Neurocomputing 614 (2025) 128752

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Correcting the distribution of batch normalization signals for Trojan mitigation

Xi Li [a,c,d,1], Zhen Xiang [b,c,1], David J. Miller [c,d,*], George Kesidis [c,d]

[a] Department of Computer Science, The University of Alabama at Birmingham, Birmingham, 35294, AL, USA
[b] School of Computing, University of Georgia, Athens, 30602, GA, USA
[c] School of Electrical Engineering and Computer Science, The Pennsylvania State University, University Park, 16802, PA, USA
[d] Anomalee Inc., State College, 16803, PA, USA

ARTICLE INFO

Communicated by D. Wu

Keywords:
Backdoor (Trojan) attacks
Backdoor mitigation
Adversarial learning
Deep learning

ABSTRACT

Backdoor (Trojan) attacks represent a significant adversarial threat to deep neural networks (DNNs). In such attacks, the presence of an attacker's backdoor trigger causes a test instance to be misclassified into the attacker's chosen target class. Post-training mitigation methods aim to rectify these misclassifications, ensuring that poisoned models correctly classify backdoor-triggered samples. These methods require the defender to have access to a small, clean dataset and the potentially compromised DNN. However, most defenses rely on parameter fine-tuning, making their effectiveness dependent on the dataset size available to the defender. To overcome the limitations of existing approaches, we propose a method that rectifies misclassifications by correcting the altered distribution of internal layer activations of backdoor-triggered instances. Distribution alterations are corrected by applying simple transformations to internal activations. Notably, our method does not modify any trainable parameters of the DNN, yet it achieves generally good mitigation performance against various backdoor attacks and benchmarks. Consequently, our approach demonstrates robustness even with a limited amount of clean data, making it highly practical for real-world applications. The effectiveness of our approach is validated through both theoretical analysis and extensive experimentation. The appendix is provided as an electronic component and can be accessed via the link in the footnote.[2] The source codes can be found in the link[3] at the footnote.

1. Introduction

Deep neural networks (DNN) have shown impressive performance in many applications, but are vulnerable to adversarial attacks. Recently, backdoor (Trojan) attacks have been proposed against DNNs used for image classification [1–6], speech recognition [7], text classification [8], point cloud classification [9], and even deep regression [10]. The attacked DNN will, with high probability, classify to the attacker's target class when a test instance is embedded with the attacker's backdoor trigger. Moreover, this is achieved while maintaining high accuracy on backdoor-free instances. Typically, a backdoor attack is launched by poisoning the training set of the DNN with a few instances embedded with the trigger and (mis)labeled to the target class.

Most existing works on backdoors either focus on improving the stealthiness of attacks [11,12], their flexibility for launching [13,14], their adaptation for different learning paradigms [15–17], or develop defenses for different practical scenarios [18–22]. However, there are few works which study the basic properties of backdoor attacks. Tran et al. [23] first observed that triggered instances (labeled to the target class) are separable from clean target class instances in a feature space consisting of internal layer activations of the poisoned classifier. This property led to defenses that detect and remove triggered instances from the poisoned training set [24,25]. As another example, Zhang et al. [26] studied the differences between the parameters of clean and attacked classifiers, which inspired a stealthier attack with minimum degradation in accuracy on clean test instances.

In this paper, we investigate an interesting *distribution alteration* property of backdoor attacks. In short, the learned backdoor trigger causes a change in the distribution of internal activations for test instances with the trigger, compared to that for backdoor-free instances; and we theoretically demonstrate that **instances with the trigger are**

* Corresponding author at: School of Electrical Engineering and Computer Science, The Pennsylvania State University, University Park, 16802, PA, USA.
E-mail addresses: xli7@uab.edu (X. Li), zxiangaa@uga.edu (Z. Xiang), djmiller@engr.psu.edu (D.J. Miller), gik2@psu.edu (G. Kesidis).

[1] Equal contribution.

[2] https://arxiv.org/pdf/2308.09850

[3] https://github.com/lixi1994/BNA

https://doi.org/10.1016/j.neucom.2024.128752
Received 3 July 2023; Received in revised form 27 June 2024; Accepted 19 October 2024
Available online 7 November 2024

classified to their original source classes after such distribution alteration is reversed/corrected, with trainable parameters of the poisoned model *untouched*. Accordingly, we propose a method to mitigate backdoor attacks (post-training), such that classification accuracy on test instances both with and without the trigger will be close to the test set accuracy of a clean (backdoor-free) classifier. In particular, we correct distribution alteration by exploiting estimated triggers reverse-engineered by a post-training backdoor detector, *e.g.*, [27,28]. Thus, we propose a "detection-before-mitigation" defense strategy, where we first detect if a given model is backdoor-poisoned, and if so, mitigate the model with the target class(es) and the associated trigger(s) estimated by the post-training detector.

It is important to distinguish methods that focus on backdoor **mitigation** from methods which focus on backdoor **detection**. Examples of the latter, including [27–31], typically detect whether a given model is backdoor poisoned, and, if so, infer the target class(es) of the attack. Some detection methods (*e.g.*, the ones proposed by [27–29]) are reverse-engineering based detectors, which also estimate the backdoor trigger(s) associated with the inferred target class(es). However, if an attack is detected, these detection methods may not be able to tell whether an instance (input test sample) that is classified to the inferred target class contains the trigger; moreover, these methods do not infer the (true) original class for a backdoor-trigger image. The goals of a backdoor mitigation method are: (i) to reduce the number of backdoor-trigger test instances mis-classified to the target class(es); (ii) to correctly classify these backdoor-trigger instances and (iii) while maintaining relatively high accuracy on clean test instances.

Compared with existing mitigation approaches, which require tuning all of the DNN's (deep neural network's) parameters, our method achieves generally better performance and does so without changing *any* original, *trainable* parameters of the DNN. Also, some mitigation methods are applied irrespective of whether backdoor poisoning is detected. These methods may unacceptably degrade clean test accuracy, and do so even when the DNN is clean (attack-free). By contrast, our mitigation is performed only *after* a backdoor attack is detected (*i.e.*, "detection-before-mitigation"), and results in only modest drops in accuracy on clean test instances. Moreover, while most mitigation approaches are designed to correctly classify backdoor-trigger instances without detecting whether these samples in fact contain triggers, our method not only corrects the decisions for these instances but also makes explicit inference of whether a test instance possesses a trigger. Our main **contributions** in this paper are summarized as follows:

1. We are the first to theoretically prove that the degradation in classification accuracy on backdoor-triggered instances monotonically correlates with the divergence between the distributions of clean and backdoor-triggered samples.
2. Accordingly, we propose a novel mitigation method that rectifies backdoor misclassifications by correcting the altered distribution of internal layer activations in backdoor-triggered instances. Unlike most existing defense methods that typically involve parameter fine-tuning, our approach does not alter any trainable parameters and solely relies on applying simple transformations to the internal activations to correct the distribution alteration.
3. The effectiveness of our proposed mitigation method has been validated through extensive experiments across various backdoor attacks and benchmarks. Moreover, our method demonstrates greater robustness, particularly beneficial when the defender has access to only a limited number of clean instances, as it does not alter any trainable model parameters. This feature makes it especially suitable for practical defense scenarios.

2. Related work

There are few prior works analyzing the basic properties of backdoor attacks, *e.g.*, the studies conducted by [23,26]. Tran et al. [23] observed that triggered instances (labeled to the target class) are separable from clean *target* class instances, in a feature space consisting of internal layer activations of the poisoned classifier. They accordingly developed a *pre-training* backdoor detection system, where the detected backdoor-trigger instances are removed, and a new model is trained from scratch on the sanitized dataset. [23] thus acts like an *outlier detection* system. In contrast, we observe that backdoor attacks cause distribution alteration, in internal layers of the DNN, between clean *source* class instances and backdoor-trigger instances originating from the same (source) class. Moreover, we demonstrate that backdoor-trigger instances are correctly classified to their classes once this distribution alteration is corrected. We thus propose a *post-training* backdoor *mitigation* method based on these findings. In the post-training scenario, one often assumes the defender only has access to the given trained model and to a *small* set of clean instances, which generally does not include any of the training samples. This small clean set is inadequate for (from scratch) training an accurate, attack-free classifier.

Existing backdoor defenses are deployed either during the DNN's training stage or post-training (but pre-deployment). The ultimate goal of training-stage defenses is to train an accurate, backdoor-free DNN given the possibly poisoned training set. To achieve this goal, methods, such as [18,24,25,32–34], either identify a subset of "high-credible" instances for training, or detect and then remove, prior to model learning, training instances that may contain a backdoor trigger. Post-training defenders, however, are assumed to have *no* access to the classifier's training set. Many post-training defenses aim to detect whether a given classifier has been backdoor-compromised. Methods, such as [19,27,28,35], perform anomaly detection using triggers reverse-engineered on an assumed independent clean dataset; while [36,37] train a (binary) meta classifier using "shadow" classifier "exemplars" trained with and without attack.

However, model-detection defenses are not able to mitigate backdoor attacks at test time. Thus, there is a family of post-training backdoor mitigation approaches proposed to fine-tune the classifier on the available clean dataset. Neural Cleanse [27] is the first to mitigate backdoor attacks using the backdoor attack detection results (*i.e.*, the detected target class and the associated estimated backdoor patterns). They patch the poisoned DNN by fine-tuning the DNN on 10% of the original (backdoor-free) training set, 20% of which are embedded with the reverse engineered backdoor pattern and correctly labeled. However, it is unreasonable to assume the defender has access to the clean training set, which is inconsistent with the post-training scenario.

Some methods prune neurons that may be associated with the backdoor attack [38–41]. Liu et al. [38] recognize and prune backdoored neurons as those are dormant on clean inputs, and then fine-tune model parameters. The subsequent pruning based backdoor mitigation methods work on optimizing the strategy for recognizing backdoored neurons. Wu et al. [39] propose Adversarial Neuron Pruning (ANP), where they simulate backdoor perturbation by perturbing neurons' weight and bias and identify backdoored neurons as the ones that are sensitive to adversarial neuron perturbations. The neuron perturbations are optimized to maximizing the classification loss.

Others leverage knowledge distillation to preserve the classification function only for clean instances [42,43]. Li et al. [42] propose a mitigation framework named Neural Attention Distillation (NAD). They first fine-tune the poisoned DNN on a small set of clean data, then utilize this DNN as a teacher network to guide the fine-tuning of the poisoned student network on the same clean dataset. The fine-tuning process aims to align the internal layer attention of the student network with that of the teacher network. Xia et al. [43] improve the framework NAD by aligning Attention Relation Graphs (ARG) between teacher

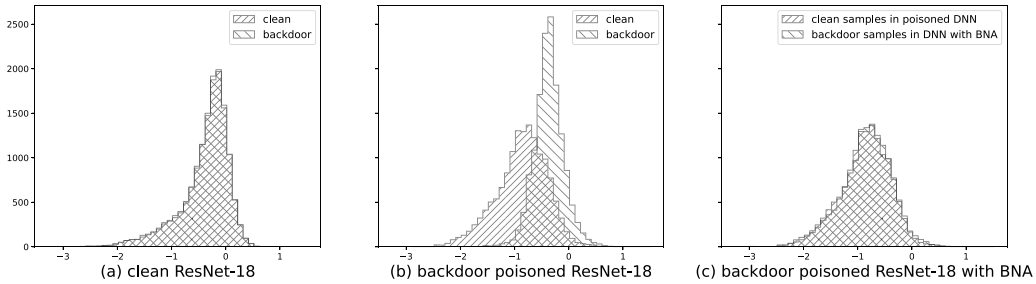

Fig. 1. Activation distribution of a neuron in the penultimate layer of ResNet-18 trained on CIFAR-10, for instances with and without a backdoor trigger, for (a) a clean classifier and (b) a backdoor-poisoned classifier (with the same trigger). In (c), the distribution alteration in (b) is reversed by our proposed method — most instances with the trigger will thus be correctly classified.

and student models during knowledge distillation. ARG fully considers the correlation of attention features of different orders, while NAD only compares the feature attentions of the same order during the fine-tuning.

