Mitigating Backdoor Attacks using Activation-Guided Model Editing

Felix Hsieh[1,2], Huy H. Nguyen[1], AprilPyone MaungMaung[1], Dmitrii Usynin[2,3], and Isao Echizen[1,4]

[1] National Institute of Informatics, Tokyo, Japan
[2] Technical University of Munich, Munich, Germany
[3] Imperial College London, London, United Kingdom
[4] The University of Tokyo, Tokyo, Japan

Abstract. Backdoor attacks compromise the integrity and reliability of machine learning models by embedding a hidden trigger during the training process, which can later be activated to cause unintended misbehavior. We propose a novel backdoor mitigation approach via machine unlearning to counter such backdoor attacks. The proposed method utilizes model activation of domain-equivalent unseen data to guide the editing of the model's weights. Unlike the previous unlearning-based mitigation methods, ours is computationally inexpensive and achieves state-of-the-art performance while only requiring a handful of unseen samples for unlearning. In addition, we also point out that unlearning the backdoor may cause the whole targeted class to be unlearned, thus introducing an additional repair step to preserve the model's utility after editing the model. Experiment results show that the proposed method is effective in unlearning the backdoor on different datasets and trigger patterns.

Keywords: Backdoor Mitigation · Machine Unlearning · Model Editing.

1 Introduction

Machine learning models highly depend on the quality and quantity of data available during training. As the demand for more powerful models increases, so does the need for vast data collections and significant computational resources for model training. Except for major corporations, most entities rely on uncurated data, such as publicly available data online and third-party services that run learning protocols. The loss of control of the training opens up an attack vector for a malicious actor to use backdoor attacks to poison the training data [29].

Gu et al. [14] proposed BadNets, the first backdoor attack. BadNets overlays a small subset of training samples with a square of fixed size and position, and it changes the labels to a target class, thus poisoning the samples. During training, the victim model learns to associate the trigger pattern with the target class, creating a hidden backdoor reactive to the trigger. During inference, the model behaves as usual on clean data. Still, when a malicious actor forwards a sample with a specific trigger, the backdoor in the neural network is activated, leading

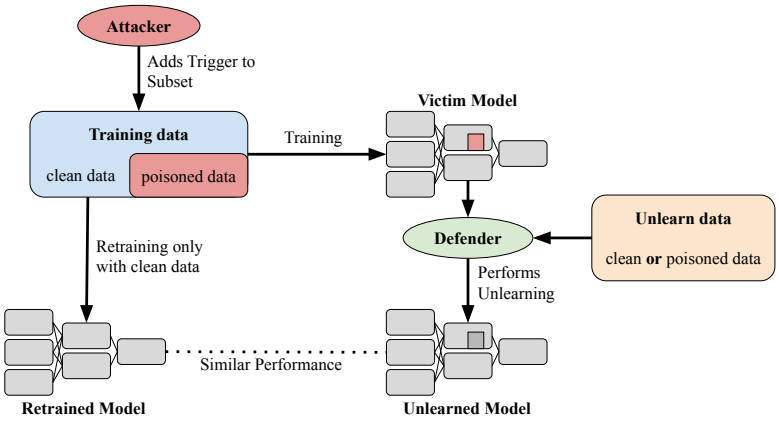


Fig. 1: Summary of backdoor unlearning setting.

to model misbehavior, such as misclassification. A survey from Microsoft stated that data poisoning is one of the top attacks on machine learning systems [24].

The security risk backdoor attacks impose creates the need for contrary defense methods. In this work, we focus on one type of defense where one mitigates the influence of a backdoor attack on an adversarial-modified model. Retraining the model from scratch with clean training data is the most straightforward approach for obtaining an adversarial-free model. Retraining is computationally expensive and requires access to clean training data. Filtering out poisoned samples in a training dataset is often unfeasible because the dataset is too large. Backdoor mitigation with machine unlearning has emerged as a promising approach to overcoming the limitations of retraining and efficiently removing a backdoor in a poisoned model. Many methods omit the need for original training data.

This work focuses on a realistic scenario where the defender cannot access the dataset used to train the victim model. Figure 1 shows an overview of our unlearning setting. Unlearning aims to obtain a model that performs similarly to a model retrained on a clean subset of training data for clean and poisoned data. For unlearning the backdoor, we have access only to a limited domain-equivalent unseen dataset, the length of which is an order of magnitude smaller than the original training dataset. In practice, collecting a fitting dataset with annotations is expensive and time-consuming. Methods that are effective with an even smaller data count available are of particular interest for this scenario because they allow secure hand-picking of clean data for unlearning.

We propose a novel backdoor-unlearning approach that uses information from an extracted activation to guide the editing of model weights. This process aims to mitigate the influence of backdoor samples in the training dataset. Editing the weights is beneficial because it allows us to selectively target and repair the parts compromised by the attack. In addition to directly editing the weights,

unlearning benefits from optionally allowing the parameters of the Batch Normalization (BN) layer to be changed during activation extraction.

The contributions of this work are as follows:

- We propose a novel model-editing method for unlearning samples with backdoor triggers by utilizing the activation of clean or poisoned samples extracted for a backdoored model. The proposed method is time- and sample-efficient.
- We point out that the proposed unlearning might unlearn the targeted class, thus introducing an optional repair process to preserve utility while forgetting only the backdoor trigger.
- We conduct experiments under two scenarios (with or without knowledge of the backdoor trigger) with three state-of-the-art backdoor attacks on different models and datasets. We present the results with an analysis.

