Universal Post-Training Backdoor Detection

Hang Wang [*] Penn State
Zhen Xiang [*] Penn State
David J. Miller Penn State
George Kesidis Penn State

Abstract

A Backdoor attack (BA) is an important type of adversarial attack against deep neural network classifiers, wherein test samples from one or more source classes will be (mis)classified to the attacker's target class when a backdoor pattern (BP) is embedded. In this paper, we focus on the *post-training* backdoor defense scenario commonly considered in the literature, where the defender aims to detect whether a trained classifier was backdoor attacked, without any access to the training set. To the best of our knowledge, existing post-training backdoor defenses are all designed for BAs with presumed BP types, where each BP type has a specific embedding function. They may fail when the actual BP type used by the attacker (unknown to the defender) is different from the BP type assumed by the defender. In contrast, we propose a *universal* post-training defense that detects BAs with arbitrary types of BPs, without making any assumptions about the BP type. Our detector leverages the influence of the BA, *independently of the BP type*, on the landscape of the classifier's outputs prior to the softmax layer. For each class, a *maximum margin* statistic is estimated using a set of random vectors; detection inference is then performed by applying an *unsupervised* anomaly detector to these statistics. Thus, our detector is also an advance relative to most existing post-training methods by *not* needing any legitimate clean samples, and can efficiently detect BAs with *arbitrary* numbers of source classes. These advantages of our detector over several state-of-the-art methods are demonstrated on four datasets, for three different types of BPs, and for a variety of attack configurations. Finally, we propose a novel, general approach for BA mitigation once a detection is made.

1 Introduction

Deep neural network (DNN) classifiers have achieved success in many research areas, but they are vulnerable to adversarial attacks [38, 28]. A Backdoor attack (BA) or Trojan is an important type of adversarial attack, under which a classifier will predict to the attacker's target class when a test sample from one or more source classes is embedded with the attacker's backdoor pattern (BP) [13, 26, 22]. A BA is typically launched by poisoning the classifier's training set with a few samples originally from the source classes, embedded with the same BP that will be used during inference, and labeled to the target class [4]. Since successful BAs do not degrade the classifier's accuracy on clean test samples (without the BP), they cannot be easily detected, e.g. using validation set accuracy [30].

Defenses against BAs can be deployed during the classifier's training stage [39, 2, 10, 17, 35], but such deployment is often infeasible (e.g., considering proprietary and legacy systems). In this paper, we consider the more practical and challenging post-training (PT) scenario, where the defender is the user of a trained classifier who aims to detect whether the classifier has been backdoor attacked. Here, the defender does not have access to the classifier's training set since it is either proprietary or has been long forgotten (legacy scenario). The defender also has no access to any unattacked classifiers for reference (e.g. for setting a detection threshold).

Existing PT defenses typically assume that the defender independently possesses a small set of clean, legitimate samples from every class. These samples may be used: i) to reverse-engineer putative BPs,

[*]The first two authors contributed equally to this paper.

Preprint. Under review.

which are the basis for anomaly detection [42, 14, 48, 25, 9, 43, 45, 34, 49]; or ii) to train shadow neural networks with and without (known) BAs – based on which a binary "meta-classifier" is trained to predict whether the classifier under inspection is backdoor attacked [18, 51, 40]. However, these methods assume the BP type (the mechanism for embedding a BP) used by the attacker is known. For reverse-engineering-based defenses (REDs), the embedding function of the BP is explicitly involved in the reverse-engineering problem [42]; thus, these methods may not be able to effectively detect BAs with other types of BPs. For the meta-classification approach, a pool of BP types is assumed for training shadow neural networks with BAs [51]; thus, these methods may not generalize well to BP types not seen in the training pool.

Our contributions. In this paper, we propose a *universal* PT backdoor detector which, unlike prior defenses, does not make any assumptions about the BP type used by the attacker. Our detection method is based on the atypicality of the landscape of the classifier's outputs (before the softmax) induced by BAs. In particular, we propose a novel *maximum margin* (MM) detection statistic, which is estimated for each class using a set of random vectors. A fully *unsupervised* anomaly detector is then applied to the MM statistics as detection inference. Thus, unlike other PT methods, *no clean samples* are required for detection. Moreover, this design allows our method to detect BAs with an *arbitrary* number of source classes, and with better *computational efficiency* than many existing methods. These advantages of our detector are demonstrated on four datasets, under three BP types, and for various DNN architectures. We also propose a novel BA mitigation approach, which does require a few clean samples. Here, we suppress the maximum possible neuron activations using a set of optimized upper bounds (one per neuron) without hurting the classifier's accuracy on clean samples. Unlike existing methods, we do not modify the DNN architecture nor any trained parameters.