Some solve a min–max optimization problem analogous to adversarial training defenses used against test-time evasion attacks [44,45]. In [44], the inner maximization problem, to align with the goal for the backdoor attacker, aims to find a trigger that causes a high loss for predicting the correct label. By contrast, the outer minimization problem is to optimize the model parameters so that the "adversarial loss" is minimized. These defenses usually incur a significant degradation in the classifier's accuracy on clean instances, especially when the clean data available for classifier fine-tuning is insufficient.

Another family of approaches are designed to catch the adversarial entities in the act, without altering the classifier [21,22,46]. A STRong Intentional Perturbation (STRIP) method, proposed in [22], is based on the observation that predictions of perturbed triggered inputs remain consistent across different perturbing patterns, unlike predictions of perturbed clean inputs, which vary significantly. Consequently, triggered inputs exhibiting consistently low entropy and clean inputs showing high entropy can be distinctly identified. STRIP method involves blending test images with a few clean images and detecting backdoor triggers by evaluating the average entropy of the model's posterior probabilities for these blended images. A low entropy value signals the presence of a backdoor-triggered input.

Most existing backdoor mitigation methods apply mitigation *independently* of detection, *e.g.*, [39,42–44,47]. That is, they apply a mitigation method on a given model without knowing whether it is backdoor poisoned, with the expectation that the mitigation method should work well regardless of the target class(es) and associated backdoor triggers(s). However, mitigation may harm the model's accuracy on clean instances, especially when mitigation is based only on a limited amount of clean labeled data, which is common in practice, see Table 4. Moreover, mitigation may waste significant computation if the given model is attack-free. Hence, we argue that mitigation should be conducted within a "detection-before-mitigation" framework. In other words, one should perform backdoor mitigation only if the given model is detected as backdoor-poisoned. This avoids a significant drop in accuracy on clean instances after mitigation (in the case where the DNN is backdoor-free). Combined with a backdoor detection method (which may be based on embedded feature activations), our proposed mitigation method is also applicable when *multiple* backdoor attacks are encoded in the DNN.

3. Distribution alteration property of backdoor attacks

In this section, we first present the *activation distribution alteration property* of backdoor attacks. Then for a simplified setting, we analytically show how closing the "gap" between the clean-instance and backdoor-trigger instance distributions improves the accuracy in classifying backdoor-trigger instances; this will guide the design of our backdoor mitigation approach in Section 4.

Property 3.1 (*Activation Distribution Alteration*). *For a successful backdoor attack, different test samples embedded with the backdoor trigger will induce perturbations to the activations of an internal DNN layer that are in a similar direction. Thus, there is effectively a "shift" in the internal layer activation distribution for backdoor-trigger instances, compared to that for backdoor-free instances.*

This property is easily demonstrated empirically, visually. Consider a set of clean instances from CIFAR-10 [48] and the *same* set of instances but with the backdoor trigger used by [1] embedded in each instance. For a ResNet-18 [49] classifier that was successfully attacked using this trigger, there is a *divergence* between the distributions of the internal layer activations induced by these two sets of instances. This is shown in Fig. 1b for a neuron in the penultimate layer as an example. In comparison, for a clean classifier (not backdoor-attacked), the divergence between the two distributions is almost negligible as shown in Fig. 1a. Based on these visualizations, we ask the following question: *Suppose the distribution alteration is* ***reversed*** *for each neuron, e.g., by applying a transformation to the internal activations of the triggered instances, so that the transformed distribution now closely agrees with the distribution for clean (without the backdoor-trigger) instances (see Fig. 1c). Then, following this compensation, will the classifier accurately predict the true class of origin for these backdoor-trigger instances?*

Here, we investigate this problem in a simplified binary classification setting similar to the one considered by [50]. For a clean training random vector $(\mathbf{X}, Y)$ with a uniform class prior, *i.e.*$Y \sim \mathcal{U}\{-1,+1\}$ and with $\mathbf{X}|Y \sim \mathcal{N}(Y \cdot \boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu} \in \mathbb{R}^d$ and $\Sigma = \sigma^2 \boldsymbol{I}$, consider a backdoor attack with *target class* '+1', *triggered instance* $\mathbf{X}_b \sim \mathcal{N}(\boldsymbol{\mu}_b, \Sigma_b)$ with $\boldsymbol{\mu}_b = -\boldsymbol{\mu} + \boldsymbol{\epsilon}$, and $\Sigma_b = \sigma_b^2 \boldsymbol{I}$. Here, class '−1' is automatically the *source class* of $\mathbf{X}_b$ since there are only two classes.

With backdoor poisoning, a multi-layer perceptron (MLP) classifier is trained with one hidden layer of J nodes, a batch normalization (BN) layer[4] [51] followed by linear activation, and two output nodes with functions $f_- : \mathbb{R}^d \to \mathbb{R}$ and $f_+ : \mathbb{R}^d \to \mathbb{R}$ corresponding to classes '−1' and '+1' respectively. An instance $\boldsymbol{x}$ will be classified to class '−1' if $f_-(\boldsymbol{x}) > f_+(\boldsymbol{x})$; else it will be classified to '+1'.

Definition 3.1 (*η-Erroneous Classifier*). A classifier is said to be η-erroneous if the error rate for each class is upper bounded by η.

Definition 3.2 (*ψ-Successful Attack*). A backdoor attack is said to be ψ-successful if its attack success rate (ASR), *i.e.*, the probability for triggered instances being (mis)classified to the attacker's target class [52], is at least ψ; in our case, this means that $P[f_+(\mathbf{X}_b) > f_-(\mathbf{X}_b)] \geq \psi$.

[4] Here we utilize the transformations in the BN layer to reverse the distribution alteration for simplicity. Our method does not truly rely on the existence of BN layers in the trained network, as one can always insert a BN layer between any two layers of (an already trained) network.

Given the settings above, for an arbitrary input $\boldsymbol{x}$, the activation of the jth node ($j \in \{1, \dots, J\}$) (after BN with trained parameters γ_j and β_j), with weight vector $\boldsymbol{w}_j$ in the hidden layer, is:

$$a_j(\boldsymbol{x}) = \frac{\boldsymbol{w}_j^\top \boldsymbol{x} - m_j}{\sqrt{v_j}}\gamma_j + \beta_j, \tag{1}$$

where m_j and v_j respectively are the mean and variance stored by the BN layer during training on the *poisoned training set*. Then the activation distribution for clean source class instances $(\mathbf{X}|Y = -1) \sim \mathcal{N}(-\boldsymbol{\mu}, \Sigma)$ is a Gaussian specified by mean $\mathbb{E}[a_j(\mathbf{X})|Y = -1]$ and variance $\mathrm{Var}[a_j(\mathbf{X})|Y = -1]$; while for triggered instances $\mathbf{X}_b \sim \mathcal{N}(\boldsymbol{\mu}_b, \Sigma_b)$, the activation follows a Gaussian specified by mean $\mathbb{E}[a_j(\mathbf{X}_b)]$ and variance $\mathrm{Var}[a_j(\mathbf{X}_b)]$. An easy way to eliminate the *divergence* between these two distributions is to create a classifier *for triggered instances* $\mathbf{X}_b$[5] by replacing a_j in Eq. (1) with $a_j^*(\boldsymbol{x}) = (\boldsymbol{w}_j^\top \boldsymbol{x} - m_j^*)\gamma_j/\sqrt{v_j^*} + \beta_j$ for each node j, where (see Apdx. A.1 for derivation):

$$m_j^* = \frac{\sigma_b}{\sigma} m_j + (\frac{\sigma_b}{\sigma} - 1)\boldsymbol{w}_j^\top \boldsymbol{\mu} + \boldsymbol{w}_j^\top \boldsymbol{\epsilon} \quad \text{and} \quad v_j^* = \frac{\sigma_b}{\sigma} v_j. \tag{2}$$

With these choices, $\mathbb{E}[a_j^*(\mathbf{X}_b)] = \mathbb{E}[a_j(\mathbf{X})|Y = -1]$ and $\mathrm{Var}[a_j^*(\mathbf{X}_b)] = \mathrm{Var}[a_j(\mathbf{X})|Y = -1]$ are achieved. But here, we aim to study the quantitative relationship between the distribution divergence and the SIA metric of Definition 3.3 below. Thus, we consider an "intermediate state" with a classifier specified by output node functions $g_-(\cdot|\alpha) : \mathbb{R}^d \to \mathbb{R}$ and $g_+(\cdot|\alpha) : \mathbb{R}^d \to \mathbb{R}$, where for each output node $i \in \{-, +\}$, $g_i(\boldsymbol{x}|\alpha) = \boldsymbol{u}_i^\top \hat{\boldsymbol{a}}(\boldsymbol{x}|\alpha)$ depends on a "transition variable" $\alpha \in [0, 1]$, with $\boldsymbol{u}_i$ the weight vector for the original output function f_i. $\hat{\boldsymbol{a}}(\boldsymbol{x}|\alpha) = [\hat{a}_1(\boldsymbol{x}|\alpha), \dots, \hat{a}_J(\boldsymbol{x}|\alpha)]^\top$ is the activation vector for input $\boldsymbol{x}$ where $\hat{a}_j(\boldsymbol{x}|\alpha) = (\boldsymbol{w}_j^\top \boldsymbol{x} - \hat{m}_j(\alpha))\gamma_j/\sqrt{\hat{v}_j(\alpha)} + \beta_j$, with $\hat{m}_j(\alpha) = \alpha m_j + (1-\alpha)m_j^*$ and $\hat{v}_j(\alpha) = (\alpha\sqrt{v_j} + (1-\alpha)\sqrt{v_j^*})^2$ being the "intermediate" mean and variance respectively. Given these settings, our main theoretical results are presented below.

Definition 3.3 (*Source Inference Accuracy (SIA)*)**.** SIA is the probability that a triggered instance is classified to its original source class [53], *i.e.*, $P[g_-(\mathbf{X}_b|\alpha) > g_+(\mathbf{X}_b|\alpha)]$.