In the experiments, the proposed method can consistently outperform other baseline methods.

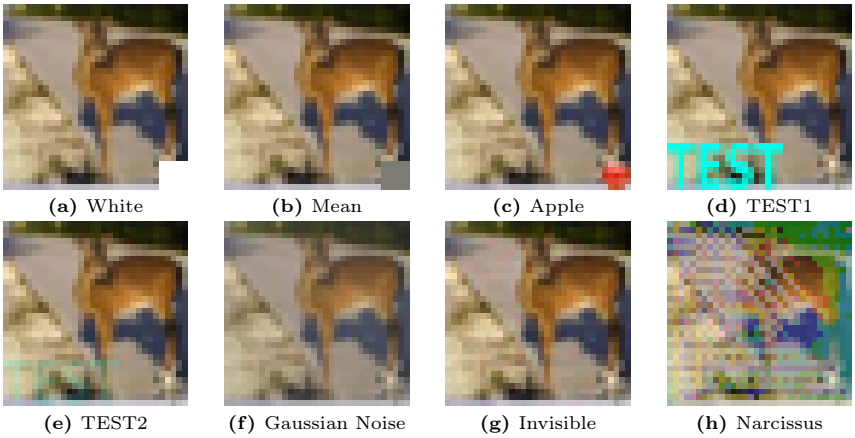
2 Related Work

This section briefly reviews backdoor attacks, backdoor defenses, and machine unlearning.

2.1 Backdoor Attacks

Backdoor attacks involve preemptively poisoning a subset of training data with a specific backdoor trigger pattern and a target label. During training, a neural network learns that images with a specified trigger correspond to a target class, thus introducing an additional adversarial task. During inference, the network works as usual on benign data. A malicious actor can activate the backdoor to manipulate model response, causing misbehavior, such as misclassifications.

There exist various types of backdoor attacks [29]. BadNets [14], the first backdoor attack, uses noticeable square patches as triggers. In contrast to visible triggers, for invisible triggers, poisoned images are indistinguishable from clean ones, as in [27, 30]. Backdoor attacks with optimized triggers [34, 50] are designed to be more effective and thus usually require fewer poisoned training samples. Moreover, a shared semantic part of the images can be used as a trigger [1, 31] without manipulating the images and only changing the labels. In addition, instead of using a single trigger pattern, certain methods allow for varying sample-specific triggers [36]. Although the targeted label is usually for a single class, there are all-to-all attacks [15] that use different target labels. In this work, we use visible, invisible, and optimized triggers for our experiments, and examples of such triggers are shown in Figure 2.



(a) White **(b)** Mean **(c)** Apple **(d)** TEST1

(e) TEST2 **(f)** Gaussian Noise **(g)** Invisible **(h)** Narcissus

Fig. 2: Examples of eight backdoor triggers on CIFAR10. Images (a)–(f) are poisoned by BadNets [14] with different patches, (g) is with Steganography [27], and (h) is with Narcissus [50].

2.2 Backdoor Defenses

With the development of backdoor attacks, researchers have proposed various backdoor defenses as countermeasure [29]. Most defense methods should instead be considered mitigation methods because, in most cases, they cannot entirely erase the influence of the attacks. One such mitigation is data pre-processing prior to model inference, which aims to perturb the trigger to not activate the backdoor [7, 35]. Another form of defense, typically used to aid other defense methods, is trigger synthesis [45]. With reverse engineering, trigger synthesis approximates the trigger, which can be used for trigger-guided defense or to retrieve the target class. Model diagnosis [22, 49] is another type of defense to detect a backdoor and prevent model deployment. Moreover, poison suppression defenses modify the training process to be robust against backdoor creation [9, 19]. Other methods utilize sample filtering to detect trigger images and remove them from the training set or decline them during model inference [5, 10, 44]. Some defense methods aim to remove the backdoor from an infected model by directly modifying the model [32, 35]. In this work, we propose a method that directly edits model weights to erase the backdoor in a poisoned model. The editing uses the activation of clean or poisoned samples as a guide.

2.3 Machine Unlearning

The influence of specific training samples on a model can be mitigated with machine unlearning. This influence can be entirely removed by retraining the model from scratch without the data we want to forget. One limitation of this approach is the requirement for training data, which can be inaccessible or too big to filter out the data we want to forget. Another issue is the high computational and time resource expense associated with retraining [48].

Different machine unlearning methods try to evade some of those limitations [48]. One approach is data obfuscation [13, 42], where the model is fine-tuned with additional obfuscated data that disturbs the functionality of the data we want to forget. Certain approaches require design choices prior to training, like multi-model-aggregation [3, 17] or a transformation layer inserted between data and model [4]. For specific model manipulation methods, model weights can be shifted by an update value [11, 16], replaced by new values [38, 47], or pruned [2,46] and usually repaired with a subsequent fine-tuning step. The scope of the information targeted for unlearning can range from whole classes [39, 42] to individual samples [13]. This work focuses on backdoor attacks and aims to unlearn the features of a backdoor trigger pattern learned by the victim model.