2 Related Work

2.1 Backdoor Attacks

For a classification domain with sample space $\mathcal{X}$ and label space $\mathcal{Y}$, a backdoor attack (BA) aims to have the victim classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$: (a) learn to classify to the attacker's target class $t \in \mathcal{Y}$, when a test sample $\mathbf{x} \in \mathcal{X}$ from any of the source classes $\mathcal{S} \subset \mathcal{Y}$ is embedded with the backdoor pattern (BP); and (b) correctly classify clean test samples (without BP) [13]. BAs are typically launched by poisoning the classifier's training set, as discussed above.

BAs and BA defenses were initially proposed and studied for image classification, but more recently also for other domains and tasks [46, 53, 23]. While we focus on image domains, our method (unlike existing methods) is also likely broadly applicable since it does not rely on knowing the (often domain-dependent) BP type. For images, common BP types include: 1) an additive perturbation $\mathbf{v}$ embedded by $\tilde{\mathbf{x}} = [\mathbf{x}+\mathbf{v}]_{\rm c}$, where $||\mathbf{v}||_2$ is small (for imperceptibility) and $[\cdot]_{\rm c}$ is a domain-dependent clipping function [4, 54, 48]; 2) a local patch $\mathbf{u}$ embedded using an image-wide binary mask $\mathbf{m}$ by $\tilde{\mathbf{x}} = (1-\mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \mathbf{u}$, where $||\mathbf{m}||_1$ is small for imperceptibility and $\odot$ represents element-wise multiplication [13, 42, 50]; and 3) a local or global pattern $\mathbf{u}$ "blended" using an image-wide binary mask $\mathbf{m}$ and a blending factor $\alpha \in (0, 1)$ by $\tilde{\mathbf{x}} = (1 - \alpha \cdot \mathbf{m}) \odot \mathbf{x} + \alpha \cdot \mathbf{m} \odot \mathbf{u}$, where α is close to 0 for imperceptibility. Other more recent BP types include one that mimics physical reflection on smooth surfaces [27], a wrapping-based BP that introduces a spatial transformation to existing pixels [32], and a BP embedded in the frequency domain [44].

2.2 Backdoor Defenses

Defenses against BAs can be deployed during the classifier's training to detect BAs and sanitize a BA-poisoned training set [39, 2, 10, 17, 35]. However, these defenses are not applicable when the defender has no access to the training process. In this paper, we consider a popular *post-training* (PT) scenario where the defender aims to detect whether a trained classifier is backdoor attacked, but has *no access* to the classifier's training set or any clean classifiers for reference. In a similar scenario considered by the defenses of [12, 8, 6], the main goal is to detect if a test samples is embedded with a BP, thus is not directly in the scope of this paper.

Existing PT defenses include a family of "meta-learning" approaches, where a large number of shadow neural networks, labeled "attack" or "no attack", are trained – features extracted from these labeled networks are treated as supervised examples, input to a binary "meta-classifier" trained to discriminate "attack" from "no attack" [18, 51]. However, these methods employ a pool of BP types, and may fail when the actual type used by the attacker is not in the pool. Moreover, training the shadow networks requires a relatively large number of clean samples and is highly computational. REDs, another family of PT defenses, trial reverse-engineer the BP for each putative target class [42, 14, 25, 9, 43, 45]. Such reverse-engineering is performed using a small set of clean samples possessed by the defender [42], or using simulated samples obtained by model inversion [3, 11]. Detection inference is then based on statistics obtained from the estimated BPs (e.g. the ℓ_1 norm of the estimated mask for patch replacement BPs). However, reverse-engineering relies on knowledge of the BP type. Also, most REDs require some clean samples, which are not always available. In comparison, our universal detector does not rely on knowledge of the BP type and does not need any clean samples. Moreover, some REDs fail when the BA involves only a few source classes. [48] addressed this issue by estimating a putative BP for each class pair, but this greatly increases the required computation. While [34] and [49] estimate the BA source classes and the target class via a complicated optimization procedure, our method can accurately detect BAs with an arbitrary number of source classes and is computationally efficient, as will be shown in our experiments.

Once a BA is detected, a secondary goal is to mitigate the attack when there are no replacement classifiers available. In [24, 42, 52], a detected classifier is fine-tuned on a relatively large number of clean samples to "unlearn" the backdoor mapping. Although the main focus of this paper is BA detection, in Sec. 3.3, we also propose a method that mitigates BA with most types of BPs using very few clean samples, and without fine-tuning any of the classifier's original parameters. Our method can also be combined with existing ones to achieve even better performance.

3 Method

Again, we focus on the PT scenario here. Moreover, we do not make any assumptions about the BP type or the number of source classes involved in a possible BA. We first discuss the key ideas behind our detection method in Sec. 3.1. Then, in Sec. 3.2, we present our detection procedure, which consists of an estimation step to obtain a novel *maximum margin* (MM) statistic for each class, followed by unsupervised detection inference. We then propose a mitigation procedure in Sec. 3.3.

