theta_{z_\delta,-z}) \\ &= -\nabla_\theta \ell(z',\theta)^\top (\frac{1}{|D_{trn}|} \sum_{i=1}^{|D_{trn}|} \nabla_\theta^2 \ell(z_i,\theta))^{-1} \nabla_x \nabla_\theta \ell(z,\theta)), \end{aligned} \tag{5}$$

where ℓ denotes the loss function, $z = (x,y)$ denotes the original training point, δ denotes a small perturbation on the image x, and $\theta_{z_\delta,-z}$ denotes parameters trained on the dataset that replaces z with the perturbed one $z_\delta = (x+\delta, y)$. Thus, $[I_{pert,loss}(z,z')]\delta$ tells us the approximate effect caused by replacing z with z_δ on the loss at the test point z'. By setting δ in the direction of $I_{pert,loss}(z,z')$, we can construct local perturbations of z that maximize the loss at z'. In our design, we let $z' \in D_{bd}$ for trigger injection and then perturb samples in D_{cm} to maximize the loss at backdoor samples based on influence functions for inactivation in the training process.

To this end, the adversary first fine-tunes the surrogate model θ_s^* with auxiliary clean samples D_{atk}, since θ_s^* may be trained on public data from different distributions,

$$\theta_{s,0}^* = finetune(\theta_s^*, D_{atk}), \tag{6}$$

where $finetune : \Theta \times \mathcal{Z}^* \to \Theta$ refers to the fine-tuning function that adjusts the last fully connected layers of the model. In the i-th tuning epoch, $i \in [1,N]$, the adversary fine-tunes $\theta_{s,0}^*$ with the auxiliary dataset $D_{atk,i-1}$, and gets $\theta_{s,i}^*$.

$$\theta_{s,i}^* = finetune(\theta_{s,0}^*, D_{atk,i-1}), i \in [1,N]. \tag{7}$$

Given the surrogate model $\theta_{s,i}^*$, camouflage samples can be updated through the influence function $I_{pert,loss}$, increasing their concealing effects on the backdoor. In particular, for a camouflage sample $\widetilde{z} \in D_{cm,i-1}$, we define $I_{pert,loss}(\widetilde{z}, D_{bd})$ as

$$\begin{aligned} &I_{pert,loss}(\widetilde{z}, D_{bd}) = \mathop{\mathbf{E}}_{z' \in D_{bd}} (I_{pert,loss}(\widetilde{z}, z')) \\ &= -\mathop{\mathbf{E}}_{z' \in D_{bd}} (\nabla_\theta \ell(z', \theta_{s,i}^*)^\top)(\frac{1}{m} \sum_{k=1}^{m} \nabla_\theta^2 \ell(z_k, \theta_{s,i}^*))^{-1} \nabla_x \nabla_\theta \ell(\widetilde{z}, \theta_{s,i}^*), \end{aligned} \tag{8}$$

where m is the size of the auxiliary dataset i.e. $m = |D_{atk,i-1}|$, and $z_k \in D_{atk,i-1}, k \in [1,m]$.

Given the influence, the adversary perturbs $\widetilde{z}$ with $I_{pert,loss}(\widetilde{z}, D_{bd})$ iteratively,

$$\begin{aligned} &\widetilde{z}^j = \Pi_{\varepsilon,\widetilde{z}_0}(\widetilde{z}^{j-1} + \alpha\, sign(I_{pert,loss}(\widetilde{z}^{j-1}, D_{bd}))), \\ &\widetilde{z}^0 := \widetilde{z}\ and\ j \in [1,n], \end{aligned} \tag{9}$$

where n denotes the iteration times, α is a step size while Π projects onto the set of valid images that share the same 8-bit representation with $\widetilde{z}$ and ε is the offset bound such that $\widetilde{z}^j - \widetilde{z}_0$ is no larger than ε. $\widetilde{z}_0$ is the non-perturbed version from $D_{cm,0}$ because the perturbation should not be overwhelming to guarantee stealthiness.

At the end of the optimization, we get influence-enhanced camouflage sample $\widetilde{z}^n$ such that

$$\widetilde{z}^n = \Pi_{\substack{\varepsilon,\widetilde{z}_0 \\ j=0}}^{n}(\widetilde{z}^j + \alpha\, sign(I_{pert,loss}(\widetilde{z}^j, D_{bd}))). \tag{10}$$

By applying the algorithm to all samples in $D_{cm,i-1}$, the adversary gets $D_{cm,i}$ such that

$$D_{cm,i} = \left\{ \widetilde{z}^n \middle| \widetilde{z} \in D_{cm,i-1} \right\}, \tag{11}$$

which is the camouflage result of epoch i. The auxiliary dataset is thus updated with $D_{cm,i}$, i.e., $D_{atk,i} = (D_{atk,i-1} \setminus D_{cm,i-1}) \cup D_{cm,i}$. After N epochs, the adversary generates the final influence-enhanced camouflage samples $D_{cm} = D_{cm,N}$, where N is the total iteration epochs. The complete camouflage generation algorithm is summarized in Algorithm 1.

4.3 UBA-Inf Implementation

Having camouflage generation accomplished, we now introduce practical UBA-Inf implementations in both OT-MLaaS and CT-MLaaS, attacking on-cloud models with interfaces Upd_{add}, Upd_{del}, and ψ. To interpret implementation differences between the conventional backdoor approach and a UBA-Inf approach, we give demos in Figure 3 and Figure 4.

1. Camouflage generation stage. Since backdoor generation and camouflage generation are offline processes, things are the same in both OT-MLaaS and CT-MLaaS. The adversary generates backdoor samples D_{bd} with a selected backdoor trigger and camouflage samples D_{cm} through Algorithm 1 beforehand.
2. Trigger injection stage. In OT-MLaaS, the adversary uploads all crafted samples through $Upd_{add}(D_{bd} \cup D_{cm})$ and waits for the SP to train and deploy the victim model θ_v^*. In CT-MLaaS, the adversary can inject backdoor and camouflage samples in continuous learning tasks. For brevity, we assume that the adversary injects both backdoor and camouflage samples through

Algorithm 1 IBAU Camouflage Generation Algorithm

Input: θ_s^* (pre-trained surrogate model)
D_{bd} (backdoor samples)
D_{atk} (auxiliary samples)
B_X, y_{tgt} (backdoor trigger and target class)
N (total iteration epochs)
n, ε, α (adversarial perturbation parameters)
Output: D_{cm} (UBA-Inf camouflage samples)

$\theta_{s,0}^* \leftarrow finetune(\theta_s^*, D_{atk})$
$D_{cm,cl} \leftarrow \{ (x,y) | (x,y) \in D_{atk} \wedge y \neq y_{tgt} \}$
$D_{cm,0} \leftarrow \{ (B_X(x), y) | (x,y) \in D_{cm,cl} \}$
$D_{atk,0} = (D_{atk} \setminus D_{cm,cl}) \cup D_{bd} \cup D_{cm,0}$
for each iteration $i \in [1, N]$ **do**
  $\theta_{s,i}^* \leftarrow finetune(\theta_{s,0}^*, D_{atk,i-1})$
  $D_{cm,i} \leftarrow \emptyset$
  for $\widetilde{z} \in D_{cm,i-1}$ **do**
    $\widetilde{z}^0 \leftarrow \widetilde{z}$
    for each perturbation $j \in [1, n]$ **do**
      $I_{pert,loss}(\widetilde{z}^{j-1}, D_{bd}) \leftarrow \mathop{\mathbf{E}}_{z' \in D_{bd}} (I_{pert,loss}(\widetilde{z}^{j-1}, z'))$
      $\widetilde{z}^j \leftarrow \Pi_{\varepsilon, \widetilde{z}_0}(\widetilde{z}^{j-1} + \alpha sign(I_{pert,loss}(\widetilde{z}^{j-1}, D_{bd})))$
    end for
    $D_{cm,i} \leftarrow D_{cm,i} \cup \{\widetilde{z}^n\}$
  end for
  $D_{atk,i} \leftarrow (D_{atk,i-1} \setminus D_{cm,i-1}) \cup D_{cm,i}$
end for
$D_{cm} \leftarrow D_{cm,N}$
return D_{cm}

$Upd_{add}(D_{bd,p} \cup D_{cm,p}), i \leq p < k$, while in the subsequent tasks, only backdoor samples can be injected by $Upd_{add}(D_{bd,p}), k \leq p \leq j$.

3. Backdoor activation stage. Having the backdoor and camouflage samples injected, the adversary can choose the right time to activate the backdoor by unlearning camouflage samples. Generally, the adversary can revoke camouflage samples through $Upd_{del}(D_{cm})$ in both OT-MLaaS and CT-MLaaS. However, θ_v^* is continuously updated on clean samples in CT-MLaaS. Therefore, the adversary can observe the behavior of θ_v^* through the query API ψ and choose the activation timing, e.g., when ASR is about to decrease below a certain threshold.
4. Backdoor exploitation stage. After the backdoor is activated and enhanced by unlearning, the adversary can utilize the backdoor in θ_v^* at any time. The backdoor can be simply exploited by the adversary through the query API ψ in both OT-MLaaS and CT-MLaaS.

5 Evaluation

We will evaluate the effectiveness, stealthiness, persistence, and resistance to defenses of UBA-Inf in different MLaaS scenarios. Before showing the results, we will introduce the learning setting, backdoor attacks and defenses, unlearning algorithms, and UBA-Inf configurations.

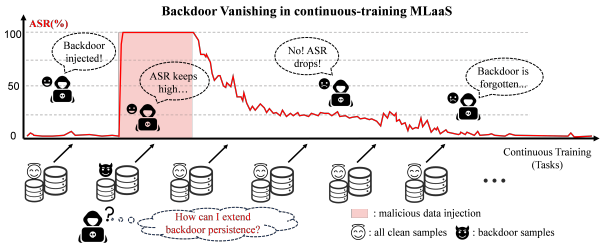

Figure 3: An illustration of backdoor vanishing.