3 Methodology

We consider an adversarial-modified (backdoored) image classifier f_θ parameterized by θ, which is trained with a dataset D that is comprised of clean and backdoored data ($D = D_C \cup D_B$). Samples in D_B contain the backdoor trigger δ and have the target label y_t. D_B is usually a small fraction of a clean training set D_T with a budget ρ such that $|D_B| \leq \rho|D_T|$. f_θ takes an input image $x \in \mathcal{X}$ and $f_\theta(x)_i$ represents the probability that x corresponds to label $i \in \mathcal{Y}$. $\mathcal{X}$ is the input space, and $\mathcal{Y}$ is the label space. The predicted label $\hat{y}$ is obtained by using the arg max operation ($\arg\max_i f_\theta(x)_i$). Since f_θ is backdoored, f_θ works as normal on a clean input x_c (*i.e.* predicting $\hat{y}$) and predicts y_t for input x_b embedded with the backdoor trigger δ. We aim to unlearn D_B that f_θ does not predict y_t when given x_b. Here, we slightly abuse the notation and imply that f_θ is a deep neural network with multiple layers. Specifically, we consider a neural network with multiple blocks of convolutional layers with or without BN.

Given f_θ without having access to the training dataset D, we propose an activation-guided model editing approach to unlearn D_B under two assumptions: (1) we have Backdoor Knowledge (BDK), and (2) we do not have it (¬BDK). For both assumptions, we split the total weights of f_θ into two halves and add those layers corresponding to the weights of the second half to a layer list L, for which we want to edit the weights. We target those later layers because they have the highest proximity to the classification output. We do not want to edit the early layers associated with general low-level feature extraction [12]. The authors of other backdoor mitigation approaches also suggest that focusing the unlearning on the later layers improves performance [32, 45]. First, we prepare an unlearning dataset D_U with the same distribution as the training dataset D. However, D_U is not used in training f_θ. Our empirical experiments suggest that D_U can be as small as four samples.

3.1 Assumption 1 - BDK

As we assume we have backdoor information in this scenario, we poison D_U with the known backdoor trigger δ. In addition, for models with BN, we freeze

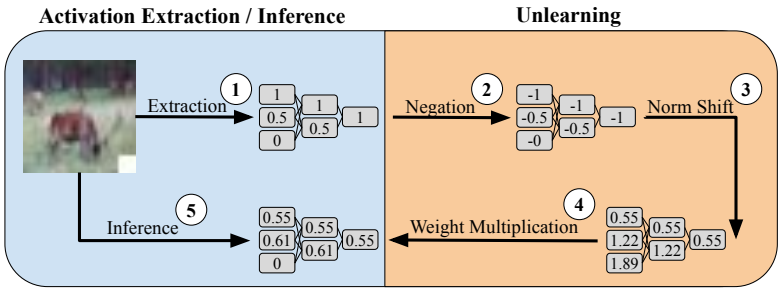


Fig. 3: Overview of proposed activation-guided model editing approach.

Moving Average (MA) parameters during activation extraction because this was experimentally proven to be more effective for unlearning. Figure 3 shows an overview of the proposed model editing process, which works as follows.

① Extract model activation A for the whole dataset D_U. Therefore, iterate batch-wise over data X in D_U infected with trigger δ and forward it through model f_θ given $f_\theta(X+\delta)$. Capture and average the activations across all batches for each layer l in layer list L.

② Next, negate each value in the activation A as

$$A = -A. \tag{1}$$

Then, iterate over l in list L of the layers targeted for editing. For each layer l, check if the later multiplication of weights θ_l and activation A_l would result in a matrix-multiplication-error caused by shape mismatch. If this is the case, use adaptive average pooling to adjust the shape of A_l to match the shape of θ_l.

③ Then, compute layer-wise mean and standard deviation statistics as

$$\mu_l = \frac{1}{m}\sum_{i=1}^{m} a_i, \quad \sigma_l^2 = \frac{1}{m}\sum_{i=1}^{m}(a_i - \mu_l)^2, \tag{2}$$

where $a_i \in A_l$. Then, use the calculated statistics to normalize activation A_l for each layer l as

$$A_l = \frac{A_l - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}}, \tag{3}$$

where ϵ is a small value to avoid division by zero. Next, rescale A_l with scale and shift hyperparameters γ and λ as

$$A_l = \gamma \cdot A_l + \lambda. \tag{4}$$

④ With the calculated activation factor A_l, edit the model weights θ_l as

$$\theta_l = \theta_l \cdot A_l. \tag{5}$$

Modifying θ for all layers in L completes the unlearning process. We obtain an unlearned model with a mitigated backdoor.

⑤ Finally, evaluate the classification performance on a separate verification test set D_V to evaluate the unlearning process.

In Equation 4, we scale and shift the activation with values 0.5 and 1.0 for γ and λ hyperparameters, which changes the activation mean to 1.0. With this, we aim to preserve the utility of the model during unlearning and mitigate the internal covariate shift, especially after significant distribution changes by Equation 1. Choosing those values should lead to minimal change in the inherent mean of the weights when multiplying with the shifted activation. We experimentally confirmed the performance with those values in the supplementary material.

After model editing, we observe that the model tends to unlearn the whole class instead of unlearning backdoored samples only. To address this issue, we introduce an optional repair phase to restore some model utility if we have unlearned more than intended. Specifically, we fine-tune the model on the samples in D_U and on the same samples poisoned with δ for one epoch each, using the correct ground truth as the target label in both cases.