3.1 Key Ideas

Our detector is based on the influence of a BA on the landscape of the classifier's logit (i.e. the output before softmax) functions, independent of the BP type. Consider a BA with target class $t \in \mathcal{Y}$ and denote the classifier's logit function associated with any class $c \in \mathcal{Y}$ as $g_c : \mathcal{X} \rightarrow \mathbb{R}$ (assuming $\mathcal{X}$ compact without loss of generality). Regardless of the BP type, we will likely observe:

$$\max_{\mathbf{x}\in\mathcal{X}} \left[g_t(\mathbf{x}) - \max_{k\in\mathcal{Y}\setminus t} g_k(\mathbf{x})\right] \gg \max_{\mathbf{x}\in\mathcal{X}} \left[g_c(\mathbf{x}) - \max_{k'\in\mathcal{Y}\setminus c} g_{k'}(\mathbf{x})\right], \quad \forall c \in \mathcal{Y} \setminus t. \tag{1}$$

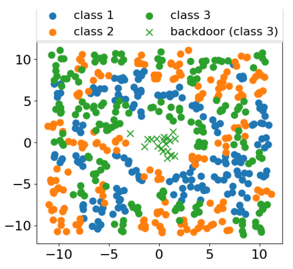
That is, the MM statistic for the true BA target class (defined as the left hand side of (1)) will tend to be *much larger* than the MM statistics for all other classes (right hand side of (1)).

Why a BA causes the above phenomenon? Note that a BP is a common pattern[2] embedded in all samples used for poisoning the classifier's training set; and the same BP will be embedded in test samples to induce test-time misclassifications. By contrast, non-BP class-discriminating features typically exhibit high variability[3] – e.g., birds of different species, or even the same object captured under different views, range, or lighting conditions. The commonality of a BP is critical for it to override class-discriminating features that will favor deciding to the source class and so that the BP can be easily learned (at a low poisoning rate [22]) by the victim classifier during training. However, the repetition of the common BP in the training set also induces an inevitable overfitting which: a) boosts the target class logit (by causing abnormally large activations from neurons positively correlated with this logit), and b) suppress the logits of all other classes (as will be shown empirically in Sec. 4.3.2). Consequently, an abnormally large margin between the target class logit and logits of all other classes will be created, due to both the "boosting" and the "suppression" effects.

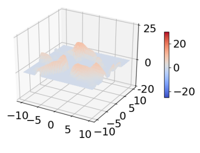
The above reasoning is visualized in Fig. 1 using a toy example. We consider two-dimensional inputs from three classes, with the sample distribution for each class specified by a unique Gaussian mixture

[2]Some recently proposed BAs use BPs that are sample-specific in the input space [31, 21]. However, these BPs embedded in different samples still share common semantic features and are similar to each other in some latent embedding space. Thus, these BAs are still detectable by our method, as will be shown in Apdx. N.

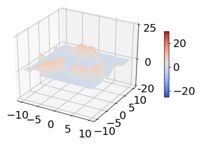
[3]In extreme cases, class-discriminating features can be highly common across samples from the same class, which creates an "intrinsic backdoor" hardly distinguishable from backdoors planted by an attacker [47]. A general solution to rule out such "intrinsic backdoors" is still an open problem (see discussion in Apdx. P).



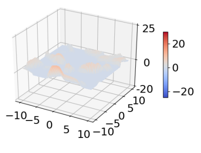
(a) Illustration of the distribution of samples from all three classes.



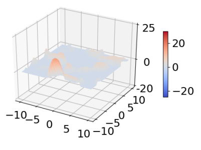
(b) BA10-class1



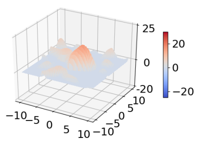
(c) BA100-class1



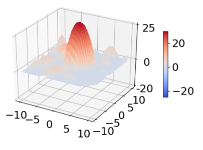
(d) BA10-class2



(e) BA100-class2



(f) BA10-class3



(g) BA100-class3

Figure 1: Illustration of the key ideas of our detection method. For two-dimensional input and three classes following the sample distribution in (a), consider two BAs (BA-10 and BA-100) with different poisoning rate but the same target class 3. For both BAs, the maximum margin for target class 3 ((f) and (g)) is abnormally large compared with that for the non-target classes ((b), (c), (d), (e)).

model (with a large number of components to mimic the high variability of class-discriminating features). We generate 500 training samples per class, and launch two BAs (both with target class 3), respectively with 10 (BA-10) and 100 (BA-100) "backdoor samples" inserted for poisoning. For each BA, we train a multilayer perceptron with three hidden layers, which achieves nearly 91% accuracy on clean test samples for both BAs. The attack success rates of the trained classifiers on test "backdoor samples" are 92% and 99% for BA-10 and BA-100, respectively. In Fig. 1, for each BA and each class, we plot the margin between the class logit and the largest logit among the other two classes as a function of the input space (only positive values are kept for better visualization). We clearly observe that the backdoor target class (Fig. 1f and 1g) has abnormally large MM for both BAs (which agrees with Eq. (1)). Moreover, compared with BA-10, the atypicality of MM for the BA target class is more obvious for BA-100 (note that the higher poisoning rate may in fact be preferred by the attacker since this yields a higher attack success rate).