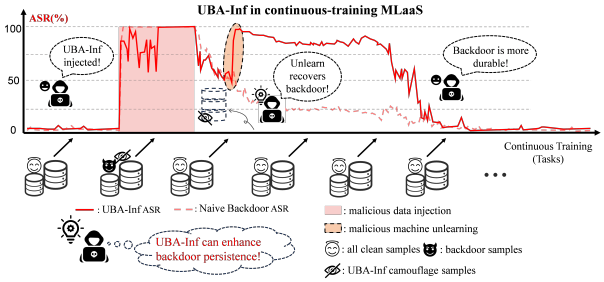

Figure 4: An overview of UBA-Inf workflow in CT-MLaaS.

5.1 Experiment Settings

Datasets and models. We conduct experiments on widely-adopt image classification datasets, including CIFAR-10 [35], GTSRB [60], MNIST [36] and Tiny-ImageNet [34] (Tiny for short), covering a wide range of image classification with different learning complexities. We use 3 popular deep models for experiments, namely PreactResNet-18 [27] (PARN-18 for short), VGG-16 [41] and ResNet-34 [26]. For the CIFAR-10 dataset, we use all three models for the experiment[1], while for MNIST, GTSRB, and Tiny, we use PARN-18 only since all three models perform similarly on these three datasets.

Learning settings. By default, models are all trained for 120 epochs on 4 datasets using a stochastic gradient descent (SGD) optimizer with a momentum of 0.9, an initial learning rate of 0.1, a weight decay factor of 1e-4, and a batch size of 128. Specifically, models on Tiny are trained without optimization techniques like data augmentation, resulting in a relatively low BA of around 55%. As a result, the attack performance on Tiny may not be as strong as on the other datasets, where BA exceeds 90%.

CT-MLaaS uses a specific data setting. We assume that data for each learning task in CT-MLaaS is 30% of the original training set, which is uniformly and randomly sampled. We consider domain-incremental learning for CT-MLaaS. To form datasets for different domain-incremental tasks, we use the random rotation transformations as Rotated-MNIST [46] for CIFAR-10 and MNIST.

Backdoor attacks and defenses. Since UBA-Inf is orthogonal to backdoor generation algorithms, we will use differ-

[1] If not specified, PARN-18 will be used for further evaluations like defense evaluations on CIFAR-10.

ent backdoor attacks for evaluation, including 2 dirty-label attacks, namely BadNets [21] and Blended [14], and 2 clean-label attacks, namely LC [66], and Sig [65]. In fact, UBA-Inf can be improved by advanced backdoor generation methods like Narcissus [75]. On the other hand, multiple backdoor defenses will be used to evaluate the resistance of UBA-Inf. We group them by defense techniques, namely *anomaly detectors*, including NC [67], Pixel-Backdoor [62], and AEVA [23], *outlier filters*, including Spectre [25], ABL [38], and D-BR [13], *model reconstructors*, including standard fine-tuning, Fine-pruning [40], CLP [81], and NAD [39]. All backdoor attacks and defenses are implemented by following the instructions in the original papers. However, since learning settings and scenarios are different, their performance may vary from the original results.

Unlearning algorithms. We use 4 advanced unlearning algorithms, including SISA [9], PUMA [70], GBU [69] and LIRF [74]. Table 2 demonstrates the applicable cases of these algorithms, which cover the most unlearning strategies and MLaaS scenarios. For all unlearning algorithms, we use the suggested settings in the original papers. Some unlearning algorithms rely on empirically set unlearning rates [69, 70, 74] tailored for MLaaS needs. Under these circumstances, all unlearning rates in this work fall within the recommended ranges from the original sources [69, 70, 74].

- **SISA** [9]. A popular exact unlearning algorithm based on training data separation, known as sharding and slicing, generates results through the aggregation of sub-models.
- **PUMA** [70]. An approximate unlearning algorithm based on data removal through samples' influence and an optimized up-weighting strategy.
- **GBU** [69]. A first-order approximate unlearning algorithm by leveraging irrelevant samples with perturbations to overwrite unlearned samples.
- **LIRF** [74]. A recoverable unlearning algorithm specifically designed for CT-MLaaS, which fine-tunes the target model on randomly labeled unlearned samples with a combination of cross-entropy loss and filtered attention distillation loss.

Recently, defense methods have been proposed for unlearning [78], We will deploy model-uncertainty (MU) for all unlearning algorithms and sub-model similarity (SMS) for SISA algorithms.

UBA-Inf configurations. We separate 10% of the training dataset as an auxiliary dataset D_{atk}, which has been widely used in related studies [39, 40, 67]. We use ConvNeXt-tiny [45] pre-trained on ImageNet [15] as an off-the-shelf surrogate model, which is publicly available on PyTorch website [53]. We set total iterations $N = 8$, tuning epochs $E = 60$, perturbation iterations $n = 40$, perturbation limit $\varepsilon = 2/255$, perturbation stepsize $\alpha = 0.5/255$. As for backdoor configurations, we assume that the total number of D_{bd} and D_{cm} is less than 5% of the total training data, while the ratio of camouflage samples to backdoor samples is no greater than 2. That means $|D_{cm}| + |D_{bd}| < 0.05 * |D_{trn}|$ and $|D_{cm}|/|D_{bd}| \leq 2$ if without any explanations. We compare UBA-Inf with related work BAMU [78], which is implemented based on BadNets for retraining based unlearning.

5.2 Effectiveness Evaluation

Unlike conventional backdoor attacks, UBA-Inf maintains a low ASR in a camouflage state and is expected to activate the backdoor by machine unlearning for fine-grained control. In brief, camouflage samples in UBA-Inf are used to enhance backdoor stealthiness and robustness, improve persistence in continual learning, and provide fine-grained backdoor activation control. Therefore, we will evaluate the camouflage effectiveness first and evaluate backdoor effectiveness in various unlearning scenarios as shown in Table 2.

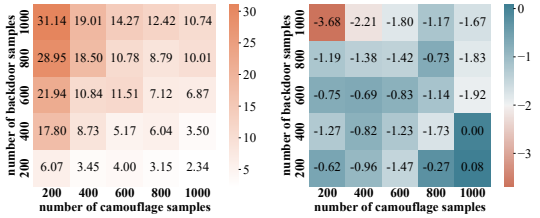

Figure 5: ASR (left figure) and NC anomaly index (right figure) of UBA-Inf for different numbers of backdoor and camouflage samples. Backdoors are detected by NC with an anomaly index below -2 [67].

We first evaluate camouflage effectiveness by checking the ASR of UBA-Inf with BadNets (left of Figure 5). We train CIFAR-10 models in 25 settings (5 amounts of backdoor samples times 5 amounts of camouflage samples). The ASR remains at 31.94% even with 5 times more backdoor samples. By increasing the camouflage sample amount, UBA-Inf can achieve ASR as low as 2.34%.

NC [67] detects backdoors with an anomaly index below -2. The right of Figure 5 shows UBA-Inf protects backdoors from detection with enough camouflage samples. Models with an ASR below 30% typically evade NC detection with an anomaly index greater than -2. We can conclude that UBA-Inf has better camouflage effectiveness and attack performance than existing work like [78].

5.2.1 UBA-Inf Activation Effectiveness in FA-MLaaS

In FA-MLaaS, all training samples are accessible to the SP for unlearning. As indicated in Table 2, we can evaluate UBA-Inf through multiple unlearning algorithms in this setting, including full retrain, SISA, and PUMA[2].

Table 3 and 4 show the results of BA and ASR before and after the camouflage samples are removed by full retraining and SISA [9], respectively. Different backdoor attacks are evaluated. The ASR varies from 0.02% to 29.42% when

[2]The injection rate of evaluations can be found in Appendix A. We suppose all samples are exactly unlearned at once. Appendix D demonstrates evaluations on different unlearning batch sizes.

Table 3: Backdoor effectiveness evaluation for full retrain. Italic numbers with underlines indicate backdoor ASR after unlearning.

Models		BadNets		Blended		LC		Sig	
		BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)
CIFAR-10									
PARN-18	conceal	93.26	**21.94**	93.54	**23.52**	93.48	**21.57**	93.67	**8.61**
	unlearn	93.48	***100.00***	93.79	***91.66***	93.45	***91.81***	93.50	***89.59***
ResNet-34	conceal	93.47	**22.10**	93.64	**21.06**	93.39	**20.31**	93.74	**6.68**
	unlearn	94.23	***100.00***	94.18	***95.30***	94.35	***94.67***	94.19	***91.71***
VGG-16	conceal	90.71	**22.24**	90.54	**20.52**	90.41	**19.60**	90.59	**10.37**
	unlearn	90.62	***100.00***	90.27	***89.64***	90.62	***90.39***	90.17	***87.60***
MNIST									
PARN-18	conceal	99.50	**29.42**	99.74	**24.78**	99.64	**0.13**	99.63	**0.23**
	unlearn	99.64	***100.00***	99.72	***100.00***	99.51	***99.31***	99.55	***99.68***
GTSRB									
PARN-18	conceal	98.34	**22.15**	98.49	**20.59**	98.51	**0.02**	98.32	**2.79**
	unlearn	97.85	***99.89***	98.39	***96.35***	98.26	***5.15***†	98.30	***79.04***
Tiny									
PARN-18	conceal	55.56	**16.57**	55.53	**18.99**	56.02	**2.15**	55.91	**10.44**
	unlearn	56.09	***92.26***	55.83	***90.03***	56.05	***28.11***†	55.69	***90.72***

† LC does not work properly on GTSRB and Tiny due to its limitations. To avoid such a situation, the UBA-Inf adversary can choose a proper backdoor attack alternatively.

backdoors are concealed. After unlearning, backdoors are activated, and ASR rises sharply to nearly 100% in most cases.