3.2 Assumption 2 - ¬BDK

In this scenario, we do not have any information about the backdoor trigger or algorithm (¬BDK). Therefore, we cannot poison the unlearning dataset D_U and use the clean D_U as it is in the unlearning process. For model editing with ¬BDK, we perform the process in Figure 3 with two modifications: (1) backdoor trigger δ is zero as we do not have any information about it, and (2) we update MA parameters during activation extraction to aid unlearning of dataset D_U for models with BN layers, unlike the unlearning process with BDK. We perform the optional repairing by fine-tuning on the clean D_U set.

4 Experiments

In this section, we perform various experiments to show the effectiveness of our method in different settings and compare it with other existing backdoor unlearning methods. We conducted all experiments three times, and the averaged results are summarized as follows.

4.1 Setup

Datasets. We explored our method on MNIST [26], CIFAR10, CIFAR100 [23], CINIC10 [6], and TinyImageNet [25].

Models. We used ResNet18 [18], VGG16 [40], EfficientNetV2-S [41], and small MobileNetV3 [20].

Backdoors. We considered eight different triggers applied with three state-of-the-art attack methods: six different patch triggers with BadNets [14], an invisible trigger with Steganography [27], and an optimized trigger with Narcissus [50]. Among the methods, Narcissus is the only one that solely infects target

Table 1: Evaluation of proposed unlearning on five different datasets. Additional repairing was performed with learning rate value of 1e−2. We compare base state after unlearning with one after repairing. We used Top-5 accuracy for CIFAR100 and TinyImageNet. Best results are highlighted in bold.

Dataset (ASR/ACC)	BDK	State	Metric		
			ASR (↓)	ACC (↑)	CTCA (↑)
MNIST (99.99/98.98)	✓	Base	0.0±0.0	86.04±1.2	0.0±0.0
		+repair	0.41±0.16	**97.77±0.37**	**96.09±2.51**
	✗	Base	66.81±5.87	77.37±2.84	59.19±11.56
		+repair	67.3±12.4	**97.31±1.27**	**98.13±0.31**
CIFAR10 (94.53/70.5)	✓	Base	0.0±0.0	**65.49±0.39**	0.0±0.0
		+repair	9.79±0.88	59.87±3.04	**42.07±7.72**
	✗	Base	**0.3±0.21**	59.18±2.63	0.14±0.19
		+repair	23.09±8.07	61.83±1.33	**55.02±4.41**
CIFAR100 (93.84/63.06)	✓	Base	0.92±0.33	50.68±1.36	3.91±2.8
		+repair	4.13±1.61	**60.42±0.96**	**41.27±10.27**
	✗	Base	**0.0±0.0**	37.88±2.19	0.0±0.0
		+repair	20.3±14.16	**60.83±0.81**	**58.78±12.86**
CINIC10 (96.28/57.35)	✓	Base	0.0±0.0	**53.08±0.49**	0.0±0.0
		+repair	0.0±0.0	10.0±0.0	0.0±0.0
	✗	Base	0.01±0.01	**41.89±0.83**	0.08±0.1
		+repair	0.0±0.0	10.0±0.0	0.0±0.0
TinyImageNet (95.25/45.21)	✓	Base	0.48±0.32	34.21±1.0	1.59±2.24
		+repair	0.86±0.33	**44.33±0.16**	4.14±3.75
	✗	Base	0.0±0.0	20.26±1.87	0.0±0.0
		+repair	0.72±0.6	**44.03±0.21**	**7.72±5.71**

Table 2: Evaluation of proposed unlearning on different models. Additional repairing was performed with learning rate value of 1e−2. We compare base state after unlearning with one after repairing. Best results are highlighted in bold.

Model (ASR/ACC)	BDK	State	Metric		
			ASR (↓)	ACC (↑)	CTCA (↑)
ResNet18 (94.53/70.5)	✓	Base	**0.0±0.0**	**65.49±0.39**	0.0±0.0
		+repair	9.79±0.88	59.87±3.04	**42.07±7.72**
	✗	Base	**0.3±0.21**	59.18±2.63	0.14±0.19
		+repair	23.09±8.07	61.83±1.33	**55.02±4.41**
VGG16 (96.25/78.39)	✓	Base	0.0±0.0	71.23±0.38	0.0±0.0
		+repair	4.36±1.15	70.15±2.71	**65.5±4.69**
	✗	Base	**0.0±0.0**	70.13±1.74	0.0±0.0
		+repair	92.17±1.17	72.23±0.32	**57.83±7.3**
EfficientNetV2-S (89.74/48.83)	✓	Base	0.0±0.0	**47.49±2.79**	0.0±0.0
		+repair	6.82±2.96	31.51±5.45	**7.94±3.66**
	✗	Base	0.0±0.0	11.07±1.5	0.0±0.0
		+repair	1.71±2.02	**31.41±6.19**	**4.6±4.35**
MobileNetV3 (small) (95.67/62.3)	✓	Base	**0.0±0.0**	**55.8±1.84**	0.0±0.0
		+repair	26.28±7.53	50.63±3.63	**49.18±11.63**
	✗	Base	**6.09±4.54**	54.35±2.93	4.19±2.66
		+repair	84.69±6.09	51.05±3.68	**46.51±15.49**

class samples, thus making it more stealthy without requiring a label change. Examples of the applied triggers are visualized in Figure 2.

Baselines. We considered four suitable backdoor unlearning methods for comparison, two of which require BDK: (1) fine-tuning, which penalizes a high difference between activations of clean and poisoned data (actFT) [37], and (2) BaEraser, which uses gradient ascent for unlearning [33]. The other two methods work with ¬BDK: (3) fine-tuning on clean D_U in the same way as the initial training (basicFT), and (4) Neural Attention Distillation (NAD), a knowledge distillation approach where the basicFT model, acting as the teacher model, only passes on its ability to clean data [28].