Theorem 3.1 (*Monotonicity of SIA with Divergence*)**.** *If the binary classifier with f_- and f_+ is η-erroneous with $\eta < 1/2$, the attack is ψ-successful with $\psi > 1/2$, and $\sigma_b \leq \sigma$, then SIA of the modified classifier, i.e., $P[g_-(\mathbf{X}_b|\alpha) > g_+(\mathbf{X}_b|\alpha)]$, monotonically decreases as $\alpha \in [0, 1]$ increases.*

The proof of the theorem is given in Apdx. A.2. Note that the assumptions for Theorem 3.1 are very mild and reasonable. For example, $\eta < 1/2$ is a minimum requirement for the classifier and $\psi > 1/2$ is a minimum requirement for a successful backdoor attack. Moreover, $\sigma_b \leq \sigma$ generally holds empirically since trigger embedding (*e.g.*, consider a patch attack) typically reduces the variance of source class instances (while additive attacks do not change the variance). Also note that α merely gives a way of quantifying distribution divergence for purpose of analysis. According to these results, the core part of our proposed backdoor mitigation approach should be to find a modified classifier $g(\cdot|\boldsymbol{\Theta})$ by minimizing (*e.g.*, using sub-gradient methods) a measure of distribution divergence over a well-chosen *set* of parameters, $\boldsymbol{\Theta}$. This approach is next explicated.

4. Reversing distribution alteration for backdoor mitigation

4.1. Problem description

Threat model. For input space $\mathcal{X}$ and label space C, a classifier that has been successfully backdoor-attacked will predict to the attacker's target class $t^* \in C$ when a test instance $\boldsymbol{x} \in \mathcal{X}$ is embedded with the backdoor trigger using an incorporation function $\Delta : \mathcal{X} \to \mathcal{X}$. In addition to this "*all-to-one*" setting, we also consider the "*all-to-all*" setting where a test instance from any class $c \in C$ will be (mis)classified to class $(c + 1) mod |C|$ when it is embedded with the trigger [1].

Defender's goals. Given a trained classifier $f : \mathcal{X} \to C$ that may possibly be attacked, the defender aims to mitigate possible attacks by producing a mapping $\hat{f} : \mathcal{X} \to C$ which (a) has high accuracy in classifying clean instances; (b) when there is a backdoor attack, classifies triggered instances to their original source class, as though there is no trigger embedded, *i.e.*, achieves a high SIA; and (c) *detects* whether or not a test sample contains a backdoor trigger.

Defender's assumptions. We consider a *post-training* scenario where the defender has *no access* to the training set of the classifier. The defender does possess an independent clean dataset, but this dataset is *too small* to train an accurate classifier from scratch, and even too small to effectively fine-tune the full set of classifier parameters [27,38,44]. The defender has full (white box) access to the classifier, but does not know whether it has been attacked and, if so, does not know the trigger pattern that was used, *i.e.*, the defense is unsupervised — we will leverage existing post-training detectors to determine if the classifier was attacked and, if so, to estimate the target class(es) of the attack and the backdoor trigger.

The "detection-before-mitigation" scenario: We propose that backdoor detection should generally be performed before mitigation. That is, one should first apply a backdoor detection method on the given model and perform backdoor mitigation on it *only* if it is detected as backdoor poisoned. Otherwise, backdoor mitigation may harm the accuracy of the model and is a waste of computation if there is no attack. On the other hand, if there is an attack, utilizing the detection results (*e.g.*, the detected target class(es)) helps to reduce the degradation in the classifier's accuracy on clean test data brought about by mitigation. (See the experimental results of Section 5.3 and Table 10.) Our method indeed mitigates only when a backdoor is detected, and exploits knowledge of the detected target class(es), as well as the estimated backdoor trigger, produced by post-training detectors such as [27,28].

4.2. Method

Key idea: The principle behind our mitigation method is simple: A backdoor-trigger instance will be correctly classified to its original source class by the poisoned model if the model is altered in such a way as to follow the same distribution as the clean source class instances in each internal layer feature space of the model (as shown in Fig. 1 and proved by Theorem 3.1). For this purpose there is *no need to modify the trainable parameters*. As demonstrated in Fig. 2, we align distributions through simple transformations, *e.g.*, those used in batch normalization, on internal layer feature maps, produced for clean samples that are embedded with the estimated backdoor trigger (heretofore referred to as "backdoor-trigger instances"). The transformation parameters are optimized by minimizing the divergence between the distributions of clean instances and triggered instances. To illustrate the distribution correction, consider two distributions, p and q. Our objective is to align q with p using a transformation h applied to q. This transformation, parameterized by θ, is defined as follows:

$$h(q|\theta) = \max(\min(\frac{q - \mu}{\sigma}, w), v).$$

Here, μ and σ are the mean and standard deviation, while w and v define the upper and lower saturation limits. The transformation parameters $\theta = \{\mu, \sigma, w, v\}$ are optimized by minimizing the, *e.g.*, Kullback–Leibler divergence between p and the transformed distribution $h(q|\theta)$:

$$\arg\min_\theta D_{KL}(p \parallel h(q|\theta)).$$

[5] These can be constructed in practice, given an estimated backdoor trigger (obtained by applying a reverse-engineering based backdoor detector, *e.g.*, [27, 28]), by embedding the trigger in clean instances available to the defender.

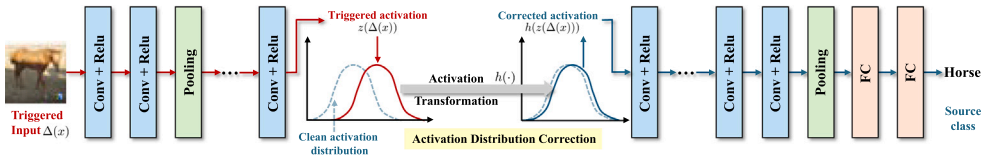

Fig. 2. Demonstration of activation distribution correction.

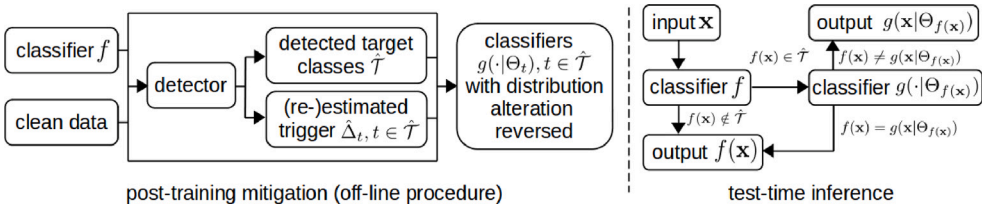

Fig. 3. Illustration of our backdoor mitigation framework with a test-time inference rule.

To approximate the distributions, we calculate the histogram of (transformed) feature maps in each internal layer. Our mitigation framework, along with the test-time detection rule, is visually summarized in Fig. 3. A detailed explanation will follow the introduction of our mitigation strategy.

Now we elaborate our mitigation strategy, which is also summarized in Fig. 2 and Algo. 1. Based on Theorem 3.1, it would seem that a good mitigation approach involves modifying the classifier f, *i.e.*, creating a new classifier $g(\cdot|\boldsymbol{\Theta}) : \mathcal{X} \to C$ from f by applying a transformation function $h_{j,l}(\cdot|\theta_{j,l}) : \mathbb{R} \to \mathbb{R}$ to the activation of each neuron $j \in \{1, \dots, J_l\}$ in each layer $l \in \{1, \dots, L\}$. The transformation parameters $\boldsymbol{\Theta} = \{\theta_{j,l}\}$ should be jointly chosen so as to minimize the aggregation (*e.g.*, sum) of the divergences between the distributions $q_{j,l}(\theta_{<l} \cup \theta_{j,l})$ obtained using $h_{j,l}(z_{j,l}(\Delta(\mathbf{X})|\theta_{<l})|\theta_{j,l})$ (*i.e.*, distributions of the transformed activations of backdoor-trigger samples) and the target distributions $p_{j,l}$ for $z_{j,l}(\mathbf{X})$ (*i.e.*, distributions of the activations of clean samples) for $\forall j, l$, where $\mathbf{X}$ follows the clean data distribution, *i.e.*:

$$\underset{\boldsymbol{\Theta}=\{\theta_{j,l}\}}{\text{minimize}} \quad \sum_{j,l} D_k\big(p_{j,l} || q_{j,l}(\theta_{<l} \cup \theta_{j,l})\big) \tag{3}$$

where: $z_{j,l} : \mathcal{X} \to \mathbb{R}$ is the activation functions for neuron j in layer l for DNN f; $\theta_{<l} = \{\theta_{j,l'} | l' < l\}$ represents all transformation parameters prior to layer l; $D_k(p||q) := \mathbb{E}_q[k(p/q)]$ for a convex function $k : [0, \infty) \to \mathbb{R}$ satisfying $k(1) = 0$ and belonging to the family of f-divergences for any distributions p and q [54].

However, in practice, we have the following challenges. **Challenge 1**: Unlike the distributions of clean samples $\{p_{j,l}\}$, which can be simply approximated by feeding the small number of clean samples possessed by the defender to the poisoned model[6] and calculating the histograms of internal activations, the distributions of backdoor-trigger samples $\{q_{j,l}\}$ are unknown. **Challenge 2**: The density form for the activation of backdoor-trigger samples $z_{j,l}(\Delta(\mathbf{X}))$ may get altered by the trigger Δ and will likely be different from the density form for the activation of clean samples $z_{j,l}(\mathbf{X})$; moreover, both will likely be non-Gaussian. Thus, minimizing Eq. (3) is not trivial. One cannot align the distributions by *e.g.*, simply matching the mean and variance.

To address Challenge 1, we approximate the distributions of true backdoor-triggers samples by those of defender's samples that are embedded with the trigger(s) estimated by a post-training detector. This can be accomplished with widely used post-training reverse-engineering based backdoor detection (RED) approaches which have the same assumption as in Section 4.1, *e.g.*, the ones proposed by [19, 27,28,55]. These REDs investigate whether the classifier f is compromised by a backdoor attack and if so, infer the source and target classes and estimate the associated backdoor trigger(s).[7]

To solve a broad range of attack settings, *e.g.*, all-to-one and all-to-all attacks, we apply the detection methods in a general way following [28]. We first reverse-engineer a trigger by solving an optimization problem defined on the clean set to get a detection statistic for each ordered *putative class pair* $(s,t) \in C \times C$. A statistic which [28] suggests is (the reciprocal of) the estimated perturbation size inducing high (mis)classifications from s to t. For [27], it is the estimated patch size inducing high (mis)classifications from s to t. Then we apply the anomaly detection approach in [27], based on the MAD criterion [56], to all the obtained statistics to find *all* the *outlier* statistics. We denote the set of detected class pairs associated with these outlier statistics as $\hat{\mathcal{P}}$, and denote $\hat{\mathcal{T}} = \{t \in C \mid \exists s \in C \text{ s.t. } (s,t) \in \hat{\mathcal{P}}\}$ as the set of detected target classes.

For *each* $t \in \hat{\mathcal{T}}$, we (re-)estimate a trigger $\hat{\Delta}_t$ (as a *surrogate* for the true backdoor trigger, which is unknown) using clean instances from *all* detected source classes[8] $\hat{S}(t) = \{s \in C | (s,t) \in \hat{\mathcal{P}}\}$. Then, *for each detected target class* $t \in \hat{\mathcal{T}}$, we construct a classifier $g(\cdot|\boldsymbol{\Theta}_t)$ by solving the distribution divergence minimization problem using the (re-)estimated $\hat{\Delta}_t$.