3.2 Detection Procedure

Estimation step. For each putative target class $c \in \mathcal{Y}$, we estimate a MM statistic by solving:

$$\underset{\mathbf{x}\in\mathcal{X}}{\text{maximize}} \quad g_c(\mathbf{x}) - \max_{k\in\mathcal{Y}\setminus c} g_k(\mathbf{x}) \tag{2}$$

using gradient ascent with projection onto $\mathcal{X}$ (e.g. $\mathcal{X} = [0,1]^{H\times W\times C}$ for color images with height H, width W, and C channels) until convergence (which is guaranteed for continuous logit functions and compact $\mathcal{X}$). As is common practice, we perform multiple random initializations in $\mathcal{X}$ (e.g., for images, pixel values are uniformly randomly initialized in the interval $[0,1]$), and pick the best locally optimal solution. Compared with REDs that may suffer from mismatch between the assumed BP embedding type and the true attack BP type, our optimization problem does not require assuming a BP embedding type. Moreover, REDs' BP estimation using clean samples from *all* non-target classes has been found experimentally to fail when the majority of these classes are not source classes [49, 34]. By contrast, our method can detect BAs with an arbitrary number of source classes, since problem (2) above does not involve any legitimate samples from the domain.

Detection inference step. We propose an unsupervised anomaly detector inspired by [48]. Denote the estimated MM statistic for each class $c \in \mathcal{Y}$ as r_c and the largest statistic as $r_{\max} = \max_{c\in\mathcal{Y}} r_c$. We hypothesize that, when there is a BA, $r_{\max}$ will be associated with the BA target class and will be an outlier to the distribution of MM statistics for non-target classes. Thus, we estimate a null distribution H_0 using all statistics excluding $r_{\max}$. Given that MM is strictly positive (both theoretically and empirically for estimated MM in our experiments), we choose single-tailed density forms for the null distribution, e.g. a Gamma distribution in our experiments. To evaluate the atypicality of $r_{\max}$ under the estimated null, we compute an *order statistic* p-value [48]:

$$\text{pv} = 1 - H_0(r_{\max})^{K-1}, \tag{3}$$

where $K = |\mathcal{Y}|$ is the total number of classes in the domain. It can easily be shown that pv follows a uniform distribution on $[0,1]$ under the null hypothesis of "no attack". Thus, we claim a detection with confidence $1 - \theta$ (e.g. with the classical $\theta = 0.05$) if $\text{pv} < \theta$. If a BA is detected, the class associated with $r_{\max}$ is inferred as the BA target class.

3.3 Mitigation of Backdoor Attack

When a BA is detected, the victim classifier can be replaced by a new one from another trustworthy training authority; otherwise, prediction made by the victim classifier to classes other than the detected BA target class might still be trustworthy. An alternative choice is to *mitigate* the backdoor attack.

Our mitigation approach is based on the observation that a BA induces a subset of neurons in each layer to have an abnormally large activation (which is accumulated layer by layer, resulting in a large MM for the BA target class). Such "large activation" phenomenon is also the basis of the BA *detection* approach in [25], though several hyperparameters are required, e.g., to identify the neurons responsible for the large activation. In contrast, we apply to *every neuron* a specific optimized upper bound in order to suppress any possible large activation caused by a BA, without significant degradation in the classifier's accuracy on clean samples. Let $\sigma_l : \mathbb{R}^{n_{l-1}} \rightarrow \mathbb{R}^{n_l}$ be the activations of the l-th layer (for $l = 1, \cdots, L$) of the victim classifier (as a function of the activations of the previous layer). Here, we neglect the parameters in each layer for brevity, since none of them will be modified during our *mitigation* process. Then, the logit function for any class $c \in \mathcal{Y}$ and any input $\mathbf{x} \in \mathcal{X}(= \mathbb{R}^{n_0})$ can be represented by:

$$g_c(\mathbf{x}) = \mathbf{w}_c^T(\sigma_L \circ \cdots \circ \sigma_1(\mathbf{x})) + b_c, \tag{4}$$