Evaluation Metrics. For evaluation, we used the Attack Success Rate (ASR), which is the ratio of a backdoor sample being misclassified as y_t, Clean Test Accuracy (ACC), and Clean Target Class Accuracy (CTCA), which is the ACC for samples of class y_t. We were interested in examining the change in CTCA because our unlearning method often leads to a drop in CTCA alongside ASR. Repairing can mitigate this side effect.

Table 3: Score (↑) comparison of proposed method with state-of-the-art unlearning methods on different infected backdoors. For backdoor verification, *alpha* value is multiplied by three up to maximum of 100%. Best results for each trigger are highlighted in bold.

Infected Trigger	poisoned			BDK		
	(ASR/ACC)	ρ	*alpha*	actFT [37]	BaEraser [33]	Ours
(a) White [14]	(94.53/70.5)	5%	1.0	**94.62±2.61**	59.5±4.62	93.02±0.66
(b) Mean	(88.49/68.71)	10%	1.0	61.81±6.09	70.85±7.33	**95.23±1.79**
(c) Apple	(99.23/69.63)	5%	1.0	85.21±11.5	62.87±11.29	**92.36±2.11**
(d) TEST1	(99.75/70.73)	5%	1.0	90.69±2.43	36.99±8.55	**92.19±0.87**
(e) TEST2	(99.99/69.66)	5%	0.15	88.3±1.19	48.57±19.28	**91.31±1.41**
(f) Gaussian Noise	(85.77/68.19)	5%	0.25	52.97±2.19	89.6±1.56	**93.57±0.35**
(g) Invisible [27]	(97.97/61.4)	50%	-	84.99±3.83	40.51±12.39	**92.24±2.38**
(h) Narcissus [50]	(99.32/63.72)	5%	0.2	4.09±1.61	51.35±14.04	**95.45±3.1**

We introduce a two-part scoring function to estimate the forgetting and utility quality after unlearning combined in one value. The forgetting quality is estimated by subtracting the ASR ratio of the unlearned (U) and victim model (V) from 1. A higher drop in ASR after unlearning indicates a higher score for the forgetting part. The ACC ratio of the unlearned and the retrained model (R), which is trained on D_C from scratch, estimates the utility part. We strive to achieve the same or even higher ACC on the unlearned model compared to the retrained model that was never poisoned before. We use the retrained model for this ratio because, especially in cases with a high poisoning rate ρ, the poisoning can negatively influence the ACC of the victim model, thus not representing a clean model performance. The final score value is calculated as

$$\text{Score} = (1 - \frac{\text{ASR}^U}{\text{ASR}^V}) \cdot \frac{\text{ACC}^U}{\text{ACC}^R}. \tag{6}$$

Base Configuration. We used specific base configurations if not stated otherwise for an experiment. Experiments were performed on the CIFAR10 dataset and ResNet18 as the victim model. The training dataset D_T was infected with a poisoning rate ρ of 5%, with the trigger displayed in Figure 2a. The backdoor target class y_t was two, representing birds. Unlearn dataset D_U consisted of 5000 samples, but our method only used 512 by default.

4.2 Results

We examined the performance of our unlearning in different settings.

Different Datasets. In this experiment, we trained ResNet18 models poisoned with backdoor triggers on five datasets: MNIST, CIFAR10, CIFAR100, CINIC10, and TinyImageNet. Table 1 summarizes the evaluation of the proposed unlearning method with the different datasets in terms of ASR, ACC, and CTCA. The

Table 4: Score (↑) comparison of proposed method with state-of-the-art unlearning methods on different infected backdoors. For backdoor verification, *alpha* value is multiplied by three up to maximum of 100%. Best results for each trigger are highlighted in bold.

Infected Trigger	poisoned			¬BDK		
	(ASR/ACC)	ρ	*alpha*	basicFT	NAD [28]	Ours
(a) White [14]	(94.53/70.5)	5%	1.0	61.9±3.63	65.33±3.04	**83.73±3.29**
(b) Mean	(88.49/68.71)	10%	1.0	50.01±25.7	71.79±2.52	**89.95±1.73**
(c) Apple	(99.23/69.63)	5%	1.0	69.1±2.7	70.86±0.15	**80.41±3.78**
(d) TEST1	(99.75/70.73)	5%	1.0	31.44±23.82	62.26±3.38	**83.4±3.89**
(e) TEST2	(99.99/69.66)	5%	0.15	48.56±14.03	68.17±4.35	**82.47±0.65**
(f) Gaussian Noise	(85.77/68.19)	5%	0.25	41.92±22.62	31.19±13.93	**82.81±0.77**
(g) Invisible [27]	(97.97/61.4)	50%	-	58.66±14.52	73.7±1.1	**87.26±0.58**
(h) Narcissus [50]	(99.32/63.72)	5%	0.2	38.52±16.51	**75.75±1.38**	17.73±11.77

proposed method effectively reduced the ASR on every dataset, except MNIST (grayscale images), when we had ¬BDK. Repairing improved ACC for several datasets and restored CTCA while increasing ASR by a lesser extent. There was an exclusively negative influence on performance with CINIC10 repairing.