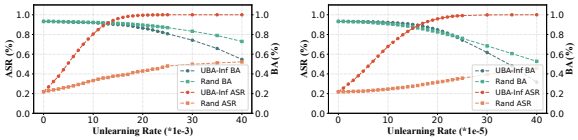

Figure 6: Unlearn UBA-Inf camouflage (UBA-Inf) and Random clean samples (Rand) by PUMA [70] (left figure) and GBU (right figure) [69] with different unlearning rate on a CIFAR-10 model.

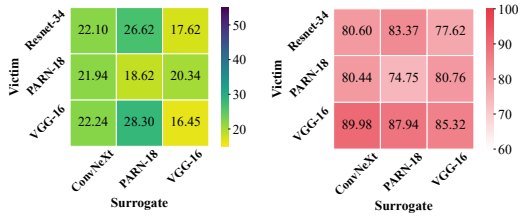

Figure 7: The ASR before unlearning (left figure) and after PUMA unlearning (right figure) with different UBA-Inf surrogate models and target models.

The evaluation results of UBA-Inf with PUMA are shown in Table 5. We find that for all cases after unlearning, the ASR rises above 80% with BA dropping less than 4%. This indicates that UBA-Inf can suit PUMA very well. Unlike full retrain and SISA, the unlearning rate in PUMA is an important hyper-parameter since it can affect PUMA performance significantly. Thus, we evaluate the effects of unlearning rates on BA and ASR for PUMA in the left of Figure 6. By observing the effects on BA when unlearning clean samples, we can tell that the unlearning rate should not exceed 3e-2, in which case ASR can keep above 90% if camouflage samples are unlearned.

5.2.2 UBA-Inf Activation Effectiveness in RA-MLaaS

In RA-MLaaS, the access to original training data during unlearning is highly restricted [29], and only data to be removed is available. In RA-MLaaS, GBU [69][3] is a gradient-based unlearning algorithm. By adopting GBU in RA-MLaaS, the data to be removed will be replaced by perturbed data for unlearning. The results of UBA-Inf with GBU are shown in Table 6. Similar to the results of UBA-Inf with PUMA, unlearning through GBU makes ASR increase from around 22% to above 80%, with BA dropping less than 4% in all cases. The right of Figure 6 presents the effects of unlearning rates for UBA-Inf with GBU. When the unlearning rate is no larger than 1e-4, unlearning clean samples has a negligible effect on BA, while unlearning camouflage samples can increase ASR to around 70%.

5.2.3 Effectiveness Comparisons with BAMU

Like UBA-Inf, BAMU [78] uses unlearning camouflage samples to activate backdoors. However, Table 7 shows UBA-Inf surpasses BAMU in camouflage effectiveness and activation effectiveness. With the same injection rate, UBA-Inf's ASR is at least 10% lower than BAMU's before unlearning, showing that UBA-Inf can provide better camouflage effectiveness. After exact unlearning like full retraining, the ASR after unlearning largely depends on the backdoor trigger instead of the camouflage-generating algorithm, so the ASR is similar for both methods. Meanwhile, BAMU doesn't support approximate unlearning algorithms, so we reproduced it with PUMA and GBU following its original idea. After unlearning using PUMA [70] and GBU [69], UBA-Inf's ASR is about 20% higher than BAMU's, indicating UBA-Inf's adaptability and superior attack performance in different unlearning strategies.

5.2.4 Evaluation of the Surrogate Model

We evaluate UBA-Inf with different surrogate model settings. Figure 7 demonstrates attack results of UBA-Inf on CIFAR-10 with different surrogate models and different victim models. The ASR before unlearning keeps lower than

[3] Here we only evaluate on the first-order GBU [69]. We further evaluate the second-order GBU [69] in Appendix B.

Table 4: Backdoor effectiveness evaluation for SISA. Two different numbers of training data shards are considered.

Shards		BadNets[1]		Blended[2]		LC[3]		Sig[4]	
		BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)
CIFAR-10									
shard=3	conceal	90.76	**12.26**	90.62	**22.72**	90.43	**23.54**	90.96	**9.24**
	unlearn	90.65	***99.98***	90.26	***89.92***	90.30	***88.65***	90.95	***89.42***
shard=5	conceal	88.74	**17.01**	88.30	**22.88**	88.62	**27.12**	88.82	**17.50**
	unlearn	88.68	***99.94***	88.59	***91.82***	88.11	***88.00***	88.66	***96.36***
MNIST									
shard=3	conceal	99.58	**6.58**	99.70	**25.03**	99.66	**0.28**	99.63	**0.38**
	unlearn	99.66	***100.00***	99.66	***100.00***	99.65	***73.50***	99.68	***65.35***
shard=5	conceal	99.64	**1.90**	99.67	**18.33**	99.56	**0.35**	99.56	**0.48**
	unlearn	98.57	***100.00***	99.67	***100.00***	99.53	***54.03***†	99.49	***34.66***†
GTSRB									
shard=3	conceal	99.59	**23.31**	98.36	**24.32**	98.23	**0.03**	98.32	**5.48**
	unlearn	99.61	***100.00***	98.50	***88.86***	98.24	***4.61***†	98.13	***72.30***
shard=5	conceal	99.59	**15.21**	97.98	**24.60**	98.27	**0.03**	98.01	**10.01**
	unlearn	99.58	***100.00***	97.96	***83.24***	97.41	***3.15***†	97.76	***69.58***
Tiny									
shard=3	conceal	51.47	**20.60**	51.38	**20.12**	52.03	**3.23**	51.81	**10.25**
	unlearn	51.40	***87.73***	52.15	***82.27***	51.45	***47.35***†	51.73	***79.66***
shard=5	conceal	48.36	**24.60**	47.91	**16.46**	48.12	**5.83**	48.36	**9.35**
	unlearn	47.63	***82.47***	48.06	***85.21***	48.02	***32.75***†	47.45	***79.23***

† Similar to full retrain, LC does not work properly on GTSRB and Tiny, while Sig has problems with SISA on MNIST. To avoid such a situation, the UBA-Inf adversary can choose a proper backdoor attack alternatively.

Table 5: Backdoor effectiveness evaluation for PUMA.

Dataset	Models	conceal		unlearn	
		BA(%)	ASR(%)	BA (%)	ASR(%)
CIFAR-10	PARN-18	93.26	**21.94**	89.50	***80.44***
	ResNet-34	93.47	**22.10**	89.91	***80.60***
	VGG-16	90.71	**22.24**	89.52	***89.68***
MNIST	PARN-18	99.50	**29.42**	98.27	***81.51***
GTSRB	PARN-18	98.34	**22.15**	98.19	***81.46***
Tiny	PARN-18	55.56	**16.57**	50.06	***71.72***

Table 6: Backdoor effectiveness evaluation for GBU.

Datasets	Models	conceal		unlearn	
		BA(%)	ASR(%)	BA(%)	ASR(%)
CIFAR-10	PARN-18	93.26	**21.94**	90.53	***83.60***
	ResNet-34	93.47	**22.10**	90.19	***86.25***
	VGG-16	90.71	**22.24**	89.28	***89.96***
MNIST	PARN-18	99.50	**29.42**	98.28	***89.01***
GTSRB	PARN-18	98.34	**22.15**	95.18	***80.20***
Tiny	PARN-18	55.56	**16.57**	49.98	***64.26***

Table 7: The comparison between the camouflage and activation effectiveness of BAMU [78] and UBA-Inf.

State	Method	CIFAR-10		MNIST		GTSRB		Tiny	
		BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)
before unlearn	UBA-Inf	93.26	**21.94**	99.50	**29.42**	98.34	**22.15**	55.56	**16.57**
	BAMU	93.19	36.71	99.47	90.14†	98.51	28.44	56.20	37.95
after full retrain	UBA-Inf	93.34	**100.00**	99.64	**100.00**	97.85	**99.89**	56.09	**92.26**
	BAMU	93.12	100.00	99.58	100.00†	98.23	99.63	55.90	88.73
after PUMA	UBA-Inf	89.50	**80.44**	98.27	**81.51**	98.27	**81.51**	50.06	**71.72**
	BAMU	89.97	50.10	98.39	99.93†	94.90	64.13	50.02	56.21
after GBU	UBA-Inf	90.53	**83.60**	98.28	**89.01**	95.18	**80.20**	49.98	**64.26**
	BAMU	90.11	52.53	98.47	92.49†	94.82	59.71	50.24	47.15

† BAMU fails in MNIST with ASR higher than 80%, which completely has no camouflage effect.

30% while rising to at least around 75% after PUMA unlearning, which means UBA-Inf achieves satisfactory camouflage effectiveness and activation effectiveness under different surrogate model settings. In other words, the attack performance of UBA-Inf is not depended on a specific surrogate model setting.

5.3 Stealthiness Evaluation

Having noticed backdoor threats, the SP deploys backdoor defenses to protect MLaaS from suspicious backdoor samples, including outlier filter defenses [13, 25, 38] and model scanner defenses [23, 62, 67]. Furthermore, recent studies provide the SP with defenses against UBA [78] in particular. We will demonstrate the stealthiness of our UBA-Inf by evaluating the deceiving result of conventional backdoor defenses and UBA defenses in comparison with BAMU [78]. In previous experiments, we have evaluated different backdoor generation algorithms on multiple datasets, from which we can conclude that BadNets is suitable for further evaluation as a typical generation method to simplify the discussion.

5.3.1 Evaluation of Outlier Filter Defense

Outlier filters defend MLaaS against backdoor attacks by filtering out suspicious samples in the training and inference processes. Three popular outlier filters, Spectre [25], D-BR [13], and ABL [38], are used for evaluating the stealthiness of camouflage samples in UBA-Inf. We use outlier filters to detect suspicious samples and present the result of poisoning samples in Figure 8. It shows that Spectre, ABL (with isolation rate equal to 10%), and D-BR filter about half of and almost backdoor samples generated by naive BadNets, respectively. However, only about 10% UBA-Inf backdoor samples under camouflage are correctly detected by Spectre and ABL. In the worst case, D-BR detects about 30% UBA-Inf backdoor samples on GTSRB. Even so, the rest samples are enough for the attack according to the effectiveness evaluation. We find that UBA-Inf's ASR is around 20% before unlearning and rises to over 85% after unlearning under outlier filters like Spectre and D-BR. UBA-Inf also protects more poisoning samples from detection than BAMU [78], providing better stealthiness.