Now we address Challenge 2, which is *critical* to the estimation of $\boldsymbol{\Theta}_t$ using the reverse-engineered trigger $\hat{\Delta}_t$ for each detected target class $t \in \hat{\mathcal{T}}$. For simplicity, we will consider one target class and drop the subscript t below without loss of generality. Our main goals are: (a) specifying the structure of the transformation function $h_{j,l}$ with its associated parameters $\theta_{j,l}$, (b) empirical estimation of the distribution divergence in Eq. (3) using a clean dataset (*i.e.*, the subset of clean instances from classes in $\hat{S}(t)$ for each detected class t), and (c) choosing the convex function k to specify the divergence form. For (a), we consider the following transformation function with parameters $\theta_{j,l} = \{\mu_{j,l}, \sigma_{j,l}, \upsilon_{j,l}, \omega_{j,l}\}$:

$$h_{j,l}(z) = \max\{\min\{\frac{z - \mu_{j,l}}{\sigma_{j,l}}, \omega_{j,l}\}, \upsilon_{j,l}\} \tag{4}$$

where $\mu_{j,l}$ and $\sigma_{j,l}$ specify the location and scale of the activation distribution, respectively, while $\upsilon_{j,l}, \omega_{j,l}$ control the shape of the tail of the distribution. For goal (b), we quantize the real line into M intervals $\mathcal{I}_1 = (-\infty, b_1), \mathcal{I}_2 = [b_1, b_2), \dots, \mathcal{I}_M = [b_{M-1}, \infty)$, for M sufficiently large. Then the distribution divergence in Eq. (3) for each node j and layer l

[6] As previously discussed, mitigation methods should only be applied to a model if it has been detected as poisoned.

[7] Note that these REDs can be the backdoor detectors used prior to applying mitigation methods. Thus, there is no additional computation cost involved.

[8] More reliable trigger estimation can be achieved in this way for a detected target class, compared with estimating the trigger based on only one (source, target) class pair.

is computed on discrete distributions $\hat{p}_{j,l}$ and $\hat{q}_{j,l}$ over these intervals. Specifically, the discrete distributions are estimated using a subset $\mathcal{D}_t$ of instances from classes $\hat{\mathcal{S}}(t)$, with the probabilities for interval $\mathcal{I}_i$ computed by:

$$\hat{p}^{(i)}_{j,l} = \frac{1}{|\mathcal{D}_t|}\sum_{\boldsymbol{x}\in\mathcal{D}_t}\mathbb{1}[z_{j,l}(\boldsymbol{x})\in\mathcal{I}_i] \text{ and}$$
$$\hat{q}^{(i)}_{j,l} = \frac{1}{|\mathcal{D}_t|}\sum_{\boldsymbol{x}\in\mathcal{D}_t}\mathbb{1}[h_{j,l}(z_{j,l}(\hat{\Delta}_t(\mathbf{X})|\boldsymbol{\theta}_{<l})|\boldsymbol{\theta}_{j,l})\in\mathcal{I}_i]. \quad (5)$$

To ensure that the distribution divergence is differentiable with reference to the parameters, such that it can be minimized using (*e.g.*,) gradient descent, we approximate the non-differentiable indicator function $\mathbb{1}[\cdot]$ in Eq. (5) using differentiable functions such as the sigmoid, *i.e.*, we redefine:

$$\mathbb{1}[z\in\mathcal{I}_i] = sigmoid(\tau(z-b_{i-1})) - sigmoid(\tau(z-b_i)) \quad (6)$$

where τ is a scale factor controlling the error of approximation. For $\mathcal{I}_1$ and $\mathcal{I}_M$, which have semi-infinite support, we use a single sigmoid in Eq. (6). The choice of the intervals and τ is not critical to the performance, as long as the length of the finite intervals is sufficiently small, as will be shown in Table 8 in Section 5. Finally, for goal (c), we consider several different divergence forms including the total variation (TV) divergence with $k(r) = |r-1|/2$, the Jensen–Shannon (JS) divergence with $k(r) = r\log\frac{2r}{r+1} + \log\frac{2}{r+1}$, and the Kullback–Leibler (KL) divergence with $k(r) = r\log r$. The choice of the divergence form is also not critical to the mitigation performance (see Apdx. E).

We now provide a detailed explanation of our backdoor mitigation framework, which is visually summarized in Fig. 3. For any test input $\boldsymbol{x}\in\mathcal{X}$, if classifier f is deemed attack-free, *i.e.*, $\hat{\mathcal{P}} = \emptyset$, the classification output under our mitigation framework will be $\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x})$. Otherwise, if $f(\boldsymbol{x})\in C\setminus\hat{\mathcal{T}}$, we trust the class decision and set $\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x})$ both because $\boldsymbol{x}$ is unlikely to possess a trigger and because a successful attack should not degrade the classifier's accuracy on clean instances. However, if $f(\boldsymbol{x}) = t\in\hat{\mathcal{T}}$, there are two main possibilities: (1) $\boldsymbol{x}$ is a clean instance truly from class t; (2) $\boldsymbol{x}$ is classified to class t due to the presence of the trigger. To distinguish these two cases, we feed $\boldsymbol{x}$ to the optimized $g(\cdot|\boldsymbol{\Theta}_t)$. If $g(\boldsymbol{x}|\boldsymbol{\Theta}_t)\neq f(\boldsymbol{x})$, $\boldsymbol{x}$ likely contains a trigger, and thus we should set $\hat{f}(\boldsymbol{x}) = g(\boldsymbol{x}|\boldsymbol{\Theta}_t)$, which is likely the original source class of $\boldsymbol{x}$ based on our theoretical results. Note that in the test-time inference procedure above, the major (additional) computation for both backdoor trigger instance detection and source class inference is a forward propagation, feeding $\boldsymbol{x}$ to $g(\cdot|\boldsymbol{\Theta}_t)$, which is comparable to the computation required for classification using f. Moreover, such additional computation occurs only if an attack is detected and $f(\boldsymbol{x}) = t$; thus, our test-time inference is very efficient.

Algorithm 1: Backdoor Mitigation by Activation Distribution Correction

Input: The detected target classes $\hat{\mathcal{T}}$, the estimated trigger embedding function $\hat{\Delta}_t$, $\forall t\in\hat{\mathcal{T}}$, the detected source classes $\hat{\mathcal{S}}(t)$, $\forall t\in\hat{\mathcal{T}}$, the number of activation distribution intervals M, the clean dataset $\mathcal{D}$, and the poisoned DNN $f(\cdot)$.

Output: The activation transformation function parameters Θ^t, $\forall t\in\hat{\mathcal{T}}$.

for *Each detected target class* $t\in\hat{\mathcal{T}}$ **do**
  Select the subset $\mathcal{D}_t$ of instances from source classes $\hat{\mathcal{S}}(t)$ from the clean dataset $\mathcal{D}$: $\mathcal{D}_t = \{(x,y)\in\mathcal{D}|y\in\hat{\mathcal{S}}(t)\}$.
  Embed the estimated backdoor trigger to instances of $\mathcal{D}_t$: $\tilde{\mathcal{D}}_t = \{(\hat{\Delta}_t(x),y)|(x,y)\in\mathcal{D}_t\}$.
  for *Each layer* $l = 1,\dots,L$ *in DNN* $f(\cdot)$ **do**
    for *Each neuron* $j = 1,\dots,J_L$ *in layer* l **do**
      Calculate discrete distributions $\hat{p}_{j,l}$ for clean instances from $\mathcal{D}_t$: $\hat{p}_{j,l} = \{\frac{1}{|\mathcal{D}_t|}\sum_{x\in\mathcal{D}_t}\mathbb{1}[z_{j,l}(\boldsymbol{x})\in\mathcal{I}_i]\}_{i=1,\dots,M}$.
      Calculate discrete distributions $\hat{q}_{j,l}$ for triggered instances from $\tilde{\mathcal{D}}_t$: $\hat{q}_{j,l}(\theta^t_{j,l}) = \{\frac{1}{|\tilde{\mathcal{D}}_t|}\sum_{\tilde{\boldsymbol{x}}\in\tilde{\mathcal{D}}_t}\mathbb{1}[h(z_{j,l}(\tilde{\boldsymbol{x}}|\theta^{t*}_{<l})|\theta^t_{j,l})\in\mathcal{I}_i]\}_{i=1,\dots,M}$.
      Update transformation parameters for neuron j in layer l $\theta^t_{j,l}$ by minimizing the KL divergence between the clean and triggered instances distributions: $\theta^{t*}_{j,l} = \arg\min_{\theta^t_{j,l}} D_{KL}(\hat{p}_{j,l}||\hat{q}_{j,l}(\theta^t_{j,l}))$.
return $\Theta^t = \{\theta^{t*}_{j,l}\}$, $\forall t\in\hat{\mathcal{T}}$

5. Experiments

5.1. Experiment setup

Datasets: Our main experiments are conducted on the benchmark CIFAR-10 dataset, which contains 60,000 32 × 32 color images from 10 classes, with 5000 images per class for training and 1000 images per class for testing [48]. We also show the effectiveness of our proposed mitigation framework on other benchmark datasets including GTSRB [57], CIFAR-100 [48], ImageNette [58], TinyImageNet [59], and VGGFace2 [60]. Details of these datasets can be found in Apdx.B.1. Data allocation in our experiments strictly follows the assumptions in Section 4.1. For each dataset, we randomly sample 10% of the test set to form the small, clean dataset $\mathcal{D}_{\text{Defense}}$ assumed for the defender. The remaining test instances, denoted by $\mathcal{D}_{\text{Test}}$, are reserved for performance evaluation.

Attack settings: In this paper, we consider standard backdoor attacks launched by poisoning the training set of the classifier [1,2]. In particular, we consider both the *all-to-one* (**A2O**) attacks and the *all-to-all* (**A2A**) attacks in our main experiments on CIFAR-10. For A2O attacks on CIFAR-10, we arbitrarily choose class 9 as the target class; while for A2A attacks, as described in Section 4.1, triggered instances from any class $c\in C$ are supposed to be (mis)classified to class $(c+1)mod|C|$. For each attack setting, we consider the following triggers: (1) a 3 × 3 random patch (**BadNet**) with a randomly selected location (fixed for all triggered images for each attack) used in [1]; (2) an additive perturbation (with size 2/255) resembling a chessboard (**CB**) used in [28]; (3) a single pixel (**SP**) perturbed by 75/255 with a randomly selected location (fixed for all triggered images for each attack) used by [23]; (4) invisible triggers generated with l_0 and l_2 norm constraints (**l_0 inv** and **l_2 inv** respectively) proposed by [6]; (5) a warping-based trigger (**WaNet**) proposed by [3]; (6) a Hello Kitty blending trigger (**Blend**) used by [2]; (7) a trigger generated by the horizontal sinusoidal function (**SIG**) defined in [61]. Details for generating these triggers are deferred to Apdx.B.2. We randomly created 5 attacks for each attack setting *e.g.* by randomly locating the trigger. We also evaluated against the "label-consistent" (**CL**) backdoor attack proposed by [62] on the CIFAR-10 dataset, which only embeds the backdoor trigger into the target class training samples. Details are given in Apdx.B.2. For experiments on the other five datasets, we only consider A2O attacks for a subset of triggers where sufficiently high success rate can be achieved. For each dataset, we create one attack for each trigger being considered. A2A attacks are not considered for these datasets since there is insufficient data per class for them to achieve successful attacks. More details about the attacks, including the number of backdoor-trigger images used for poisoning and the target class selected to create A2O attacks for the five datasets other than CIFAR-10, are shown in Apdx.B.2.