Different Models. In this experiment, we trained different models: ResNet18, VGG16, EfficientNetV2-S, and MobileNetV3 (small version). Table 2 presents the performance of the proposed unlearning method with the different models. The proposed method with BDK was effective on every tested model. After unlearning, we retained a good ACC on EfficientNetv2 with BDK, while the utility was lost with ¬BDK. However, repairing both models resulted in similar final performance, which benefited ¬BDK but decreased performance for BDK.

Comparison with State-of-the-Art Methods. In this experiment, we trained models and performed unlearning with different backdoor triggers. As described in Section 4.1, we considered four baseline methods: actFT and BaEraser under BDK, and basicFT and NAD under ¬BDK with eight poison triggers for comparison. The models were trained with different poisoning budgets ρ and *alpha* values of the RGBA-coded trigger. Figure 2 depicts the triggers.

Tables 3 and 4 summarize the performance of the proposed unlearning method with the different baseline methods in terms of score (see Section 4.1). The score metric measured the forgetting and utility quality after unlearning. For backdoor verification, the *alpha* value was multiplied by three up to a maximum of 100%. For actFT to be effective, we multiplied the *alpha* value for unlearning by the same magnitude. Our method outperformed the previous methods in terms of score with or without BDK for most triggers.

Table 5: Performance of proposed unlearning when using different numbers of samples for unlearning. Results represent the model state after unlearning without repairing. Best results are highlighted in bold.

Number of	BDK			¬BDK		
samples	ASR (↓)	ACC (↑)	CTCA (↑)	ASR (↓)	ACC (↑)	CTCA (↑)
2	0.0±0.0	63.92±1.91	0.0±0.0	24.6±17.84	61.1±1.94	**6.48±9.02**
4	0.0±0.0	65.18±0.91	0.0±0.0	**0.92±1.15**	60.77±2.25	0.0±0.0
8	0.0±0.0	65.14±0.65	0.0±0.0	5.76±6.26	60.19±2.96	0.0±0.0
16	0.0±0.0	65.17±0.68	0.0±0.0	4.82±6.75	60.77±2.91	0.0±0.0
32	0.0±0.0	65.47±0.44	0.0±0.0	9.26±12.93	**63.77±0.58**	0.07±0.1
64	0.0±0.0	64.7±1.45	0.0±0.0	7.63±8.32	61.9±2.22	1.94±2.74
128	0.0±0.0	65.53±0.45	0.0±0.0	4.38±5.58	61.67±1.22	0.07±0.1
256	0.0±0.0	65.51±0.41	0.0±0.0	**0.97±1.02**	58.86±2.42	0.2±0.16
512	0.0±0.0	65.49±0.39	0.0±0.0	**0.3±0.21**	59.18±2.63	0.14±0.19
5000	0.0±0.0	64.57±1.53	0.0±0.0	10.48±11.24	61.41±0.82	**6.75±9.54**

4.3 Analysis

We analyze the proposed unlearning method in terms of sample efficiency, time efficiency, and potential backdoor detection application.

Sample Efficiency. Table 5 shows the performance of the proposed unlearning method when using different numbers of samples for unlearning. With BDK, the performance did not depend on the sample count. With ¬BDK, the performance with different sample counts did not follow a clear pattern. Notably, the ASR with two samples was exceptionally high compared with others. Therefore, we recommend using a minimum of four samples for unlearning with ¬BDK.

Table 6 presents a performance comparison of the proposed unlearning and state-of-the-art methods in terms of several metrics, including the time required for unlearning. The baseline methods compared with ours depended more on a high sample count in D_U. For most of the baselines, more samples resulted in a higher score. An exception is BaEraser, which had the best performance with 500 samples.

Time Efficiency. In the particular scenario where training data is available and retraining is feasible, assessing the computational cost saved with unlearning compared with retraining is an important metric. When unlearning is not drastically more time efficient, retraining is the preferred choice to perfectly remove the influence of the data we want to forget. The unlearning time in our scenario with training data unavailability is not a deciding factor. Still, we have to consider the trade-off between unlearning performance and the cost of computing for the benefit of scalability.

As evident in Table 6, our method requires significantly less time and fewer samples for unlearning than other methods. Our method uses only a single forward pass to extract the activation, and the remaining operations are simple matrix operations. In comparison, all baseline methods require optimization with

Table 6: Efficiency of proposed unlearning compared with state-of-the-art methods. Experimented with 50%(5000), 5%(500), 0.5%(50), and 0.05%(5) of unseen CIFAR10 data for unlearning. Table displays only sample runs with highest and second-highest scores. Full table is displayed in supplementary material. Best results are highlighted in bold.