5.3.2 Evaluation of Model Scanner Defense

Model scanners verify whether the model has potential backdoor threats and attempt to reverse-engineer the backdoor trigger. Three popular model scanner defenses and corresponding anomaly indexes are used for the evaluation, i.e.,

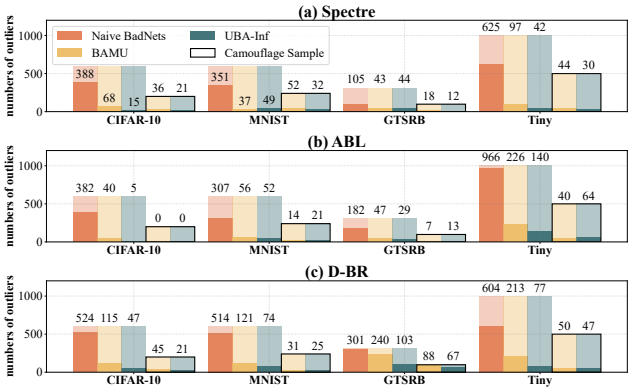

Figure 8: Filtered suspicious samples in backdoor and camouflage by Spectre, ABL, and D-BR. Bars without and with a border represent backdoor and camouflage samples, respectively. Bars in light and dark colors indicate all and detected samples, respectively.

NC [67], PB [62], and AEVA [23]. SP can employ model scanners before and after unlearning. However, frequent scanning after each unlearning request is overwhelmingly time-consuming.

Figure 9 demonstrates different anomaly indexes of one target class before unlearning. The more negative the anomaly indexes, the more likely a class has backdoors. With naive BadNets, all three scanners detect the backdoor in the target class with highly negative anomaly indexes. However, with UBA-Inf camouflage, these indexes become less negative or even positive. Using the -2.0 threshold from [67], none of the scanners can detect the backdoor injected by UBA-Inf, while BAMU camouflages still get detected. We have only listed the target class because other classes are irrelevant to the backdoor attack. In Figure 10, we show an example of reverse-engineering a backdoor trigger using NC, from which we can easily conclude that UBA-Inf camouflage protects backdoor triggers from being revered effectively compared with naive BadNets and BAMU.

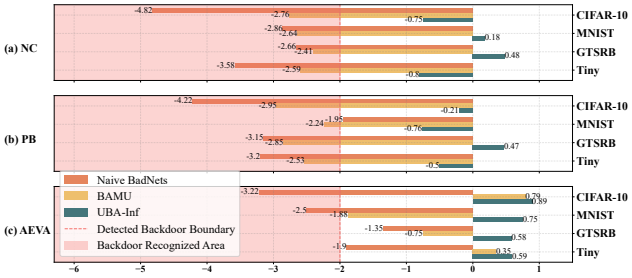

Figure 9: Anomaly indexes of one class (the 6th class in all datasets) calculated by NC, PB, and AEVA before unlearning.

When SP scans the model after unlearning, UBA-Inf with BadNets may be detected by backdoor scanners. However, UBA-Inf offers better flexibility with diverse backdoor generating algorithms and can evade detection by employing appropriate methods like Narcissus [75]. We compared BAMU with UBA-Inf using BadNets and UBA-Inf using Narcissus.

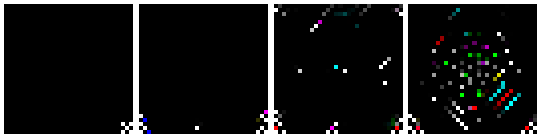

Figure 10: From left to right is the true trigger and the reversed trigger of naive BadNets, BAMU [78] and UBA-Inf, respectively. The camouflage of UBA-Inf is more effective.

Figure 11 demonstrates that BadNets-based UBA-Inf is vulnerable to post-unlearning defenses, UBA-Inf with Narcissus effectively eludes these scanners with anomaly indexes above -0.67 or even positive. This suggests UBA-Inf's resilience to model scanning persists even after unlearning, leveraging suitable backdoor generating methods.

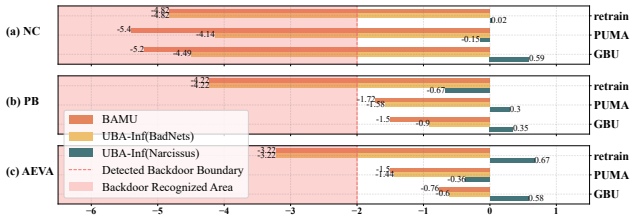

Figure 11: Anomaly indexes of one class (the 6th class in CIFAR-10 [35]) calculated by NC, PB, and AEVA after full retrain, PUMA [70] and GBU [69] unlearning.

5.3.3 Evaluation of UBA Defense

In a recent study [78], defenses dedicated to UBA have been proposed. In particular, model uncertainty (MU) for deep learning models and sub-model similarity (SMS) for SISA models are calculated to distinguish malicious samples from clean ones and reject suspicious unlearning requests[4]. Figure 12 indicates that MU fails to distinguish UBA-Inf samples from clean samples on all 4 datasets because the Gini impurity distribution of malicious samples generated by UBA-Inf is very similar to that of clean samples, which makes it impossible to define a proper threshold for identifying malicious samples. Table 8 shows UBA-Inf effectiveness when considering stealthiness under MU defense. If MU allows 10% false negative, which means 90% clean samples can pass MU defense, a large portion of malicious samples can also escape the defense and increase the ASR to over 80% in most cases through unlearning. Meanwhile, MU defenses cast more BAMU camouflage samples with the same 10% false negative. After unlearning, BAMU reaches lower ASRs against MU defenses, which is around 60% in most cases. This concludes that UBA-Inf is more stealthy and effective compared with BAMU, and MU cannot prevent UBA-Inf effectively.

Experimental results of SMS defense are given in Figure 13, showing the variance of the predicted value of SISA sub-models. On CIFAR-10 and Tiny, the distribution of UBA-Inf malicious samples looks very similar to the one of clean

[4]Here we only evaluate UBA-Inf against the MU defense and the SMS defense. More defenses designed for other unlearning vulnerabilities are represented in Appendix C.

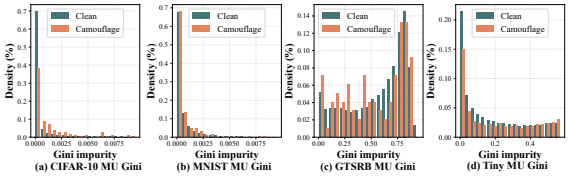

Figure 12: MU measured by Gini impurity on different datasets.

Table 8: UBA-Inf evaluation under MU defense.

Datasets	UBA-Inf without MU		UBA-Inf with MU		BAMU with MU	
	BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)
full retrain						
CIFAR-10	93.48	100.00	92.75	**73.20**	93.15	42.34
MNIST	99.64	100.00	99.54	**90.21**	99.47	90.14†
GTSRB	97.85	99.89	98.08	**86.92**	98.32	72.46
Tiny	56.09	92.26	55.94	**81.38**	55.47	62.21
PUMA						
CIFAR-10	89.50	80.44	89.50	**80.44**	89.20	64.28
MNIST	98.27	81.51	98.27	**74.59**	99.47	90.14†
GTSRB	96.19	81.46	96.19	**81.45**	95.21	50.36
Tiny	50.06	71.72	50.14	**63.22**	50.15	38.56
GBU						
CIFAR-10	90.03	80.04	90.33	**77.00**	90.12	63.47
MNIST	98.28	89.01	98.70	**84.71**	99.47	90.14†
GTSRB	95.18	80.20	95.96	**77.54**	95.25	61.93
Tiny	49.98	64.26	49.35	**52.78**	48.62	40.17

† Please note that BAMU does not have a masking effect on the MNIST dataset.

samples. Thus, most malicious samples generated by UBA-Inf can escape SMS defense when considering a reasonable false negative ratio. However, UBA-Inf malicious samples on MNIST and GTSRB can be effectively distinguished by SMS, which means most malicious samples generated by UBA-Inf will be rejected by SMS. Table 9 gives UBA-Inf effectiveness when considering stealthiness under SMS defense. The backdoor can be successfully activated and exploited on CIFAR-10 and Tiny while probabilistically activated on MNIST and GTSRB when using 5 shards for SISA. In the worst case, UBA-Inf can hardly succeed on GTSRB shen using 3 shards for SISA because most unlearning requests are rejected by SMS. We have observed that UBA-Inf effectiveness under SMS defense gets better when the shard number increases. Actually, practical applications commonly use more shards way beyond demo cases, which means UBA-Inf can cause practical threats when SISA uses more than 5 shards.

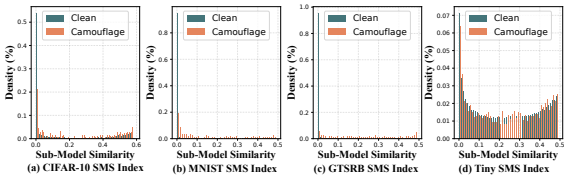

Figure 13: SMS measured by sub-model prediction variance for 3-shard SISA.

Table 9: UBA-Inf evaluation under SMS defense.

Shards		CIFAR-10		MNIST		GTSRB		Tiny	
		BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)
shard=3	without SMS	90.65	99.98	99.66	100.00	95.58	100.00	51.40	87.73
	with SMS	91.55	**92.09**	98.28	**23.00**	98.68	**1.68**	51.67	**77.23**
shard=5	without SMS	88.68	99.94	98.57	100.00	99.58	100.00	47.63	82.47
	with SMS	89.06	**90.99**	99.64	**59.86**	99.67	**55.40**	48.23	**76.45**

5.4 Evaluation of Resistance to Reconstruction

Model sanitizing or transforming modifies model parameters to eliminate potential backdoor threats. Such model reconstruction may affect the activation of UBA-Inf. Three model reconstructors are used for UBA-Inf resistance evaluation, i.e., naive fine-tuning (FT), fine-pruning (FP) [40], and neural attention distillation (NAD) [39]. Please note that since exact unlearning algorithms retrain model parameters on contaminated datasets, they invalidate previous model reconstruction defenses. Thus, we will focus on the evaluation of approximate unlearning like PUMA and GBU. Like model scanning, SP can employ a model reconstructor before or after model updating. However, reconstruction defenses after each unlearning inevitably bring significant time overheads.