where $\mathbf{w}_c \in \mathbb{R}^{n_L}$ and $b_c \in \mathbb{R}$ are the weight vector and bias associated with class c respectively. For each layer $l = 2, \cdots, L$, we also denote a bounding vector $\mathbf{z}_l \in \mathbb{R}^{n_l}$, such that the logit function, *with bounded activation*, for each class $c \in \mathcal{Y}$ and any input $\mathbf{x}$ can be represented by:

$$\bar{g}_c(\mathbf{x}; \mathbf{Z}) = \mathbf{w}_c^T(\bar{\sigma}_L(\bar{\sigma}_{L-1}(\cdots \bar{\sigma}_2(\sigma_1(\mathbf{x}); \mathbf{z}_2) \cdots ; \mathbf{z}_{L-1}); \mathbf{z}_L)) + b_c, \tag{5}$$

where $\mathbf{Z} = \{\mathbf{z}_2, \cdots, \mathbf{z}_L\}$ and $\bar{\sigma}_l(\cdot; \mathbf{z}_l) = \min\{\sigma_l(\cdot), \mathbf{z}_l\}$ for any $l = 2, \cdots, L$. To find the minimum activation upper bound for each neuron without affecting the classifier's performance on clean test samples, we propose to solve the following problem:

$$\begin{aligned} \min_{\mathbf{Z}=\{\mathbf{z}_2,\cdots,\mathbf{z}_L\}} \quad & \sum_{l=2}^{L} ||\mathbf{z}_l||_2 \\ \text{subject to} \quad & \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \mathbb{1}[y = \arg\max_{c\in\mathcal{Y}} \bar{g}_c(\mathbf{x}; \mathbf{Z})] \geq \pi, \end{aligned} \tag{6}$$

where $\mathbb{1}[\cdot]$ represents the indicator function, $\mathcal{D}$ is a small set of clean samples that are correctly classified (by the poisoned network) and used for BA mitigation, and π is the minimum accuracy benchmark (e.g., set $\pi = 0.95$). Here, we minimize the ℓ_2 norm of the bounding vectors to penalize activations with overly large absolute value in each layer. To practically solve the above problem, we propose to minimize the following Lagrangian:

$$L(\mathbf{Z}, \lambda; \mathcal{D}) = -\frac{1}{|\mathcal{D}| \times |\mathcal{Y}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \sum_{c\in\mathcal{Y}} [\bar{g}_c(\mathbf{x}; \mathbf{Z}) - g_c(\mathbf{x})]^2 + \lambda \sum_{l=2}^{L} ||\mathbf{z}_l||_2, \tag{7}$$

using gradient descent, where $\mathbf{Z}$ is initialized large. The first term of Eq. (7) aims to keep the classifier's logits for the samples in $\mathcal{D}$ unchanged. Such a design not only helps satisfy the accuracy constraint in problem (6), but also avoids having the logit associated with the class label for a given sample to further grow (i.e. overfitting), allowing mitigation to be achieved with limited samples[4]. λ is updated automatically (following Alg. 1 in Apdx. A) to fulfill the constraint of problem (6). Finally, the classifier with the BA mitigated is obtained by applying a softmax to the logits $\{\bar{g}_c(\mathbf{x}; \mathbf{Z}^*)\}_{c\in\mathcal{Y}}$ using the optimized $\mathbf{Z}^*$.

4 Experiments

Our experiments are conducted mainly on four benchmark datasets with different image resolution, image size and number of classes: CIFAR-10, CIFAR-100 [19], TinyImageNet, and GTSRB [37]. Details of these datasets are deferred to Apdx. B.

[4]Other choices, such as a differentiable surrogate for correct classification count, might lead to overfitting.

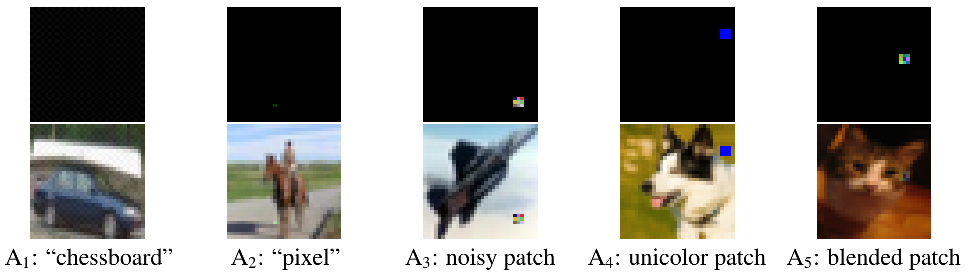


Figure 2: Example BPs used in our experiments (top) and images with these BPs embedded (bottom).