Method	Number of Samples	Metric				
		Score (↑)	ASR (↓)	ACC (↑)	CTCA (↑)	Time (↓)
actFT [37]	5000	**92.95±4.72**	5.28±2.48	**69.64±0.92**	**58.1±9.1**	3.96±0.34
	500	7.0±2.09	87.17±2.86	69.41±1.02	61.47±7.99	2.81±0.04
BaEraser [33]	500	80.69±1.93	3.81±2.41	59.45±0.83	32.98±1.1	138.88±2.87
	5000	57.47±15.65	2.0±2.79	41.81±12.3	14.79±20.77	623.3±111.46
Ours(BDK)	50	**92.4±1.39**	**0.0±0.0**	65.29±0.58	0.0±0.0	**0.38±0.01**
	5	92.51±1.34	0.0±0.0	65.37±0.56	0.0±0.0	0.38±0.02
basicFT	5000	69.49±1.11	6.0±1.07	52.57±0.91	**33.86±4.07**	71.56±1.47
	5	14.17±0.11	0.0±0.0	10.01±0.0	0.0±0.0	71.49±0.03
NAD [28]	5000	71.23±3.2	4.54±1.48	52.95±1.41	32.05±1.3	114.9±1.45
	500	42.18±5.42	6.68±2.06	32.09±3.97	23.18±9.4	117.69±1.9
Ours(¬BDK)	50	**89.9±2.02**	**0.0±0.0**	**63.52±1.24**	0.0±0.0	**0.36±0.02**
	5	89.5±1.72	0.0±0.0	63.23±0.86	0.0±0.0	0.4±0.01
Retraining	47500	-	4.03±0.99	70.27± 0.66	60.55±4.3	423.56±73.96

backpropagation, which generally is more computationally expensive, resulting in a higher unlearning time.

Target Class Detection. Our experiments show that the proposed unlearning method reduced the backdoor class accuracy (CTCA). To address this issue, we introduce a repair step to preserve utility. Before repairing, we can utilize significant decreases in target class accuracy with ¬BDK to detect a backdoor and the target class. We carried out a simple experiment on poisoned models on all ten classes of CIFAR10. We can usually observe an unusual decrease in accuracy for a single class. When we assumed the single class as the target class, we got a target class prediction accuracy of 80%. A formula sets the accuracy of the different classes into relation and returned a classification value. Comparing the value to a threshold value gives us a binary prediction for the existence of a backdoor. The backdoor detection accuracy was 67% when poisoned and 80% when having a clean model.

5 Discussion

We demonstrated a model-editing method that unlearns the backdoor trigger feature embedded in a backdoored model by utilizing the activation of clean or poisoned samples. Our method achieves consistent unlearning performance

across various settings with different models, datasets, and backdoor triggers by state-of-the-art attacks. Apart from the unlearning performance, there are two key factors where our method exceeds current state-of-the-art methods by a significant margin. Our unlearning process is exceptionally fast to compute and, most of the time, requires only a handful of samples to unlearn the backdoor effectively. Additionally, we can use information gained after unlearning for backdoor presence and target class prediction.

We experimented with our algorithm and found specific activation-manipulating formulas that gave us the best unlearning performance for model editing. In Equation 1, negating poisoned activation with BDK and clean activation with ¬BDK worked the best. With BDK, we negate the activation of the trigger-infected data we want to forget. Previously, Ilharco et al. [21] arrived at the same conclusion as we did, that moving in the negative direction of extracted information can lead to unlearning.

The most significant limitation of our method is that it disturbs the overall utility and unlearns the targeted class instead of only backdoor samples. Therefore, repairing is used to restore lost utility. The experimental scope was limited, and we covered only convolutional neural networks.

Hence, for future work, we shall explore the unlearning method with different architectures, such as vision transformers [8], mixers [43], *etc*. We shall investigate explainability methods to better understand the parts of the algorithm that are responsible for effective unlearning and ideally improve the unlearning performance without loss of utility. In addition, not limiting the method to backdoor unlearning, we shall expand the applications of unlearning, such as privacy-related unlearning applications. In this work, we analyzed backdoors in images, but for future work, we shall expand experiments to other data types, such as text or audio data.

6 Conclusion

Our method offers a new approach to tackling the security issue posed by backdoor attacks by mitigating the influence of attacks on a backdoor-infected model without requiring access to the original training data. Multiple experiments show the broad applicability of our method in various settings. It performs better than previous backdoor unlearning methods in most scenarios. Moreover, it executes faster and requires fewer samples for unlearning than the previous methods.

Acknowledgements. This work was partially supported by JSPS KAKENHI Grants JP21H04907, 23K19983, and JP24H00732, by JST CREST Grants JPMJCR18A6 and JPMJCR20D3 including AIP challenge program, by JST AIP Acceleration Grant JPMJCR24U3, by JST K Program Grant JPMJKP24C2 Japan, and by the project for the development and demonstration of countermeasures against disinformation and misinformation on the Internet with the Ministry of Internal Affairs and Communications of Japan.