Table 10: UBA-Inf evaluation after model reconstruction.

Defenses	before unlearn		PUMA unlearn		GBU unlearn	
	BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)
CIFAR-10						
FT	93.28	**8.18**	85.62	***80.44***	85.71	***80.95***
FP	93.18	**5.00**	85.53	***72.68***	86.44	***83.13***
NAD	92.87	**14.87**	86.62	***70.60***	88.06	***87.54***
MNIST						
FT	99.67	**11.05**	99.01	***77.23***	99.09	***89.12***
FP	99.59	**3.49**	98.77	***62.87***	99.00	***99.56***
NAD	99.62	**17.09**	98.59	***79.17***	98.92	***90.46***
GTSRB						
FT	98.20	**11.45**	95.13	***76.93***	95.39	***71.51***
FP	98.31	**9.29**	95.19	***81.57***	95.09	***70.73***
NAD	98.09	**9.80**	95.37	***88.92***	95.38	***65.31***
Tiny						
FT	55.26	**9.12**	50.16	***40.15***	50.01	***43.29***
FP	55.14	**8.54**	50.02	***42.15***	49.95	***45.16***
NAD	55.25	**10.25**	50.11	***44.74***	50.03	***41.63***

Table 10 gives the results of UBA-Inf effectiveness after model reconstruction before unlearning, from which we can tell that model reconstruction certainly increases the robustness of the target model, lowering ASR of UBA-Inf than without reconstructing, as shown in Table 5 and Table 6. Even so, UBA-Inf gives satisfying ASRs on CIFAR-10, MNIST, and GTSRB. Even in the worst case of Tiny, the ASR of UBA-Inf exceeds 40%, still posing high backdoor risks.

Table 11: Backdoor evaluations of BAMU, UBA-Inf(BadNets), and UBA-Inf(Narcissus) against FT, FP, and NAD defenses after full retrain, PUMA, GBU unlearning, respectively.

Method	Unlearn algorithm	After unlearn		FT after unlearn		FP after unlearn		NAD after unlearn	
		BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)
BAMU	full retrain	93.48	100.00	91.89	10.14	91.83	7.86	92.59	9.24
	PUMA	89.50	50.10	91.88	11.49	91.58	5.02	91.81	7.74
	GBU	90.11	52.53	91.90	11.71	91.56	5.16	90.89	7.52
UBA-Inf (BadNets)	full retrain	93.48	100.00	91.89	10.14	91.83	7.86	92.59	8.24
	PUMA	89.50	80.44	91.88	11.63	91.34	5.25	90.71	8.02
	GBU	90.53	83.60	90.74	10.24	91.05	5.19	90.92	7.93
UBA-Inf (Narcissus)	full retrain	93.43	96.29	92.84	90.45	93.01	88.80	93.34	83.50
	PUMA	90.32	97.84	93.82	53.38	93.70	51.96	93.65	55.46
	GBU	90.77	84.52	93.84	53.61	93.74	54.61	93.62	50.90

As discussed in model scanning, when SP will perform defenses at any cost after unlearning, UBA-Inf using BadNets can be mitigated. In this case, UBA-Inf can use suitable triggers like Narcissus to resist post-unlearning model reconstruction. We compared BAMU with UBA-Inf using BadNets and UBA-Inf using Narcissus. Table 11 shows UBA-Inf's resilience to defenses after unlearning. While BAMU's ASR drops to around 10%, UBA-Inf maintains at least 50% ASR by switching the trigger from BadNet to Narcissus. For models with full retraining, UBA-Inf's ASR stays above 80% with Narcissus. This suggests that UBA-Inf can withstand model

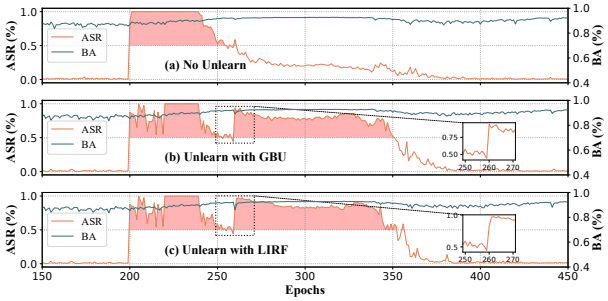

Figure 14: Persistence evaluation on CIFAR-10.

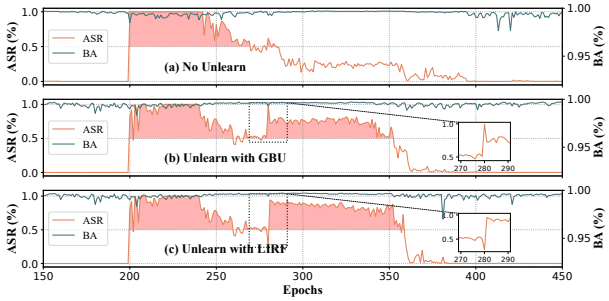

Figure 15: Persistence evaluation on Rotated-MNIST.

reconstruction even after unlearning when leveraging proper backdoor triggers.

5.5 Persistence Evaluation

Unlike OT-MLaaS, CT-MLaaS continuously updates the on-cloud model with newly coming training data [58, 59]. Thus, we should consider the persistence of the injected backdoor. The adversary of UBA-Inf expects the injected backdoor to keep away from backdoor vanishing caused by catastrophic forgetting [68]. We will evaluate the backdoor persistence after being activated using two unlearning methods as indicated in Table 2, i.e., GBU [69] and LIRF [74].

Assuming new training data comes in tasks, we use CIFAR-10 for continuous learning in the same domain [52], and Rotated-MNIST [46] for domain incremental learning where training data in different tasks comes from different domains [68, 82]. In either case, we train a PRAN-18 model with clean samples for 10 tasks (200 epochs) before injecting malicious samples and with malicious samples for 2 tasks (40 epochs), after which the model will learn continuously with only clean samples. For naive BadNets, the adversary injects 400 backdoor samples in 2 tasks (epoch 200 to 240). As for UBA-Inf, 400 backdoor samples are injected in the same way, while 200 camouflage samples will also be injected from epoch 200 to 220. UBA-Inf achieves backdoor activation through unlearning instead of learning. Hence, the criteria is timing for unlearning. Better ASR is achieved with earlier unlearning before catastrophic forgetting. UBA-Inf can significantly prolong the backdoor effect even if unlearning occurs when the ASR drops to 50%, which corresponds to epoch 260 for CIFAR-10 and epoch 280 for Rotated-MNIST.

The first row of Figure 14 and Figure 15 demonstrates the ASR trends of naive BadNets in continuous learning on CIFAR-10 and Rotated-MNIST respectively, which drop below 50% in about 20 epochs after backdoor samples have been injected. The last two rows of Figure 14 and Figure 15 show the ASR of UBA-Inf with GBU unlearning and the ASR of UBA-Inf with LIRF unlearning, respectively. ASRs of the last two rows share a similar dropping to below 50% before unlearning, just like the naive BadNets. However, when the adversary activates the backdoor through unlearning camouflage samples, ASRs of UBA-Inf rise to a new peak immediately. Moreover, ASRs of UBA-Inf remain around 70% to 80% and drop below 50% after 80 epochs, which significantly extends the persistence of the backdoor's effect. Compared with the conventional backdoor attack, UBA-Inf enhances the backdoor persistence after activation by about 4 times.

6 Related work

6.1 Backdoor Attacks and Defenses

Neural backdoor attack embeds backdoors into DNNs to induce twisted predictions evoked by the trigger pattern. [44]. BadNets [21] leverages a small patch as the trigger pattern. LC [66] combines BadNets with adversarial perturbations to perform a clean-label attack effectively. Blended [14] directly blends a cartoon image or a random pattern with the input. Sig [65] injects a sinusoidal signal pattern on images, which is a strip-like pattern. Narcissus [75] optimizes the trigger pattern in a way that points towards the inside of the target class. More backdoor generating methods have been proposed for better effectiveness and stealthiness [37, 76].

Recent works have shown the worrying threat of more complicated backdoor attack workflows. For example, latent backdoor attacks conceal backdoors initially but then activate it with fine-tuning [50] or exact machine unlearning [78]. The comparison between recent backdoor activation approaches is given in Table 12. Existing approaches either require full control of the training process for poisoning [32, 50] or depend on replayed backdoor injection [82]. Meanwhile, recent works also expose the backdoor vanishing problem in continuous learning [32, 82]. However, UBA-Inf can achieve the same adversarial goals under more restrictions, including continuous learning, making it a more disturbing attack on MLaaS.

Various defenses have been proposed against backdoor attacks. Generally, backdoor defenses can be categorized as 1) *model scanners* [23, 62, 67], which conduct reverse engineering to decide whether a certain class has been poisoned and identify the potential backdoor triggers, 2) *outlier filters* [13, 25, 38], which detect malicious training data and gives them up during training or fine-tunes the model to mitigate backdoor after training; and 3) *model reconstructors* [39, 40, 81], which reconstruct the victim model through

Table 12: Different properties and applicable scenarios of existing latent backdoor approaches.