Performance evaluation metrics: (1) The attack success rate (**ASR**) is the fraction of clean instances in $\mathcal{D}_{\text{Test}}$ (mis)classified to the designated target class when the backdoor trigger is embedded. (2) The clean test accuracy (**ACC**) is the DNN's accuracy on $\mathcal{D}_{\text{Test}}$ without trigger embedding. (3) The **SIA** (Definition 3.3) is the fraction of clean instances in $\mathcal{D}_{\text{Test}}$ classified to the original source class when the trigger is

Table 1
Average ACC, ASR, and SIA for BNA, compared with NC, NAD, I-BAU, ANP, and ARGD, against all the created attacks applied to ResNet-18 trained on the CIFAR-10 dataset. Best performances are indicated in bold.

Trigger type		BadNet		CB		l_0 inv		l_2 inv		SP		WaNet	
		A2O	A2A	A2O	A2A	A2O	A2A	A2O	A2A	A2O	A2A	A2O	A2A
Vanilla	ACC	0.9122	0.9121	0.9135	0.9098	0.9135	0.9131	0.9130	0.9126	0.9138	0.9060	0.9032	0.8994
	ASR	0.9573	0.8658	0.9685	0.8692	0.9989	0.8973	0.9889	0.8620	0.8912	0.8550	0.9153	0.8216
	SIA	0.0397	0.0432	0.0293	0.0257	0.0010	0.0151	0.0107	0.0194	0.1016	0.0593	0.0772	0.0714
NC	ACC	0.8797	0.8762	0.8735	0.8776	0.8835	0.8767	0.8750	0.8690	0.8854	0.8614	0.8748	0.8756
	ASR	0.0130	**0.0154**	**0.0064**	0.0155	0.0120	0.0150	0.0080	0.0179	0.0335	0.0188	0.0144	0.1381
	SIA	0.8532	0.8614	0.8312	0.8597	0.8654	0.8650	0.7932	0.8254	0.8362	0.8477	0.8231	0.7183
I-BAU	ACC	0.8500	0.8758	0.8812	0.8719	0.8452	0.8800	0.8825	0.8726	0.8666	0.8745	0.8777	0.8700
	ASR	**0.0094**	0.0164	0.1973	0.0811	0.0091	0.0133	0.2600	0.3353	0.0172	**0.0154**	0.1339	0.1253
	SIA	0.8301	0.8583	0.6399	0.7756	0.8277	0.8673	0.5549	0.4928	0.8479	0.8609	0.7059	0.7269
ANP	ACC	0.8644	0.8492	0.8241	0.8577	0.8455	0.8648	0.8345	0.8421	0.8195	0.8411	0.8298	0.8607
	ASR	0.0474	0.1199	0.3351	0.0927	0.0836	0.1326	0.4703	0.2648	0.1229	0.0495	0.0263	0.0835
	SIA	0.8184	0.7205	0.4587	0.7168	0.7697	0.7324	0.3351	0.4976	0.7060	0.7942	0.7368	0.7451
NAD	ACC	0.8814	0.8819	0.8800	**0.8908**	0.8958	**0.9047**	0.8991	**0.8781**	0.8813	0.8761	0.8592	**0.8963**
	ASR	0.0193	0.7132	0.0871	0.0681	0.0356	0.0457	0.0254	0.0191	0.0667	0.0647	0.0571	0.1056
	SIA	0.8498	0.1520	0.7711	0.8084	0.8504	0.8534	0.8221	0.8337	0.8123	0.8082	0.7710	0.7773
ARGD	ACC	0.8689	0.8482	0.8800	0.8774	0.8880	0.8885	0.8669	0.8583	0.8899	0.8728	0.8739	0.8755
	ASR	0.0368	0.0839	0.0099	**0.0117**	0.0079	0.0122	0.0125	0.0179	0.0955	0.0452	0.0111	0.0362
	SIA	0.8217	0.7544	0.8657	0.8690	0.8725	0.8786	0.8168	0.8297	0.7934	0.8295	0.8283	0.8241
MCR	ACC	0.8751	0.8840	0.8126	0.8618	0.8534	0.8834	0.8688	0.8481	0.8886	0.8613	0.8808	0.8661
	ASR	0.1447	0.1001	0.6015	0.1004	0.2900	**0.0000**	0.9769	0.0971	0.0136	0.1166	0.0268	0.0974
	SIA	0.7572	0.0811	0.3151	0.1163	0.0927	0.1000	0.0210	0.3524	0.8570	0.2239	0.7999	0.7789
BNA (ours)	ACC	**0.9032**	**0.8951**	**0.9072**	0.8615	**0.9068**	0.8944	**0.9005**	0.8638	**0.9058**	**0.8921**	**0.8945**	0.8792
	ASR	0.0139	0.0189	0.0127	0.0202	**0.0033**	0.0111	**0.0042**	**0.0168**	**0.0104**	0.0225	**0.0041**	**0.0191**
	SIA	**0.8835**	**0.8841**	**0.8787**	**0.8820**	**0.8924**	**0.8942**	**0.8383**	**0.8522**	**0.8863**	**0.8811**	**0.8530**	**0.8607**

embedded. For a successful backdoor attack, ASR and ACC should be high, while SIA should be low. For a successful mitigation approach, the resulting ASR should be low, while ACC and SIA should be high.

Training settings: We train one classifier for each attack to evaluate our mitigation approach against existing ones. Training configurations, including the DNN architecture, batch size, number of epochs, etc., are detailed in Tab.B.13 in Apdx.B.3. Data augmentation choices, including random cropping and horizontal flipping, are applied to each training instance. As shown in Table 1, the defenseless "vanilla" classifiers being attacked achieve high ACC but suffer high ASR and low SIA (averaged over the five attacks we created) for all trigger types and for both A2O and A2A settings, *i.e.*, the attacks are all successful and hence adequate for performance evaluation.

Hyper-parameter settings: We compare our mitigation approach (named 'Batch Normalization Alteration' (**BNA**) in the sequel) with six well-known and/or state-of-the-art backdoor mitigation methods, including NC [27], NAD [42], I-BAU [44], ANP [39], ARGD [43], and MCR [47]. Besides, we also consider two during-training backdoor defense methods GeodesicAdv [63] and WaveletAdv [64]. For MCR, in their original paper, the defender is assumed to have access to two poisoned models, which may be impractical. Thus, we fine-tune the given model on the defender's dataset and use it as the second model (which is also suggested in their paper). For all these other methods, we used their officially posted code for implementation. For BNA, following Section 4.2, we first perform detection by reverse-engineering a backdoor trigger for each class pair using objective functions from [27,28] and then feed the statistics obtained based on the estimated trigger to an anomaly detector. Our anomaly detector is based on MAD, which is a classical approach also used by [27,35,55]. Here, we set the detection threshold at "7-MAD" which easily catches all the backdoor class pairs. More details, including pattern estimation and detection statistics are shown in Apdx.C. Then, for each detected target class, we solve the divergence minimization problem to optimize the transformation functions using learning rate 0.01 for 10 epochs. Since our mitigation method applies simple transformations which are also used in BN, we consider model structures that contain BN layers (which is very common) for simplicity. But note that the proof of monotonicity of SIA with distribution divergence (Theorem 3.1) and our method (Section 4.2) do not truly rely on the presence of BN layers — one can always insert a BN layer between any two given layers of the trained network. If a neuron is followed by a BN, instead of applying an additional transformation function $h_{j,l}$, we treat the mean and standard deviation of BN as the parameters $\mu_{j,l}$ and $\sigma_{j,l}$ associated with $h_{j,l}$ respectively. We optimize the mean and standard deviation by minimizing distribution divergence for all the BN layers. In Section 5.2, we only show results for BNA with the total variation divergence. Results for KL-divergence and JS-divergence are deferred to Apdx.E. To compute the divergence, we use the "interval trick" (Eq. (5)) to obtain the discrete empirical distribution. For simplicity, we let all finite intervals, $\mathcal{I}_i = [b_{i-1}, b_i), i = 1, \ldots, M$, have the same length $\Delta b = 0.1$. For each neuron, we set $b_{\min}$ and $b_{\max}$ as the minimum and maximum activations, respectively, when feeding in clean instances from $\mathcal{D}_{\text{Defense}}$ to the poisoned classifier f. Then, the number of intervals is $M = \lceil \frac{b_{\max} - b_{\min}}{\Delta b} \rceil$; and all intervals can be specified by $b_0 = b_{\min}$ and $b_i = b_{i-1} + \Delta b$. Finally, the scale factor in Eq. (6) is set to $\tau = 150$, which is obtained by line search to minimize the total variation between the "soft" distribution and the empirical one on $\mathcal{D}_{\text{Defense}}$. In fact, the choices for Δb and τ (over reasonable ranges) have little impact on our mitigation performance, as shown in Table 8.

5.2. Backdoor mitigation results

In Table 1, we show the ASR, ACC, and SIA for BNA compared with the other six methods (which are all DNN tuning-based) for attacks on CIFAR-10. Each metric is averaged over the five attacks created for each trigger type and attack setting, with the highest ACC and SIA, and the lowest ASR in bold. We found that these tuning-based methods are sensitive to the choices of hyper-parameters, such as the learning rate. Hence, for these methods, we optimize the hyper-parameter values to show the *best* results for these methods in Table 1. Although these tuning-based methods (except for MCR) can effectively deactivate backdoor attacks (*i.e.*, significantly reduce ASRs), there is a clear drop (3%–20%) in both ACC and SIA, compared with those for the vanilla DNN (the first row of Table 1). This is possibly due to tuning many

Table 2
ACC, ASR, and SIA for BNA, compared with those of NC, NAD, I-BAU, ANP, and ARGD, against the ResNet-18 trained on the CIFAR-10 dataset poisoned by the label-consistent (CL) backdoor attack.

		Vanilla	NC	I-BAU	ANP	NAD	ARGD	MCR	BNA
CL	ACC	0.9062	**0.9061**	0.1735	0.8998	0.3354	0.2362	0.8829	0.8967
	ASR	0.9304	0.1444	0.4940	0.4632	0.0636	**0.0581**	0.7756	0.0594
	SIA	0.0667	0.7558	0.0896	0.4977	0.2985	0.2349	0.1932	**0.8057**

Table 3
ACC, ASR, and SIA for BNA, compared with those of two adversarial training defense methods, against all-to-one attacks on CIFAR-10 datasets.