References

1. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: International conference on artificial intelligence and statistics. pp. 2938–2948. PMLR (2020) 3
2. Baumhauer, T., Schöttle, P., Zeppelzauer, M.: Machine unlearning: Linear filtration for logit-based classifiers. Machine Learning **111**(9), 3203–3226 (2022) 5
3. Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C.A., Jia, H., Travers, A., Zhang, B., Lie, D., Papernot, N.: Machine unlearning (2020) 5
4. Cao, Y., Yang, J.: Towards making systems forget with machine unlearning. In: 2015 IEEE symposium on security and privacy. pp. 463–480. IEEE (2015) 5
5. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering (2018) 4
6. Darlow, L.N., Crowley, E.J., Antoniou, A., Storkey, A.J.: Cinic-10 is not imagenet or cifar-10. arXiv preprint arXiv:1810.03505 (2018) 7
7. Doan, B.G., Abbasnejad, E., Ranasinghe, D.C.: Februus: Input purification defense against trojan attacks on deep neural network systems. In: Proceedings of the 36th Annual Computer Security Applications Conference. pp. 897–912 (2020) 4
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale (2021) 14
9. Du, M., Jia, R., Song, D.: Robust anomaly detection and backdoor attack detection via differential privacy. arXiv preprint arXiv:1911.07116 (2019) 4
10. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: A defence against trojan attacks on deep neural networks (2020) 4
11. Golatkar, A., Achille, A., Soatto, S.: Eternal sunshine of the spotless net: Selective forgetting in deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9304–9312 (2020) 5
12. Goodfellow, I.J., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, MA, USA (2016), http://www.deeplearningbook.org 5
13. Graves, L., Nagisetty, V., Ganesh, V.: Amnesiac machine learning (2020) 5
14. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain (2019) 1, 3, 4, 7, 10, 11
15. Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019) 3
16. Guo, C., Goldstein, T., Hannun, A., Van Der Maaten, L.: Certified data removal from machine learning models. arXiv preprint arXiv:1911.03030 (2019) 5
17. Gupta, V., Jung, C., Neel, S., Roth, A., Sharifi-Malvajerdi, S., Waites, C.: Adaptive machine unlearning. Advances in Neural Information Processing Systems **34**, 16319–16330 (2021) 5
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015) 7
19. Hong, S., Chandrasekaran, V., Kaya, Y., Dumitraş, T., Papernot, N.: On the effectiveness of mitigating data poisoning attacks with gradient shaping. arXiv preprint arXiv:2002.11497 (2020) 4
20. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for mobilenetv3 (2019) 7

21. Ilharco, G., Ribeiro, M.T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., Farhadi, A.: Editing models with task arithmetic (2023) 14
22. Kolouri, S., Saha, A., Pirsiavash, H., Hoffmann, H.: Universal litmus patterns: Revealing backdoor attacks in cnns (2020) 4
23. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Master's thesis, University of Tront (2009) 7
24. Kumar, R.S.S., Nyström, M., Lambert, J., Marshall, A., Goertzel, M., Comissoneru, A., Swann, M., Xia, S.: Adversarial machine learning-industry perspectives. In: 2020 IEEE security and privacy workshops (SPW). pp. 69–75. IEEE (2020) 2
25. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**(7), 3 (2015) 7
26. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998) 7
27. Li, S., Xue, M., Zhao, B.Z.H., Zhu, H., Zhang, X.: Invisible backdoor attacks on deep neural networks via steganography and regularization (2020) 3, 4, 7, 10, 11
28. Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X.: Neural attention distillation: Erasing backdoor triggers from deep neural networks (2021) 9, 11, 13
29. Li, Y., Jiang, Y., Li, Z., Xia, S.T.: Backdoor learning: A survey (2022) 1, 3, 4
30. Li, Y., Li, Y., Wu, B., Li, L., He, R., Lyu, S.: Invisible backdoor attack with sample-specific triggers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 16463–16472 (2021) 3
31. Lin, J., Xu, L., Liu, Y., Zhang, X.: Composite backdoor attack for deep neural network by mixing existing benign features. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 113–131 (2020) 3
32. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks (2018) 4, 5
33. Liu, Y., Fan, M., Chen, C., Liu, X., Ma, Z., Wang, L., Ma, J.: Backdoor defense with machine unlearning (2022) 9, 10, 13
34. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks. In: Network and Distributed System Security Symposium (2018), https://api.semanticscholar.org/CorpusID:31806516 3
35. Liu, Y., Xie, Y., Srivastava, A.: Neural trojans (2017) 4
36. Nguyen, T.A., Tran, A.: Input-aware dynamic backdoor attack. Advances in Neural Information Processing Systems **33**, 3454–3464 (2020) 3
37. Qiao, X., Yang, Y., Li, H.: Defending neural backdoors via generative distribution modeling (2019) 9, 10, 13
38. Schelter, S., Grafberger, S., Dunning, T.: Hedgecut: Maintaining randomised trees for low-latency machine unlearning. In: Proceedings of the 2021 International Conference on Management of Data. pp. 1545–1557 (2021) 5
39. Shibata, T., Irie, G., Ikami, D., Mitsuzumi, Y.: Learning with selective forgetting. In: IJCAI. vol. 3, p. 4 (2021) 5
40. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015) 7
41. Tan, M., Le, Q.V.: Efficientnetv2: Smaller models and faster training (2021) 7
42. Tarun, A.K., Chundawat, V.S., Mandal, M., Kankanhalli, M.: Fast yet effective machine unlearning. IEEE Transactions on Neural Networks and Learning Systems (2023) 5
43. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., Dosovitskiy, A.: Mlp-mixer: An all-mlp architecture for vision (2021) 14

44. Tran, B., Li, J., Madry, A.: Spectral signatures in backdoor attacks (2018) 4
45. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 707–723. IEEE (2019) 4, 5
46. Wang, J., Guo, S., Xie, X., Qi, H.: Federated unlearning via class-discriminative pruning. In: Proceedings of the ACM Web Conference 2022. pp. 622–632 (2022) 5
47. Wu, Y., Dobriban, E., Davidson, S.B.: Deltagrad: Rapid retraining of machine learning models (2020) 5
48. Xu, H., Zhu, T., Zhang, L., Zhou, W., Yu, P.S.: Machine unlearning: A survey (2023) 4, 5
49. Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C.A., Li, B.: Detecting ai trojans using meta neural analysis (2020) 4
50. Zeng, Y., Pan, M., Just, H.A., Lyu, L., Qiu, M., 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. pp. 771–785 (2023) 3, 4, 7, 10, 11