	Properties			Scenarios			
	Is it a backdoor attack?	*Does it need access to the training process?*	*What is the activation mechanism?*	*Edge-Device Services*	*Full-Access MLaaS*	*Restricted-Access MLaaS*	*Continuous Training*
Hibernated Backdoor [50]	✓	✓	Fine-tune	✓	-	-	-
Hidden Poison [16]	-	✓	Unlearning	-	✓	-	-
Incremental Backdoor [32]	✓	✓	Incremental Learning	-	-	-	✓
BAMU [78]	✓	-	Unlearning	-	✓	-	-
UBA-Inf (Ours)	✓	-	Unlearning	-	✓	✓	✓

methods like neuron pruning [40] or channel shuffling [81]. We will show that UBA-Inf in the camouflage state is immune to these defense solutions, calling for more robust backdoor defenses and more secure MLaaS mechanisms.

6.2 Machine Unlearning

Machine unlearning aims to remove sensitive samples from the MLaaS model [72]. Intuitively, full retraining is the most naive solution, but it costs too much time and resources [29].

Exact machine unlearning is based on model retraining. SISA [9] reorganizes the training data in sharding and slicing to reduce the execution time of retraining. It aggregates the results of sub-models as the final result. Many exact unlearning solutions inherit SISA. ARCANE [73] designed an efficient architecture for exact unlearning. A study [10] introduced SISA into recommendation systems. Another work [12] adopted SISA to graph data. All of these exact unlearning algorithms require full access to the training dataset.

Approximate unlearning algorithms exploits feasible ways to speed up the unlearning process [72]. Some works exploit the second-order Hessian matrix, which requires full access to training data for unlearning. For instance, PUMA [70] relies on influence functions [33] and re-weighting methods to remove data without compromising model performance. Some works exploit first-order gradients [63, 69] or tailored loss functions [74] instead. Gradient-Based Unlearning (GBU) [69] replaces a data sample with the adversarially perturbed one through first-order or second-order gradient-decent optimization. Fine-tuning has been widely used as an empirical unlearning baseline in existing works [19, 63, 72]. Besides, LIRF [74] is tailored to recoverable knowledge removal in continual learning. It assumes that the upper layers have completely transferred to new knowledge and only modifies the lower layers through a tailored loss function combined with the cross-entropy loss and the filtered attention distillation loss on randomly labeled samples.

Recent studies have exposed vulnerabilities in machine unlearning, such as slowdowns caused by malicious data removal [49], degradation in model performance on clean data [29, 54], and latent attacks that confuse defenders [16, 78]. Some solutions have been proposed to counter malicious unlearning. For instance, Zhang et al. [78] utilized Gini impurity and sub-model similarity to identify malicious unlearning, assuming that malicious samples exhibit larger prediction uncertainty and poor generalization across sub-models. Additionally, other defenses like hashing [29] or medoid optimization [54] have been explored. However, we demonstrate that our UBA-Inf method can bypass these defenses, underscoring the need for more robust and effective unlearning defenses.

7 Conclusion

Our work presents a novel approach to implementing effective and stealthy backdoor attacks on MLaaS platforms using machine unlearning. We introduce an influence-enhanced unlearning-activated backdoor attack method, which can not only strengthen backdoor robustness but also improve backdoor persistence in continuous learning, thus shedding light on the vulnerabilities of current unlearning procedures in MLaaS contexts. Through extensive evaluations, we demonstrate UBA-Inf's resistance to defenses across various MLaaS scenarios, achieving high attack success rates with improved stealthiness and robustness compared to existing approaches.

Note that our work gives an alternative backdoor approach with fine-grained control of backdoor activation. However, dedicated backdoor designs are orthogonal to our work, which means this approach can also be applied to advanced backdoor triggers in the future. Meanwhile, we should also note that novel defenses may mitigate UBA-Inf in the future. In this case, the adversary could take novel defenses into camouflage optimization process as constraints if defense strategies are known. Moreover, UBA is an example of leveraging unlearning for attacks. Other types of attacks, like poisoning and inference, may be of interest in future work.

Acknowledgement

The authors would like to thank the shepherd and anonymous reviewers for the time and efforts they have kindly made in this paper. This work was supported in part by the National Natural Science Foundation of China under Grants NSFC-62272222, NSFC-61902176, NSFC-62272215, and in part by the Jiangsu Natural Science Foundation Excellent Youth Project under Grant BK20230080.

References

[1] Ai and machine learning - google cloud. https://cloud.google.com/products/ai. Accessed: 2024-1-30.

[2] Azure ai services - microsoft azure. https://azure.microsoft.com/en-us/products/ai-services. Accessed: 2024-1-30.

[3] Here's how to stop meta from using your data for ai training. https://interestingengineering.com/culture/heres-how-to-stop-meta-from-using-your-data-for-ai-training. Accessed: 2024-2-3.

[4] Machine learning service - amazon sagemaker. https://aws.amazon.com/sagemaker. Accessed: 2024-1-30.

[5] Openai privacy request portal. https://privacy.openai.com/policies?name=open-ai-privacy-request-portal#privacy-practices. Accessed: 2024-2-3.

[6] User deletion api - overview. https://developers.google.com/analytics/devguides/config/userdeletion/v3. Accessed: 2024-2-3.

[7] Regulation 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union 119* (2016), 1–88.

[8] BOURTOULE, L., CHANDRASEKARAN, V., CHOQUETTE-CHOO, C., JIA, H., TRAVERS, A., ZHANG, B., LIE, D., AND PAPERNOT, N. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)* (2021), pp. 141–159.

[9] BOURTOULE, L., CHANDRASEKARAN, V., CHOQUETTE-CHOO, C. A., JIA, H., TRAVERS, A., ZHANG, B., LIE, D., AND PAPERNOT, N. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)* (2021), IEEE, pp. 141–159.

[10] CHEN, C., SUN, F., ZHANG, M., AND DING, B. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022* (2022), pp. 2768–2777.

[11] CHEN, H., CHEN, H. H., SUN, M., LI, K., CHEN, Z., AND WANG, X. A verified confidential computing as a service framework for privacy preservation. In *32nd USENIX Security Symposium (USENIX Security 23)* (2023), pp. 4733–4750.

[12] CHEN, M., ZHANG, Z., WANG, T., BACKES, M., HUMBERT, M., AND ZHANG, Y. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (2022), pp. 499–513.

[13] CHEN, W., WU, B., AND WANG, H. Effective backdoor defense by exploiting sensitivity of poisoned samples. *Advances in Neural Information Processing Systems 35* (2022), 9727–9737.

[14] CHEN, X., LIU, C., LI, B., LU, K., AND SONG, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[15] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009), 248–255.

[16] DI, J. Z., DOUGLAS, J., ACHARYA, J., KAMATH, G., AND SEKHARI, A. Hidden poison: Machine unlearning enables camouflaged poisoning attacks. In *NeurIPS ML Safety Workshop* (2022).

[17] DOSOVITSKIY, A., BEYER, L., KOLESNIKOV, A., WEISSENBORN, D., ZHAI, X., UNTERTHINER, T., DEHGHANI, M., MINDERER, M., HEIGOLD, G., GELLY, S., USZKOREIT, J., AND HOULSBY, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations* (2021).

[18] GONG, X., CHEN, Y., YANG, W., WANG, Q., GU, Y., HUANG, H., AND SHEN, C. Redeem myself: Purifying backdoors in deep learning models using self attention distillation. In *2023 IEEE Symposium on Security and Privacy (SP)* (2023), pp. 755–772.

[19] GRAVES, L., NAGISETTY, V., AND GANESH, V. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), vol. 35, pp. 11516–11524.

[20] GROUP, I. Machine learning as a service (mlaas) market, 2023.

[21] GU, T., LIU, K., DOLAN-GAVITT, B., AND GARG, S. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access 7* (2019), 47230–47244.

[22] GUO, C., GOLDSTEIN, T., HANNUN, A., AND VAN DER MAATEN, L. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning* (2020).

[23] GUO, J., LI, A., AND LIU, C. AEVA: Black-box backdoor detection using adversarial extreme value analysis. In *International Conference on Learning Representations* (2022).

[24] GUO, Y., ZHAO, Y., HOU, S., WANG, C., AND JIA, X. Verifying in the dark: Verifiable machine unlearning by using invisible backdoor triggers. *IEEE Transactions on Information Forensics and Security* (2024).

[25] HAYASE, J., KONG, W., SOMANI, R., AND OH, S. Spectre: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning* (2021), PMLR, pp. 4129–4139.

[26] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778.

[27] HE, K., ZHANG, X., REN, S., AND SUN, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)* (2016), pp. 630–645.

[28] HU, H., SALCIC, Z., SUN, L., DOBBIE, G., YU, P. S., AND ZHANG, X. Membership inference attacks on machine learning: A survey. *ACM Comput. Surv. 54*, 11s (sep 2022).

[29] HU, H., WANG, S., CHANG, J., ZHONG, H., SUN, R., HAO, S., ZHU, H., AND XUE, M. A duty to forget, a right to be assured? exposing vulnerabilities in machine unlearning services. *arXiv preprint arXiv:2309.08230* (2023).

[30] HU, J., WANG, Z., SHEN, Y., LIN, B., SUN, P., PANG, X., LIU, J., AND REN, K. Shield against gradient leakage attacks: Adaptive privacy-preserving federated learning. *IEEE/ACM Transactions on Networking* (2023), 1–16.

[31] JIA, J., AND GONG, N. Z. AttriGuard: A practical defense against attribute inference attacks via adversarial machine learning. In *27th USENIX Security Symposium (USENIX Security 18)* (2018), pp. 513–529.

[32] JIANG, W., ZHANG, T., QIU, H., LI, H., AND XU, G. Incremental learning, incremental backdoor threats. *IEEE Transactions on Dependable and Secure Computing* (2022), 1–11.

[33] KOH, P. W., AND LIANG, P. Understanding black-box predictions via influence functions. In *International conference on machine learning* (2017), PMLR, pp. 1885–1894.

[34] KREMER, J., SHA, F., AND IGEL, C. Robust active label correction. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics* (09–11 Apr 2018), A. Storkey and F. Perez-Cruz, Eds., vol. 84 of *Proceedings of Machine Learning Research*, PMLR, pp. 308–316.

[35] KRIZHEVSKY, A., NAIR, V., AND HINTON, G. Cifar-10 (canadian institute for advanced research). Tech. rep., University of Toronto, 2009.