Trigger type		BadNet	CB	l_0 inv	l_2 inv	SP	WaNet
GeodesicAdv	ACC	0.8363	0.8477	0.8519	0.8474	0.8455	0.8364
	ASR	0.9196	0.0082	0.9996	0.0000	0.0107	0.0000
	SIA	0.0646	0.8392	0.0000	0.8407	0.8376	0.819
WaveletAdv	ACC	0.8137	0.8109	0.8172	0.8189	0.821	0.8163
	ASR	0.9228	0.0093	1.0000	0.0125	0.0109	0.0137
	SIA	0.0665	0.8041	0.0000	0.8093	0.8123	0.7983
BNA (ours)	ACC	0.9032	0.9072	0.9068	0.9005	0.9058	0.8945
	ASR	0.0139	0.0127	0.0033	0.0042	0.0104	0.0041
	SIA	0.8835	0.8787	0.8924	0.8383	0.8863	0.8530

Table 4
ACC, ASR, and SIA for BNA, compared with those of NC, I-BAU, ANP, NAD, ARGD, and MCR, with limited amount of clean data on VGGFace2 and CIFAR-10. Both datasets are poisoned by the BadNet attack.

	VGGFace2				CIFAR-10							
	Vanilla	NC	I-BAU	BNA	Vanilla	NC	I-BAU	ANP	NAD	ARGD	MCR	BNA
ACC	0.8989	**0.8967**	0.6828	0.8917	0.9122	0.8848	0.8539	0.6463	0.8731	0.4946	0.8650	**0.9026**
ASR	0.9771	0.9693	0.9737	**0.0046**	0.9573	0.2531	0.4175	**0.0117**	0.8880	0.0227	0.8078	0.0185
SIA	0.0216	0.0294	0.0196	**0.8889**	0.0397	0.6842	0.5148	0.6324	0.1007	0.4731	0.1684	**0.8807**

Table 5
ACC, ASR, and SIA for BNA as a function of (1) the number of poisoned instances injected into the training set; (2) the perturbation size under all-to-one CB attack.

	Number of poisoned instances per class					Perturbation size (*255)				
	50	100	150	200	250	2	3	4	5	6
ACC	0.9112	0.9094	0.9098	0.9102	0.9015	0.9094	0.9041	0.9079	0.8992	0.8912
ASR	0.0095	0.0141	0.0121	0.0170	0.0090	0.0141	0.0395	0.0222	0.0109	0.0388
SIA	0.8851	0.8837	0.8728	0.8840	0.8662	0.8851	0.8783	0.8711	0.8814	0.8435

DNN parameters using very limited data. (Note that BNA mitigation uses much less clean labeled data than what was reported for these other methods in their original papers.) Though MCR can effectively deactivate most of the backdoor attacks, excluding the global pattern CB and l_2 inv, it fails to infer the true source classes for the backdoor-triggered instances. For ANP with neuron pruning, the performance is acceptable only for A2O with the BadNet trigger. One possible reason is that invisible, perturbation-based triggers affect most neurons only moderately (which is also discussed in [65]); thus, pruning a small number of neurons cannot mitigate the attack. In contrast, our method successfully mitigates all these backdoor attacks (with generally the best ACC and ASR compared with the others) regardless of the trigger type and attack setting. Notably, *since the purpose of BNA's divergence minimization is to maximize the SIA*, it unsurprisingly achieves the best SIA with a clear margin over all other methods, in all cases (the corresponding distribution divergences are shown in Tab.D.15 in Apdx.D). We also tune the poisoning ratio and perturbation size used in A2O CB attacks, and the performance for BNA slightly declines as the attack is strengthened, as shown in Table 5. However, it still outperforms the other methods (see Tab.F.17 in Apdx.F).

For the CL attack, we poison half (2500) of the target-class training samples to achieve an effective attack (which is stronger than in the original paper [62]), as shown in Table 2. Although NAD and ARGD effectively deactivate the attack, both ACC and ASR drop significantly. For NC, ANP, and MCR, the ACC after mitigation is almost the same as the ACC before mitigation, but the ASR is still high. For ANP, nearly half of the backdoor-trigger images are unimpeded by the mitigation system. The attack is still effective after MCR is deployed. I-BAU does not perform well in mitigating the CL attack — the mitigated model fails to correctly classify most of the clean test images, but still recognizes half of the backdoor-trigger images to the target class. By contrast, BNA decreases the ACC by only a small amount, reduces ASR to around 6%, and correctly classifies 80% of the backdoor-trigger images.

Apart from post-training backdoor mitigation methods, we also assess the performance of our proposed method against adversarial training approaches, which serve as during-training backdoor defenses [63,64]. According to Table 3, adversarial training methods perform well against imperceptible backdoor patterns (*i.e.*, CB, l_2 inv, SP, and WaNet) – the ASR is low and the SIA is close to the ACC, indicating that the backdoor was not planted during training. However, these methods struggle with perceptible backdoor patterns (*i.e.*, BadNet and l_0 inv), where the ASR exceeds 90%. In contrast, without access to the original training data, our method demonstrates strong performance across various trigger types.

Results of BNA on other datasets are shown in Tables 4 and 6. We first train a DNN on the VGGFace2 dataset poisoned by the BadNet attack. As shown in Table 4, the BadNet attack is effective, with a high ASR and a nearly unchanged ACC. (The ACC for the DNN trained on the clean VGGFace2 dataset is 0.9211.) We then apply BNA, NC, and I-BAU on the poisoned DNN.[9] For all mitigation methods, we only preserve 10

[9] We did not evaluate the performance of ANP, NAD, and ARGD on VGGFace2, since these references do not provide the architecture of VGG-16 that fits their respective mitigation system. Although MCR provides the

Table 6
ACC, ASR, and SIA for BNA against all-to-one attacks on CIFAR-100, GTSRB, ImageNette, and TinyImageNet datasets.

Trigger type		GTSRB					CIFAR-100				TinyImageNet	ImageNette
		BadNet	CB	l_0 inv	l_2 inv	WaNet	BadNet	CB	l_0 inv	l_2 inv	BadNet	BadNet
Vanilla	ACC	0.9517	0.9556	0.9531	0.9521	0.9408	0.6796	0.6917	0.6863	0.6804	0.5192	0.8626
	ASR	1.0000	1.0000	1.0000	0.9794	0.9000	0.9037	0.9169	0.9935	0.9097	0.8058	0.9144
	SIA	0.0000	0.0000	0.0000	0.0169	0.0905	0.0781	0.0646	0.0063	0.0707	0.1134	0.0771
BNA	ACC	0.9491	0.9548	0.9505	0.9500	0.9404	0.6770	0.6863	0.6858	0.6787	0.5178	0.7941
	ASR	0.0000	0.0000	0.0122	0.0001	0.0041	0.0002	0.0524	0.0062	0.0016	0.0043	0.0016
	SIA	0.9312	0.9454	0.9330	0.8945	0.9338	0.6526	0.5880	0.6169	0.5224	0.4965	0.7940

Table 7
ACC, ASR, and SIA for BNA, compared with those of NC, NAD, I-BAU, ANP, and ARGD, against the ResNet-18 trained on the CIFAR-10 dataset poisoned by the all-to-one Blend and SIG backdoor attacks.

		Vanilla	NC	I-BAU	ANP	NAD	ARGD	BNA
Blend	ACC	0.9264	0.8132	0.7932	0.8667	0.8221	0.4856	**0.8942**
	ASR	0.9731	**0.0488**	0.7419	0.5119	0.0521	0.0782	0.1283
	SIA	0.0254	0.5593	0.1427	0.3369	0.6003	0.4181	**0.6252**
SIG	ACC	0.9266	0.8234	0.5696	0.8251	0.7794	0.4233	**0.8716**
	ASR	0.9991	0.1414	0.2594	0.9451	0.3223	0.0988	**0.0158**
	SIA	0.0008	0.2980	0.1319	0.0383	0.2503	0.3271	**0.3357**

Table 8
ACC, ASR, and SIA for BNA as a function of scale factor and bin size on ResNet-18 trained on CIFAR-10 poisoned by all-to-one BadNet attack.

τ (Δb=0.1)	10	100	200	300	400	500	600	700	800	900	1000
ACC	0.9024	0.9025	0.9022	0.9019	0.9024	0.9019	0.9022	0.9018	0.9017	0.9015	0.9021
ASR	0.0257	0.022	0.0206	0.0207	0.0214	0.0202	0.0212	0.0209	0.0207	0.0201	0.0204
SIA	0.8744	0.8758	0.8774	0.8768	0.8768	0.8775	0.8773	0.8768	0.8772	0.8778	0.8777
Δb (τ=150)	0.1	0.11	0.12	0.13	0.14	0.15	0.16	0.17	0.18	0.19	0.2
ACC	0.9028	0.9018	0.9019	0.9023	0.9019	0.9023	0.9019	0.9019	0.9022	0.9022	0.9022
ASR	0.0197	0.0202	0.0199	0.0204	0.0199	0.0204	0.0198	0.0206	0.0206	0.0207	0.0212
SIA	0.8799	0.8775	0.8779	0.8773	0.8772	0.8779	0.8773	0.8767	0.8779	0.8769	0.8764

clean images per class since there is a severely limited number of samples for VGGFace2. BNA effectively reduces the ASR and yields a high SIA, outperforming the other mitigation methods. We will thoroughly discuss the impact of the number of clean images possessed by the defender in Section 5.6. The ACC for DNNs trained without attack for GTSRB, CIFAR-100, ImageNette, and TinyImageNet are 0.9567, 0.6926, 0.8726, and 0.5224, respectively; while ACC, ASR, and SIA for attacked DNNs are shown in the row "Vanilla" in Table 6, which demonstrate that all the attacks are effective. We apply BNA on the poisoned DNNs, with the same settings as for CIFAR-10, which significantly reduces ASR (to less than 1.3% in all cases), with uniformly high SIA and ACC.

We also evaluated the performance of our BNA against all-to-one backdoor attacks that utilize more complex global backdoor patterns, such as the blended backdoor attack [2] and Sinusoidal Signal backdoor attack (SIG) [61]. Details of the attack configurations can be found in Apdx.B.2. The performance of our BNA as well as other mitigation methods are shown in Table 7. The results demonstrate the effectiveness of our BNA mitigation method even when dealing with complicated backdoor patterns. Compared with the other methods,[10] our BNA significantly reduces the ASRs and produces relatively satisfactory SIAs, while maintaining ACCs that are competitive with pre-mitigation figures.

5.3. The "detection-before-mitigation" scenario

As discussed in Section 4.1, BNA performs backdoor mitigation only after the model has been detected as backdoor-poisoned. To justify the "detection-before-mitigation scenario", we first apply the mitigation methods that do not involve a detection system (*i.e.*, I-BAU, ANP, NAD, ARGD, and MCR) on a ResNet-18 trained on the attack-free CIFAR-10 dataset. The resulting (absolute) drop in ACC is shown in column "clean" in Table 10. ANP has the largest impact on ACC — the ACC drops by 0.1877 after mitigation. I-BAU and ARGD respectively decrease the ACC by 0.0684 and 0.0343. NAD and MCR keep the ACC almost as high as that of the vanilla model, but they are not sufficiently effective in terms of SIA when the model is poisoned. NC and BNA detect the backdoor attack before mitigation; thus there is no impact on the ACC for clean classifiers (for which no attack is detected). We also found reduction in ACCs when applying all mitigation methods for a ResNet-18 model trained on CIFAR-10 poisoned by an all-to-one BadNet attack in column "BadNet". All the other methods decrease ACC by more than 0.03, while our method has little impact on ACC.