4.1 Main Experiments for Backdoor Attack Detection

In the following, we show the effectiveness of our *universal backdoor detector*, dubbed “UnivBD”, in terms of *detection accuracy* and *computational efficiency* compared with several state-of-the-art PT detectors for a variety of BA configurations (e.g. BP type, number of source classes).

4.1.1 Settings

We consider three common BP types in the BA literature: *additive*, *patch replacement*, and *blending* (with associated embedding functions in Sec. 2.1). Effectiveness of our UnivBD on warping-based BP and sample-specific BP are shown in Apdx. N. For the additive BP, we consider a *global* “chessboard” pattern from [48] and a *local* “pixel” perturbation from [39]. For the patch replacement BP, we consider a *noisy* patch from [41] and a “*unicolor*” patch from [42]. For the blended BP, we consider a blended *noisy* patch from [48]. For convenience, we denote the above five BPs as A_1-A_5 respectively. For A_2, the pixel being perturbed is randomly selected and fixed for each attack; for A_3-A_5, both the color and the location of the patch are randomly selected and fixed for each attack. Examples of these five BPs are shown in Fig. 2, with more details in Apdx. C.

Like most existing works (e.g. [42, 48, 14, 25]), we consider BAs with one target class (randomly selected for each attack). However, we allow BAs to have one or multiple source classes – for brevity, these two source class settings are denoted as ‘S’ (for “single”) and ‘M’ (for “multiple”) respectively. For each BA with the ‘S’ setting, the source class is randomly selected. For TinyImageNet, we randomly select ten source classes for each BA with the ‘M’ setting; while for the other three datasets, all classes other than the target class are selected as the source classes (which is assumed by the detectors in [42, 14, 25]) for the ‘M’ setting. With the above notations, a BA with “chessboard” BP and multiple source classes is denoted as ‘A_1-M’.

For CIFAR-10, we create ten ensembles of BAs for all ten combinations of A_1-A_5 and ‘S’/‘M’ respectively. For CIFAR-100 and GTSRB, we create five BA ensembles for A_1-A_5 respectively, all with the ‘M’ setting. We do not create BAs with a single source class for these two datasets because we cannot generate sufficient backdoor training images to launch a successful BA, given the limited number of images in each class for these two datasets. For TinyImageNet, we only generate one BA ensemble with the setting ‘A_3-M’ (which is arbitrarily selected) since the amount of time to just train a classifier on this dataset is extraordinarily high. For each ensemble, ten different BAs are generated independently according to the specified settings using the classical “data poisoning” protocol in [13]. Other configurations, including the number of backdoor training images created for BAs in each ensemble, are deferred to Apdx. D.

We train one classifier for each BA. The DNN architectures for BAs on CIFAR-10, CIFAR-100, TinyImageNet, and GTSRB are ResNet-18 [16], VGG-16 [36], ResNet-34 [16], and MobileNet [33], respectively. Training configurations are detailed in Apdx. E. For each dataset, we also create an ensemble of clean classifiers to evaluate the false detection rate. All the BAs we created are successful with high attack success rate (ASR) and negligible degradation in clean test accuracy (ACC)[5], compared with the clean classifiers trained for the same dataset (see Apdx. F).

4.1.2 Detection Performance

Detection configurations. We compare our UnivBD with six state-of-the-art PT detection methods: NC [42], TABOR [14], ABS [25], PT-RED[48], META [51], and TND[43]. We follow the original implementation of these methods with only modest changes (e.g. choosing the best detection threshold

[5]ASR and ACC are commonly used for assessing the effectiveness of BAs; see definitions in [48, 9, 43].

		CIFAR-10										
	N_{img}	clean	A_1-S	A_1-M	A_2-S	A_2-M	A_3-S	A_3-M	A_4-S	A_4-M	A_5-S	A_5-M
NC[42]	10	12/20	4/10	**10/10**	2/10	3/10	**8/10**	**10/10**	**9/10**	7/10	3/10	3/10
TABOR[14]	10	13/20	4/10	**8/10**	7/10	**8/10**	6/10	7/10	0/10	5/10	7/10	**9/10**
ABS[25]	1	**19/20**	2/10	7/10	4/10	6/10	**8/10**	**10/10**	7/10	5/10	7/10	**8/10**
META[51]	10k	15/20	**8/10**	6/10	0/10	0/10	**9/10**	**10/10**	4/10	2/10	**9/10**	7/10
TND[43]	5	11/20	2/10	2/10	3/10	**8/10**	3/10	3/10	1/10	0/10	5/10	6/10
PT-RED[48]	100	15/20	**10/10**	**10/10**	**9/10**	**10/10**	1/10	0/10	1/10	1/10	4/10	7/10
PT-RED+ABS	100	14/20	**10/10**	**10/10**	**9/10**	**10/10**	**8/10**	**10/10**	7/10	5/10	**8/10**	**10/10**
UnivBD(ours)	0	**18/20**	**9/10**	**8/10**	**8/10**	**10/10**	**10/10**	**10/10**	**8/10**	**10/10**	**9/10**	**10/10**