[36] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[37] LI, Y., LI, Y., WU, B., LI, L., HE, R., AND LYU, S. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 16463–16472.

[38] LI, Y., LYU, X., KOREN, N., LYU, L., LI, B., AND MA, X. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems 34* (2021), 14900–14912.

[39] LI, Y., LYU, X., KOREN, N., LYU, L., LI, B., AND MA, X. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *International Conference on Learning Representations* (2021).

[40] LIU, K., DOLAN-GAVITT, B., AND GARG, S. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses* (2018), Springer, pp. 273–294.

[41] LIU, S., AND DENG, W. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (2015), pp. 730–734.

[42] LIU, Y., FAN, M., CHEN, C., LIU, X., MA, Z., WANG, L., AND MA, J. Backdoor defense with machine unlearning. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications* (2022).

[43] LIU, Y., MA, S., AAFER, Y., LEE, W.-C., ZHAI, J., WANG, W., AND ZHANG, X. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)* (2018), Internet Soc.

[44] LIU, Y., MONDAL, A., CHAKRABORTY, A., ZUZAK, M., JACOBSEN, N., XING, D., AND SRIVASTAVA, A. A survey on neural trojans. In *2020 21st International Symposium on Quality Electronic Design (ISQED)* (2020), IEEE, pp. 33–39.

[45] LIU, Z., MAO, H., WU, C.-Y., FEICHTENHOFER, C., DARRELL, T., AND XIE, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 11976–11986.

[46] LOPEZ-PAZ, D., AND RANZATO, M. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017), NIPS'17, Curran Associates Inc., p. 6470–6479.

[47] MAO, Y., XIN, Z., LI, Z., HONG, J., YANG, Q., AND ZHONG, S. Secure split learning against property inference, data reconstruction, and feature space hijacking attacks. In *ESORICS* (2023).

[48] MAO, Y., YUAN, X., ZHAO, X., AND ZHONG, S. Romoa: Robust model aggregation for the resistance of federated learning to model poisoning attacks. In *ESORICS* (2021).

[49] MARCHANT, N. G., RUBINSTEIN, B. I., AND ALFELD, S. Hard to forget: Poisoning attacks on certified machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2022), vol. 36, pp. 7691–7700.

[50] NING, R., LI, J., XIN, C., WU, H., AND WANG, C. Hibernated backdoor: A mutual information empowered backdoor attack to deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2022), vol. 36, pp. 10309–10318.

[51] PANG, R., ZHANG, Z., GAO, X., XI, Z., JI, S., CHENG, P., LUO, X., AND WANG, T. Trojanzoo: Towards unified, holistic, and practical evaluation of neural backdoors. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)* (2022).

[52] PRAPAS, I., DERAKHSHAN, B., MAHDIRAJI, A. R., AND MARKL, V. Continuous training and deployment of deep learning models. *Datenbank-Spektrum 21*, 3 (2021), 203–212.

[53] PYTORCH CONTRIBUTORS. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. PyTorch, 2022.

[54] QIAN, W., ZHAO, C., LE, W., MA, M., AND HUAI, M. 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* (New York, NY, USA, 2023), KDD '23, Association for Computing Machinery, p. 1932–1942.

[55] QIU, P., ZHANG, X., JI, S., FU, C., YANG, X., AND WANG, T. Hashvfl: Defending against data reconstruction attacks in vertical federated learning. *IEEE Transactions on Information Forensics and Security* (2024).

[56] RAY, P. P. Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems* (2023).

[57] RIBEIRO, M., GROLINGER, K., AND CAPRETZ, M. A. Mlaas: Machine learning as a service. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)* (2015), IEEE, pp. 896–902.

[58] SEMOLA, R., LOMONACO, V., AND BACCIU, D. Continual-learning-as-a-service (claas): On-demand efficient adaptation of predictive models. *arXiv preprint arXiv:2206.06957* (2022).

[59] SOMMER, D. M., SONG, L., WAGH, S., AND MITTAL, P. Athena: Probabilistic verification of machine unlearning. *Proc. Privacy Enhancing Technol 3* (2022), 268–290.

[60] STALLKAMP, J., SCHLIPSING, M., SALMEN, J., AND IGEL, C. The german traffic sign recognition benchmark: A multi-class classification competition. In *International Joint Conference on Neural Networks (IJCNN)* (2011), IEEE, pp. 1453–1460.

[61] TAO, G., AN, S., CHENG, S., SHEN, G., AND ZHANG, X. Hard-label black-box universal adversarial patch attack. In *32nd USENIX Security Symposium (USENIX Security 23)* (2023).

[62] TAO, G., SHEN, G., LIU, Y., AN, S., XU, Q., MA, S., LI, P., AND ZHANG, X. Better trigger inversion optimization in backdoor scanning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 13368–13378.

[63] THUDI, A., DEZA, G., CHANDRASEKARAN, V., AND PAPERNOT, N. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)* (2022), IEEE, pp. 303–319.

[64] TIAN, Z., CUI, L., LIANG, J., AND YU, S. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Comput. Surv. 55*, 8 (2022).

[65] TRAN, B., LI, J., AND MADRY, A. Spectral signatures in backdoor attacks. *Advances in neural information processing systems 31* (2018).

[66] TURNER, A., TSIPRAS, D., AND MADRY, A. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771* (2019).

[67] WANG, B., YAO, Y., SHAN, S., LI, H., VISWANATH, B., ZHENG, H., AND ZHAO, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 707–723.

[68] WANG, L., ZHANG, X., SU, H., AND ZHU, J. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487* (2023).

[69] WARNECKE, A., PIRCH, L., WRESSNEGGER, C., AND RIECK, K. Machine unlearning of features and labels. *Annual Network And Distributed System Security Symposium (NDSS)* (2021).

[70] WU, G., HASHEMI, M., AND SRINIVASA, C. Puma: Performance unchanged model augmentation for training data removal. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2022), vol. 36, pp. 8675–8682.

[71] XU, H., ZHU, T., ZHANG, L., ZHOU, W., AND YU, P. S. Machine unlearning: A survey. *ACM Comput. Surv. 56*, 1 (2023).

[72] XU, H., ZHU, T., ZHANG, L., ZHOU, W., AND YU, P. S. Machine unlearning: A survey. *ACM Comput. Surv. 56*, 1 (aug 2023).

[73] YAN, H., LI, X., GUO, Z., LI, H., LI, F., AND LIN, X. Arcane: An efficient architecture for exact machine unlearning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22* (2022), pp. 4006–4013.

[74] YE, J., FU, Y., SONG, J., YANG, X., LIU, S., JIN, X., SONG, M., AND WANG, X. Learning with recoverable forgetting. In *European Conference on Computer Vision* (2022), Springer, pp. 87–103.

[75] ZENG, Y., PAN, M., JUST, H. A., LYU, L., QIU, M., AND JIA, R. Narcissus: A practical clean-label backdoor attack with limited information. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2023), CCS '23, Association for Computing Machinery, p. 771–785.

[76] ZENG, Y., PARK, W., MAO, Z. M., AND JIA, R. Rethinking the backdoor attacks' triggers: A frequency perspective. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 16473–16481.

[77] ZHANG, B., LI, Z., YANG, Z., HE, X., BACKES, M., FRITZ, M., AND ZHANG, Y. Securitynet: Assessing machine learning vulnerabilities on public models. In *33rd USENIX Security Symposium (USENIX Security 24)* (2024).

[78] ZHANG, P., SUN, J., TAN, M., AND WANG, X. Backdoor attack through machine unlearning. *arXiv preprint arXiv:2310.10659* (2023).

[79] ZHANG, Y., JIA, R., PEI, H., WANG, W., LI, B., AND SONG, D. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 253–261.

[80] ZHANG, Z., PANDA, A., SONG, L., YANG, Y., MAHONEY, M., MITTAL, P., KANNAN, R., AND GONZALEZ, J. Neurotoxin: Durable backdoors in federated learning. In *Proceedings of the 39th International Conference on Machine Learning* (17–23 Jul 2022), K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162 of *Proceedings of Machine Learning Research*, PMLR, pp. 26429–26446.

[81] ZHENG, R., TANG, R., LI, J., AND LIU, L. Data-free backdoor removal based on channel lipschitzness. In *European Conference on Computer Vision* (2022), Springer, pp. 175–191.

[82] ZHONG, Y., LIU, X., ZHAI, D., JIANG, J., AND JI, X. Backdoor attacks against incremental learners: An empirical evaluation study. *arXiv preprint arXiv:2305.18384* (2023).

APPENDIX

A Discussion of Injection Ratio

Intuitively, a high injection ratio of backdoor samples to camouflage samples brings a high ASR. However, this is bad for backdoor stealthiness. Thus, UBA-Inf aims at low injection ratios of both backdoor samples and camouflage samples. Table 13 gives injection ratios used in full retrain (shard=1), SISA (shard=3,5), and approximate unlearning algorithms (e.g., PUMA and GBU). Please note these injection ratios are relatively lower than existing studies.

Table 13: Injection ratios of backdoor(b) and camouflage(c) samples.

Shards	BadNets		Blended		LC		Sig	
	b(%)	c(%)	b(%)	c(%)	b(%)	c(%)	b(%)	c(%)
CIFAR-10								
shard=1†	1.20‡	0.40	0.50	1.00	6.00	1.00	1.20	0.40
shard=3	1.20	0.60	1.20	2.40	6.00	1.80	1.20	0.40
shard=5	2.00	1.00	2.00	4.00	10.00	3.00	3.00	1.00
MNIST								
shard=1†	0.40	1.00	0.40	0.80	8.00	1.00	8.00	1.00
shard=3	0.40	1.00	0.40	0.80	8.00	1.00	8.00	1.00
shard=5	0.40	1.00	0.40	0.80	8.00	1.00	8.00	1.00
GTSRB								
shard=1†	0.80	0.25	0.80	2.00	0.95	0.10	0.95	0.10
shard=3	0.80	0.25	0.80	2.00	0.95	0.10	0.95	0.10
shard=5	0.80	0.25	0.80	2.00	0.95	0.10	0.95	0.10
Tiny								
shard=1†	1.00	0.50	0.80	2.00	0.50	0.50	0.50	1.00
shard=3	1.00	0.50	0.80	2.00	0.50	0.50	0.50	1.00
shard=5	1.00	0.50	0.80	2.00	0.50	0.50	0.50	1.00

† shard=1 represents injection rates for full retrain, PUMA and GBU.
‡ When BadNets attacks VGG-16 trained on CIFAR-10, backdoor ratio is 0.4% instead of 1.2%.