5.4. Test-time backdoor-trigger instance detection

Different from other tuning-based backdoor mitigation approaches, our BNA can also detect backdoor-trigger instances at test-time, as described in Section 4.2 and shown in Fig. 3. Here, we evaluate accuracy of our test-time detector compared with a state-of-the-art detector named STRIP [22]. For any input image during inference, STRIP blends it with clean images possessed by the defender. The blended image is fed into the poisoned DNN, with an entropy calculated on the output posteriors. If the entropy is lower than a prescribed detection threshold, the input is deemed to be embedded with the trigger. Here, we set the detection threshold to achieve 15% FPR for STRIP, a choice which achieves a generally good trade-off between TPR and FPR. In contrast, BNA does not require setting a detection threshold. In Table 9, we show the True Positive Rate (TPR, *i.e.*, the fraction of backdoor-trigger images correctly detected) and the False Positive Rate (FPR, *i.e.*, the fraction of clean test images from the

architecture of VGG-16, it is different from the one provided by PyTorch. Therefore we cannot load the pre-trained weights and make a fair comparison.

[10] We were not able to reproduce the results reported in the published papers describing these methods due to different defense settings — in our experiments, the defender possesses far fewer clean samples.

Table 9
TPR and FPR for BNA, compared with STRIP, against all attacks created on CIFAR-10.

Trigger type		BadNet		CB		l_0 inv		l_2 inv		SP		WaNet	
		A2O	A2A	A2O	A2A	A2O	A2A	A2O	A2A	A2O	A2A	A2O	A2A
BNA	FPR	**0.1390**	**0.0606**	**0.1144**	**0.1092**	**0.1413**	**0.0600**	0.1976	**0.1027**	**0.1323**	**0.0656**	**0.1406**	**0.0865**
	TPR	**0.9872**	**0.9508**	**0.9872**	**0.9682**	0.9967	**0.9873**	**0.9958**	**0.9793**	**0.9894**	**0.9294**	**0.9959**	**0.9248**
STRIP	FPR	0.15	0.15	0.15	0.15	0.15	0.15	**0.15**	0.15	0.15	0.15	0.15	0.15
	TPR	0.9638	0.5147	0.6802	0.2089	**0.9995**	0.1053	0.9924	0.5272	0.8522	0.3123	0.0202	0.0411

Table 10
Drop in ACC when applying I-BAU, ANP, NAD, ARGD, NC and BNA on ResNet-18 trained on the clean (attack-free) CIFAR-10 dataset and trained on CIFAR-10 poisoned by the all-to-one BadNet attack.

I-BAU		ANP		NAD		ARGD		MCR		NC		BNA	
Clean	BadNet	Clean	BadNet	Clean	BadNet	Clean	BadNet	Clean	BadNet	Clean	BadNet	Clean	BadNet
0.0684	0.0622	0.1877	0.0478	0.0064	0.0308	0.0343	0.0433	0.0038	0.0371	NA	0.0325	NA	0.0090

backdoor target class(es) that are falsely detected) for both methods. Although STRIP performs well on A2O attacks for some trigger types, *e.g.*, BadNet, l_0 inv, and l_2 inv, its TPR drops drastically on attacks using human-imperceptible triggers, especially the WaNet attacks. Moreover, it does not perform well on all A2A attacks, with a largest TPR of only 0.5272. By contrast, BNA is effective for all these attacks — it detects almost all the backdoor-trigger images, with FPRs comparable to STRIP.

5.5. Mitigation performance against adaptive attacks

A recent backdoor attack proposed by [66] minimizes a metric similar to that used by our BNA defense, in order to achieve better stealthiness. This attack can be viewed as an adaptive attack against our mitigation defense since the trained classifier will be more sensitive to even a smaller distribution divergence than for ordinary backdoor attacks. Nevertheless, our method successfully mitigates this attack. In our experiment on CIFAR-10, the average distribution total variation divergence over all neurons is reduced from 8067 to 2789. Accordingly, the ACC/ASR before and after mitigation are 0.9162/0.9978 and 0.8906/0.0072, respectively, with an SIA of 0.8496.

5.6. Mitigation with a limited amount of clean data

Why tuning-based methods like NC cannot achieve SIAs as high as BNA (which does not alter the DNN's parameters) ? Note that NC tunes the classifier using instances embedded with the estimated trigger but without label flipping. This is equivalent to minimizing the divergence between internal activation distributions for clean and triggered instances (see Tab.D.15 in Apdx.D), but by altering the DNN's parameters. Even for an optimal (zero) divergence, the best achievable SIA of NC is still upper-bounded by the ACC of the classifier after tuning, which usually *drops* due to the **data insufficiency**. By contrast, the reference distribution for BNA's divergence minimization is obtained by feeding clean instances to the poisoned classifier *without* changing its parameters; thus, it is a "better" reference with a higher upper-bound ACC.

For the main experiments on CIFAR-10, we preserve 100 clean test images for all mitigation methods. In other words, all mitigation methods, excluding our BNA, tune the (around 11 million) trainable parameters of the poisoned ResNet-18 based only on 1000 clean labeled images (2% of the training set) for a few epochs. Our method is light-weight, since it only updates the (less than 10 thousand) non-trainable parameters (*i.e.*, mean and standard deviation). Note that all the mitigation methods use more clean images in their original papers than in the experiments reported herein. For example, NC and NAD respectively chose 10% and 5% of the clean training samples for mitigation. For all mitigation methods, excluding our BNA, the **insufficiency** of clean labeled data reduces the ACC by at most 10%. The SIA is upper-bounded by the ACC after mitigation and is also affected by data insufficiency. This is also verified in [42], where they varied the number of clean images from 0% to 20% of the clean training samples, with the performance of NAD significantly degraded as the number of clean samples decreases. Our BNA only aligns internal layer distributions without affecting the trainable parameters, and thus is more robust when the amount of clean samples is limited.

Table 11
Time (in seconds) used for deploying the proposed mitigation methods on ResNet-18 trained CIFAR-10 dataset.

Defense	BadNet	CB	l_0 inv	l_2 inv	SP	WaNet
BNA (ours)	156.81	161.63	162.45	160.43	160.09	149.83

To further demonstrate the impact of the number of clean samples on mitigation performance, for the ResNet-18 trained on CIFAR-10 poisoned by the BadNet attack, we reduce the number of clean images used by the defender to just 10 images per class. The corresponding performance of all mitigation methods is shown in Table 4. Although ANP and ARGD effectively de-activate the backdoor attack, both ACC and SIA dramatically decrease. For NC, I-BAU, NAD, and MCR, the ACC changes a little, but the attack is still effective, especially for NAD and MCR. However, our BNA is still effective, with the ASR less than 2% and both ACC and SIA comparable to the ACC before mitigation.

Data insufficiency is a common phenomenon in real-world applications. For example, we use a subset of VGGFace2, which consists of 18 identities, each of which has 450 training face images and 100 test face images. So, VGGFace2 is much smaller than other benchmark datasets such as CIFAR-10. We only assign 10 clean images per class for the defender. The results are shown in Table 4. On this high-resolution and insufficient dataset, both NC and I-BAU fail to de-activate the BadNet attack. In contrast, our BNA successfully reduces the ASR to 0.46% and has ACC and SIA about 89%.

5.7. Time complexity

We report execution times of our mitigation method under all attacks in the "all-to-one" setting in Table 11. Our experiments utilize an NVIDIA RTX 3090 GPU with 24 GB of memory. The execution time of our method does not exceed 165 s for any attack. For comparison, training a Resnet-18 on CIFAR-10 for 30 epochs takes approximately 20 min.

6. Conclusion

In this paper, we revealed an activation distribution alteration property for backdoor attacks. We theoretically proved that by correcting such alteration, backdoor trigger instances will be correctly classified to their original source classes. Accordingly, we proposed a post-training backdoor mitigation approach to align distributions of

clean and backdoor-trigger samples through simple transformations, *without changing millions trainable parameters* of the classifier, which outperformed methods that use DNN fine-tuning. The proposed method is *robust* especially when there is *limited amount* of clean data available to the defender, compared with parameter-tuning based methods. Besides, the proposed method is *flexible* to be integrated with existing detection systems. Moreover, our method can detect instances with the trigger during inference.

CRediT authorship contribution statement

Xi Li: Conceptualization, Methodology, Project administration, Visualization, Writing – original draft, Writing – review & editing . **Zhen Xiang:** Project administration, Visualization, Writing – original draft, Writing – review & editing . **David J. Miller:** Funding acquisition, Supervision. **George Kesidis:** Funding acquisition, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xi Li, Zhen Xiang, David Miller, George Kesidis reports financial support was provided by National Science Foundation.

Acknowledgments

This research was supported in part by NSF SBIR, USA grant 2132294.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neucom.2024.128752.

Data availability

Data will be made available on request.