		CIFAR-100				TinyImageNet		GTSRB				
		clean	A_2-M	A_3-M	A_4-M	clean	A_3-M	clean	A_1-M	A_2-M	A_4-M	A_5-M
NC	1	4/10	3/10	**10/10**	**9/10**	3/10	**8/10**	13/20	**9/10**	5/10	2/10	6/10
TABOR	1	4/10	6/10	4/10	4/10	4/10	7/10	11/20	7/10	6/10	1/10	4/10
ABS	1	**10/10**	2/10	**9/10**	**9/10**	**9/10**	2/10	**17/20**	2/10	**9/10**	4/10	6/10
TND	1	2/10	2/10	2/10	5/10	5/10	3/10	12/20	3/10	4/10	0/10	1/10
UnivBD(ours)	0	**10/10**	**10/10**	**10/10**	**10/10**	**10/10**	**9/10**	**17/20**	7/10	**10/10**	**9/10**	**10/10**

Table 1: Detection accuracy of our UnivBD compared with other detectors (acc. $\geq$0.8 is in bold).

to maximize their performance). Especially for META, we use the same code provided by the authors to train the "meta-classifier" for detection. More details regarding these methods are deferred to Apdx. G. For our UnivBD, we solve problem (2) using gradient ascent with convergence criterion[6] $\epsilon = 10^{-5}$ and 30 random initializations. These choices are not critical to the detection accuracy. For the inference stage, the detection threshold is set to $\theta = 0.05$, i.e., with 0.95 detection confidence.

Detection accuracy. In Tab. 1, we show the detection accuracy of our UnivBD compared with the other methods on the BA ensembles we created and the number of legitimate images per class N_{img} used by each method, for each dataset. A successful detection requires both the BA to be detected and the target class to be correctly inferred. We also show the proportion of clean classifiers deemed to be not attacked by each detector. We only evaluate PT-RED and META for CIFAR-10 since the empirical computational cost applying these two methods to BAs on the other datasets are overly high (e.g. more than 24 hours for PT-RED on CIFAR-100). Also due to the time constraint (and the space limitation), we include detection accuracy results for a few arbitrarily selected BA ensembles for CIFAR-100, TinyImageNet, and GTSRB, respectively. Complete results for our UnivBD for *all* BA ensembles are shown in Apdx. H.

As we have discussed in Sec. 2, existing PT detectors presume one or several BP types. For example, NC shows strong capability in detecting BAs with patch replacement BPs (A_3&A_4), for which it is designed; but NC fails to detect BAs with local additive BP (A_2)[7]. Similar results are reported for ABS and META, which are designed for patch replacement/blending BPs (A_3-A_5), while not addressing additive BPs (A_1&A_2). In contrast, PT-RED performs well for additive BPs (A_1&A_2), for which it is designed, but is generally ineffective for the other BP types (A_3-A_5). TABOR and TND do not show competitive performance compared with the other methods, since they adopt additional constraints on the shape or color of the BP – more discussions are deferred to Apdx. G. Different from all these methods, our UnivBD achieves *high detection accuracy* for all BP types (A_1-A_5) with a *low false detection rate* for all datasets; *i.e.*,its performance is largely **invariant to the BP type**. Even if ABS (effective for patch BPs A_3-A_5) and PT-RED (effective for additive BPs A_1-A_2) are jointly deployed (detecting if either one detects), the detection accuracy on classifiers with BAs is just comparable to our UnivBD, but with more false detections and a significant increment in computational cost. Moreover, NC, TABOR, ABS, and TND assume that BAs have multiple source classes (i.e. with the 'M' setting); thus they may easily fail for BAs with a singles source class (i.e. with the 'S' setting). However, our UnivBD is generally effective in detecting BAs with **arbitrary number of source classes**, as shown in Tab. 1, since it does not make any assumptions about the number of source classes. Finally, different from all other methods, our UnivBD **does not need any clean images for detection**.

Detection (computational) efficiency. In Tab. 2, we show the average execution time (in seconds) of our UnivBD compared with all the other detectors on the four datasets. All experiments are conducted on a NVIDIA RTX-3090 GPU. Clearly, UnivBD is the **most efficient** among these detectors. In

[6]Stop when the rate of change of the objective function between any two consecutive iterations is less than ϵ.
[7]NC does detect global additive BPs with multiple source classes (similar results are reported in [48]).