B Effectiveness on Second-Order GBU

In 5.2.2, we evaluate the effectiveness of UBA-Inf in RA-MLaaS with the first-order GBU unlearning algorithm [69]. The first-order GBU uses a first-order Taylor Series of model θ^* to derive the gradient updates. Here, we use $D_{rm} \subset D_{trn}$ to denote the set of targeted unlearning data and $\widetilde{D}_{rm} = \{\widetilde{z} \mid \widetilde{z} = (x+\delta, y), (x,y) \in D_{rm}\}$ where $x+\delta$ compose a perturbed version for unlearning. Then the model parameters are modified as $\theta^u \leftarrow \theta^* - \eta(\sum_{\widetilde{z}\in\widetilde{D}_{rm}} \nabla\ell(\widetilde{z},\theta^*) - \sum_{z\in D_{rm}}(z,\theta^*))$ where η is a pre-defined unlearning rate, ℓ is the loss function and θ^u denotes the unlearned model. Meanwhile, work [69] also proposed a second-order GBD algorithm. This second-order method uses the inverse Hessian matrix of the second-order partial derivatives to change the original model's parameters to obtain the unlearned model. The unlearned model can be formulated as $\theta^u \leftarrow \theta^* - H_{\theta^*}^{-1}(\sum_{\widetilde{z}\in\widetilde{D}_{rm}} \nabla\ell(\widetilde{z},\theta^*) - \sum_{z\in D_{rm}}(z,\theta^*))$ where $H_{\theta^*}^{-1}$ is the inverse Hessian matrix. As Table 6 demonstrates, Table 14 indicates second-order GBU unlearning camouflage samples make ASR increases from around 22% to above 80% with BA dropping less than 4% in all cases. As a result, we claim UBA-Inf can work with both first-order and second-order GBU algorithms.

Table 14: Backdoor effectiveness evaluation for Second-Order GBU.

Datasets	Models	conceal		unlearn	
		BA(%)	ASR(%)	BA(%)	ASR(%)
CIFAR-10	PARN-18	93.26	**21.94**	91.12	***80.86***
	ResNet-34	93.47	**22.10**	90.70	***81.25***
	VGG-16	90.71	**22.24**	89.92	***85.08***
MNIST	PARN-18	99.50	**29.42**	98.25	***89.10***
GTSRB	PARN-18	98.34	**22.15**	95.15	***80.27***
Tiny	PARN-18	55.56	**16.57**	51.02	***68.78***

C Discussion of Related Unlearning Defenses

Besides UBA defenses which we discuss in Section 5.3.3, work [29, 54, 78] also proposed possible defenses against malicious unlearning requests. Unlike UBA defenses, which are deliberately designed for unlearning camouflage samples, work [29, 54] consider a similar yet different threat model, in which the adversary perturbs some training samples and requests to unlearn these perturbed samples to corrupt the victim model. They reveal the risk of malicious unlearning and propose related unlearning defenses.

The authors of work [29] suggest *hashing* can effectively avoid malicious unlearning requests in their threat model because the samples for malicious unlearning are deliberately perturbed and different from the real ones in training. SP stores the hash value of each training sample and rejects the malicious unlearning requests if he finds that the hash values of the unlearned sample do not match the stored hash value. UBA-Inf can easily bypass the hashing defense because all camouflage samples the adversary needs to unlearn are already in the training dataset.

Work [54] finds that attackers can exploit influence function [33] to maliciously perturb training samples and mislead the victim model to make wrong predictions through unlearning. They discovered the perturbed samples for malicious unlearning are way different from clean samples in the gradient space. Therefore, instead of directly rejecting unlearning requests, they drop the corrected malicious samples that have a different gradient compared to other instances in their class. In order to find the corrected malicious samples, they can find

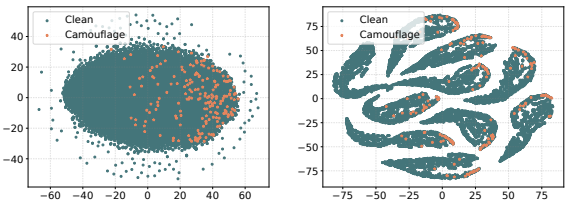

Figure 16: The t-SNE of clean and camouflage samples in the gradient space generated by the bottom layer (left one) and the top layer (right one) in the target model.

the medoids of each class in the gradient space. Such medoid-based defense fails to defeat UBA-Inf because camouflage samples share a similar gradient distribution with clean samples. Figure 16 indicates the t-SNE of clean and camouflage samples in the gradient space generated by both the bottom and top layers. The camouflage samples are completely mixed with clean samples in the gradient space, and one cannot separate them for successful medoid-based defenses.

In conclusion, there are few effective defenses against UBA-Inf, and the design of robust unlearning defenses is worryingly difficult. Firstly, recent unlearning defenses are particularly targeted at one specific risk, and such defenses often fail to defeat different unlearning attacks. This emphasizes the urge to design universal solutions to defend against various unlearning attacks. Secondly, recent defenses rely heavily on some important parameters like the threshold in [78] and the number of medoids in [54]. Setting such hyper-parameters requires a subtle balance between usability and security, which requires careful parameter tuning in real practice.

D Discussion of Unlearning Batch Size

In machine unlearning services, users remove certain data by unlearning requests. In former evaluations, we assume SP performed unlearning completely as the adversary requested, exactly removing all camouflage samples altogether. However, in some cases, SP may remove samples by batches [9, 72]. SP aggregates several small requests and divides large requests into batches for unlearning.

In exact unlearning, which is based on model re-training, batch size has little influence on UBA-Inf effectiveness because the unlearned model is essentially trained on the dataset with camouflage samples removed. However, different removal ratio (RR) of camouflage samples affects the UBA-Inf effectiveness. Table 15 demonstrates that when more camouflage samples are unlearned from the model, ASR becomes higher. When camouflage samples are completely removed, the ASR reaches nearly 100%. For better UBA-Inf effectiveness, the adversary should remove as many camouflage samples as possible.

In approximate unlearning like PUMA [70] and GBU [69], batch size of unlearn requests affects the removal of camouflage samples. If the unlearned batch size is large, camouflage samples are unlearned together with some extra clean samples.

Table 15: UBA-Inf effectiveness evaluation for full retrain with different removal ratios.

Dataset	CIFAR-10 [35]		MNIST [36]		GTSRB [60]	
$(\|D_{bd}\|, \|D_{cm}\|)$	(600,200)		(240,600)		(314,99)	
RR	BA(%)	ASR(%)	BA(%)	ASR(%)	BA(%)	ASR(%)
0	93.26	21.94	99.50	22.42	99.34	18.15
0.1	93.37	18.31	99.52	23.43	98.15	29.37
0.2	93.51	24.26	99.53	28.37	98.24	35.31
0.3	93.03	26.45	99.47	34.92	98.45	37.07
0.4	93.28	29.36	99.47	47.65	98.37	40.86
0.5	93.68	32.81	99.43	58.76	98.11	46.67
0.6	93.12	36.36	99.53	63.55	98.62	57.81
0.7	93.33	52.27	99.61	64.74	98.22	61.55
0.8	93.29	70.59	99.56	72.21	98.30	77.71
0.9	93.19	86.92	99.53	88.00	98.07	86.92
1.0	93.48	100.00	99.64	100.00	97.85	99.89

Table 16 indicates the UBA-Inf effectiveness with different unlearning batch sizes. There are 200 camouflage samples in total, and other unlearned data are randomly sampled from the clean samples. Statistics show that the UBA-Inf backdoor is successfully activated by unlearning camouflage samples mixed with some extra clean data, with ASR over 70% and BA over 85%. However, with a larger unlearning batch size, the ASR and BA become lower after unlearning.

Table 16: UBA-Inf effectiveness evaluation with different unlearning batch sizes.

	PUMA [70]		GBU [69]	
$(\|D\_bd\|, \|D\_cm\|)$	(600,200)		(600,200)	
Unlearning batch size	BA(%)	ASR(%)	BA(%)	ASR(%)
200	89.50	80.44	90.53	83.60
400	88.71	75.59	89.25	83.50
600	88.37	75.29	88.89	80.98
800	88.41	72.75	88.96	75.42
1000	88.08	74.83	88.56	67.71
2000	88.59	72.68	86.60	53.50
5000	85.55	67.13	84.51	19.71

Table 17: UBA-Inf effectiveness evaluation for multiple unlearning with small batch size.

	GBU [69]				
Unlearn times	0	1	2	3	4
BA(%)	93.26	92.76	93.28	93.31	91.20
ASR(%)	21.94	51.45	17.15	24.4	76.99
	PUMA [70]				
Unlearn times	0	1	2	3	4
BA(%)	93.26	93.26	92.14	88.22	90.33
ASR(%)	21.94	25.91	41.38	82.18	76.68

Meanwhile, when the unlearning batch size is too small, the camouflage samples need to be unlearned multiple times. If the batch size is 50, then the 200 camouflage samples should be removed in 4 batches. We evaluate the UBA-Inf effectiveness each time 50 camouflage samples are unlearned, and the results are displayed in Table 17. The UBA-Inf backdoor is activated after 4 times unlearning, with BA over 90% and ASR over 75%. In conclusion, UBA-Inf can be activated with different unlearning batch sizes, making it a more risky threat in practical MLaaS applications.