References

[1] T. Gu, K. Liu, B. Dolan-Gavitt, S. Garg, BadNets: Evaluating Backdooring Attacks on Deep Neural Networks, IEEE Access (2019).
[2] X. Chen, C. Liu, B. Li, K. Lu, D. Song, Targeted backdoor attacks on deep learning systems using data poisoning, 2017, arXiv:1712.05526.
[3] T.A. Nguyen, A.T. Tran, WaNet - Imperceptible warping-based backdoor attack, in: ICLR, 2021.
[4] S. Li, B.Z.H. Zhao, J. Yu, M. Xue, D. Kaafar, H. Zhu, Invisible backdoor attacks against deep neural networks, 2019, arXiv:1909.02742.
[5] A. Saha, A. Subramanya, H. Pirsiavash, Hidden trigger backdoor attacks, in: AAAI, 2020.
[6] S. Li, M. Xue, B. Zhao, H. Zhu, X. Zhang, Invisible Backdoor Attacks on Deep Neural Networks Via Steganography and Regularization, IEEE Trans. Dependable Secure Comput. 18 (05) (2021) 2088–2105.
[7] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks, in: NDSS, 2018.
[8] J. Dai, C. Chen, Y. Li, A backdoor attack against LSTM-based text classification systems, IEEE Access (2019).
[9] Z. Xiang, D.J. Miller, S. Chen, X. Li, G. Kesidis, A backdoor attack against 3D point cloud classifiers, in: ICCV, 2021.
[10] X. Li, G. Kesidis, D.J. Miller, V. Lucic, Backdoor attack and defense for deep regression, 2021, arXiv:2109.02381.
[11] Z. Zhao, X. Chen, Y. Xuan, Y. Dong, D. Wang, K. Liang, DEFEAT: Deep hidden feature backdoor attacks by imperceptible perturbation and latent representation constraints, in: CVPR, 2022.
[12] Z. Wang, J. Zhai, S. Ma, BppAttack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning, in: CVPR, 2022.
[13] J. Bai, K. Gao, D. Gong, S.-T. Xia, Z. Li, W. Liu, Hardly perceptible trojan attack against neural networks with bit flips, in: ECCV, 2022.
[14] X. Qi, T. Xie, R. Pan, J. Zhu, Y. Yang, K. Bu, Towards practical deployment-stage backdoor attack on deep neural networks, in: CVPR, 2022.
[15] C. Xie, K. Huang, P.-Y. Chen, B. Li, DBA: Distributed backdoor attacks against federated learning, in: ICLR, 2020.
[16] Y. Yao, H. Li, H. Zheng, B.Y. Zhao, Latent backdoor attacks on deep neural networks, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19, 2019.
[17] L. Wang, Z. Javed, X. Wu, W. Guo, X. Xing, D. Song, BACKDOORL: Backdoor attack against competitive reinforcement learning, in: IJCAI, 2021.
[18] M. Du, R. Jia, D. Song, Robust anomaly detection and backdoor attack detection via differential privacy, in: ICLR, 2020.
[19] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, X. Zhang, ABS: Scanning neural networks for back-doors by artificial brain stimulation, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19, 2019, pp. 1265–1282.
[20] Y. Dong, X. Yang, Z. Deng, T. Pang, Z. Xiao, H. Su, J. Zhu, Black-box detection of backdoor attacks with limited information and data, in: ICCV, 2021.
[21] E. Chou, F. Tramèr, G. Pellegrino, SentiNet: Detecting localized universal attacks against deep learning systems, in: 2020 IEEE Security and Privacy Workshops, 2020.
[22] Y. Gao, C. Xu, D. Wang, S. Chen, D.C. Ranasinghe, S. Nepal, STRIP: a defence against trojan attacks on deep neural networks, in: ACSAC, 2019.
[23] B. Tran, J. Li, A. Madry, Spectral signatures in backdoor attacks, in: NeurIPS, 2018.
[24] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I.M. Molloy, B. Srivastava, Detecting backdoor attacks on deep neural networks by activation clustering, in: AAAI, 2019.
[25] Z. Xiang, D. Miller, G. Kesidis, A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense, in: MLSP, Pittsburgh, 2019.
[26] Z. Zhang, L. Lyu, W. Wang, L. Sun, X. Sun, How to inject backdoors with better consistency: Logit anchoring on clean data, in: ICLR, 2022.
[27] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, B.Y. Zhao, Neural cleanse: Identifying and mitigating backdoor attacks in neural networks, in: 2019 IEEE Symposium on Security and Privacy, 2019.
[28] Z. Xiang, D.J. Miller, G. Kesidis, Detection of backdoors in trained classifiers without access to the training set, IEEE Trans. Neural Netw. Learn. Syst. (2020).
[29] Y. Dong, X. Yang, Z. Deng, T. Pang, Z. Xiao, H. Su, J. Zhu, Black-box detection of backdoor attacks with limited information and data, in: ICCV, 2021.
[30] W. Guo, L. Wang, X. Xing, M. Du, D. Song, TABOR: A highly accurate approach to inspecting and restoring trojan backdoors in AI systems, 2019, arXiv:1908.01763.
[31] Z. Xiang, D.J. Miller, G. Kesidis, Revealing backdoors, post-training, in DNN classifiers via novel inference on optimized perturbations inducing group misclassification, in: ICASSP, 2020.
[32] Y. Shen, S. Sanghavi, Learning with bad training data via iterative trimmed loss minimization, in: ICML, 2019, pp. 5739–5748.
[33] K. Huang, Y. Li, B. Wu, Z. Qin, K. Ren, Backdoor defense via decoupling the training process, in: ICLR, 2022.
[34] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, X. Ma, Anti-backdoor learning: Training clean models on poisoned data, in: NeurIPS, 2021.
[35] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, M. Wang, Practical detection of trojan neural networks: Data-limited and data-free cases, in: ECCV, 2020.
[36] X. Xu, Q. Wang, H. Li, N. Borisov, C.A. Gunter, B. Li, Detecting AI trojans using meta neural analysis, in: Proc. IEEE Symposium on Security and Privacy, 2021.
[37] S. Kolouri, A. Saha, H. Pirsiavash, H. Hoffmann, Universal litmus patterns: Revealing backdoor attacks in CNNs, in: CVPR, 2020, pp. 298–307.
[38] K. Liu, B. Dolan-Gavitt, S. Garg, Fine-pruning: Defending against backdooring attacks on deep neural networks, in: RAID, 2018.
[39] D. Wu, Y. Wang, Adversarial neuron pruning purifies backdoored deep models, in: NeurIPS, 2021.
[40] J. Guan, Z. Tu, R. He, D. Tao, Few-shot backdoor defense using Shapley estimation, 2022.
[41] R. Zheng, R. Tang, J. Li, L. Liu, Data-free backdoor removal based on channel Lipschitzness, in: Proc. ECCV, 2022.
[42] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, X. Ma, Neural attention distillation: Erasing backdoor triggers from deep neural networks, in: ICLR, 2021.
[43] J. Xia, T. Wang, J. Ding, X. Wei, M. Chen, Eliminating backdoor triggers for deep neural networks using attention relation graph distillation, in: IJCAI, 2022.
[44] Y. Zeng, S. Chen, W. Park, Z. Mao, M. Jin, R. Jia, Adversarial unlearning of backdoors via implicit hypergradient, in: ICLR, 2022.
[45] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: ICLR, 2018.
[46] B.G. Doan, E. Abbasnejad, D.C. Ranasinghe, Februus: Input purification defense against trojan attacks on deep neural network systems, in: Annual Computer Security Applications Conference, 2020, pp. 897–912.
[47] P. Zhao, P. Chen, P. Das, K.N. Ramamurthy, X. Lin, Bridging mode connectivity in loss landscapes and adversarial robustness, in: 8th International Conference on Learning Representations, ICLR, 2020.
[48] A. Krizhevsky, Hinton, Learning multiple layers of features from tiny images, 2009, http://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf.
[49] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016.

[50] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, A. Madry, Adversarial examples are not bugs, they are features, in: NeurIPS, 2019.
[51] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: F.R. Bach, D.M. Blei (Eds.), ICML, 2015.
[52] Y. Li, Y. Jiang, Z. Li, S.-T. Xia, Backdoor learning: A survey, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–18.
[53] X. Li, Z. Xiang, D.J. Miller, G. Kesidis, Test-time detection of backdoor triggers for poisoned deep neural networks, in: ICASSP, 2022.
[54] S.M. Ali, S.D. Silvey, A general class of coefficients of divergence of one distribution from another, J. R. Stat. Soc. Ser. B-Methodol. 28 (1966) 131–142.
[55] H. Chen, C. Fu, J. Zhao, F. Koushanfar, DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks, in: IJCAI, 2019, pp. 4658–4664.
[56] F.R. Hampel, The influence curve and its role in robust estimation, J. Amer. Statist. Assoc. 69 (1974).
[57] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, C. Igel, Detection of traffic signs in real-world images: The german traffic sign detection benchmark, in: International Joint Conference on Neural Networks, 2013.
[58] J. Howard, ImageNette, 2020, URL https://github.com/fastai/imagenette/.
[59] Y. Le, X.S. Yang, Tiny ImageNet visual recognition challenge, 2015, URL https://tiny-imagenet.herokuapp.com.
[60] Q. Cao, L. Shen, W. Xie, O.M. Parkhi, A. Zisserman, VGGFace2: A dataset for recognising faces across pose and age, in: 13th IEEE International Conference on Automatic Face & Gesture Recognition, FG 2018, Xi'an, China, May (2018) 15-19, IEEE Computer Society, 2018, pp. 67–74.
[61] M. Barni, K. Kallas, B. Tondi, A new backdoor attack in CNNS by training set corruption without label poisoning, in: 2019 IEEE International Conference on Image Processing, ICIP, IEEE, 2019, pp. 101–105.
[62] A. Turner, D. Tsipras, A. Madry, Label-consistent backdoor attacks, 2019, arXiv abs/1912.02771.
[63] J. Yan, H. Yin, Z. Zhao, W. Ge, J. Zhang, Enhance adversarial robustness via geodesic distance, IEEE Trans. Artif. Intell. (2024).
[64] J. Yan, H. Yin, Z. Zhao, W. Ge, H. Zhang, G. Rigoll, Wavelet regularization benefits adversarial training, Inform. Sci. (2023).
[65] H. Wang, Z. Xiang, D.J. Miller, G. Kesidis, Universal post-training backdoor detection, 2022, arXiv 2205.06900.
[66] K. Doan, Y. Lao, P. Li, Backdoor attack with imperceptible input and latent modification, in: NeurIPS, 2021.

Xi Li is an incoming Assistant Professor in the Department of Computer Science at the University of Alabama at Birmingham. She completed her Ph.D. in Computer Science at Penn State University under the guidance of Dr. George Kesidis and Dr. David J. Miller. She earned her B.S. degree in Electrical Engineering from Southeast University in Nanjing, China, in 2016, and her M.S. degree in Computer Science from Penn State University in 2018. Her research has focused on trustworthy AI and adversarial machine learning, with a specific emphasis on poisoning attacks and defenses against deep neural networks. Her work has contributed to several publications in prestigious conferences such as ICCV, AAAI, and ICASSP, and in journals like IEEE Transactions on Knowledge and Data Engineering (TKDE).

Zhen Xiang (Graduate Student Member, IEEE) is an incoming Assistant Professor at the University of Georgia. He is currently a postdoctoral researcher at the Secure Learning Lab at the University of Illinois Urbana-Champaign (UIUC). Zhen completed his Ph.D. in Electrical Engineering at Pennsylvania State University, under the guidance of Professors David J. Miller and George Kesidis. Before that, he received his B.Sc. degree in Electronics and Computer Engineering from Hong Kong University of Science and Technology with an outstanding student award in 2014, and his M.Sc. degree in Electrical Engineering from University of Pennsylvania in 2016. His research interests include adversarial machine learning and statistical signal processing. His Ph.D. thesis focuses on defending backdoor attacks against deep neural network classifiers, which received the Dr. Nirmal K. Bose Dissertation Excellence Award in 2022. He served as reviewer for journals including IEEE TNNLS, Computers & Security, and IEEE SPM, and conferences including ICASSP, MLSP.

David J. Miller (Senior Member, IEEE) received the B.S.E. degree from Princeton University in 1987, the M.S.E. degree from the University of Pennsylvania in 1990, and the Ph.D. degree from the University of California at Santa Barbara in 1995, all in electrical engineering. He has been with the Department of Electrical Engineering at the Pennsylvania State University since 1995. His research interests include machine learning, source coding, and network security. He was a member of the Machine Learning for Signal Processing Technical Committee within the IEEE Signal Processing Society from 1997 to 2010 and from 2017–2022 and was its chair from 2007 to 2009.

George Kesidis (Senior Member, IEEE) received the B.A.Sc. degree in EE from the University of Waterloo in 1988, and the M.S. degree (neural networks and stochastic optimization) and the Ph.D. degree (networking and performance evaluation) in EECS from U.C. Berkeley in 1990 and 1992, respectively. Following eight years as a professor of ECE with the University of Waterloo, he has been a Professor of EE and CSE with the Pennsylvania State University since 2000. His research interests include problems in networking, cyber security, machine learning, performance evaluation, and cloud computing. Currently his research is supported by grants from NSF, ONR and Cisco. Dr. Kesidis has served as a Technical Program Committee (TPC) Co-Chair of IEEE INFOCOM and an Associated Editor of the Computer Networks Journal, ACM Transactions on Modeling and Computer Simulation (TOMACS), and IEEE Communications Surveys and Tutorials (ComST).