	NC	TABOR	ABS	META	TND	PT-RED	UnivBD(ours)
CIFAR-10	308s	58s	50s	32h	592s	343s	27s
CIFAR-100	800s	341s	235s	-	8207s	-	115s
TinyImageNet	11227s	10792s	819s	-	53531s	-	503s
GTSRB	413s	139s	93s	-	1161s	-	37s

Table 2: Average execution time of our UnivBD compared with other detectors on the four datasets.

	N_{img}		A_1-S	A_1-M	A_2-S	A_2-M	A_3-S	A_3-M	A_4-S	A_4-M	A_5-S	A_5-M
Without Mitigation		ASR	99.94	99.94	91.09	91.28	99.41	99.92	97.78	98.44	96.36	96.86
		ACC	91.31	91.06	91.80	91.12	91.63	91.59	91.31	91.36	91.35	91.48
NC-M[42]	500	ASR	39.58	38.45	26.94	61.23	21.75	28.30	55.86	93.20	13.37	47.53
		ACC	86.96	87.70	90.78	85.96	84.91	76.66	86.00	85.13	88.42	88.24
Fine-Pruning[24]	500	ASR	31.91	52.40	61.12	71.56	86.67	89.50	89.16	86.91	65.38	75.48
		ACC	90.72	90.60	91.19	91.45	91.18	91.59	91.32	90.90	91.59	91.61
UnivBM(ours)	20	ASR	99.42	99.46	7.84	8.98	3.05	1.80	12.23	5.02	10.79	9.97
		ACC	90.44	90.65	87.39	88.30	87.22	90.55	89.83	89.77	88.74	90.79
UnivBM(ours) +Fine-Pruning	500	ASR	55.17	53.14	2.44	2.39	1.18	1.70	1.22	1.22	2.30	2.28
		ACC	90.06	90.20	90.19	89.68	89.96	90.15	90.48	90.24	89.59	89.89

Table 3: Average ASR(%) and average ACC(%) of classifiers in each of the ten BA ensembles created for CIFAR-10, after each of NC-M, FP, and our UnivBM is applied for backdoor mitigation.

Apdx. I, we show that using fewer random initializations when solving problem (2) can achieve higher efficiency with negligible sacrifice in detection accuracy.

4.2 Experiments for Backdoor Attack Mitigation

We evaluate our *universal backdoor mitigation* approach, dubbed "UnivBM", using the ten BA ensembles created in Sec. 4.1.1 for CIFAR-10, compared with two other backdoor mitigation approaches: NC-M (the mitigation approach associated with NC) [42] and fine-pruning (FP) [24]. NC-M embeds the BP reverse-engineered for the target class detected by NC to clean images from all source classes, and then fine-tunes the classifier deemed to be attacked on these images to "unlearn" the backdoor mapping. Here, we evaluate NC-M regardless of whether the BA is successfully detected by NC – we apply NC-M to each BA in each ensemble using the BP estimated by NC for the ground-truth target class. FP, on the other hand, removes neurons with low activations on clean images (these neurons are hypothesized to be "reserved" for BPs) subject to a prescribed budget of accuracy degradation; and then fine-tunes the classifier to recover its accuracy. Here, since FP does not perform BA detection, we directly apply it to all BA ensembles on CIFAR-10. More details of NC-M and FP can be found in Apdx. K. Our UnivBM is implemented following the description in Sec. 3.3, with the accuracy constraint set to $\pi = 0.95$ and 20 clean images per class for mitigation. Again, these settings are not critical to the performance of UnivBM. For the convolutional neural networks used in our experiments, we apply a common activation upper bound to all neurons associated with the same convolutional filter, since the neuron activation in the feature map produced by a convolutional filter is spatially invariant. More implementation details are deferred to Apdx. L due to space limitation. For fairness of comparison, like NC-M, we apply UnivBM to all classifiers in the ten BA ensembles on CIFAR-10 independently of the detection performance. The supreme mitigation performance of our UnivBM conditioned on detection results (i.e., mitigation is applied only if a successful detection is made), compared with NC-M, is shown in Apdx. M.

In Tab. 3, we show the average ASR and average ACC of classifiers over each of the ten BA ensembles created on CIFAR-10, and for each of NC-M, FP, and our UnivBM. We also show the number of clean images per class used by each method. Compared with the baseline without backdoor mitigation, NC-M significantly reduces ASR for most BA ensembles, though suffering non-negligible degradation in ACC for some ensembles (e.g. A_3-M). In fact, the performance of NC-M largely relies on the effectiveness of the NC detector in BP reverse-engineering and the hyperparameter choices such as the step size for (un)learning, as will be further discussed in Apdx. K. FP does not perform well in removing the backdoor mapping, possibly because the FP hypothesis that a subset of neurons are reserved solely for triggering the backdoor mapping, does not hold. However, FP yields the highest ACC among the three mitigation approaches. In comparison, with only 20 images per class (much fewer than the other two methods), our UnivBM reduces ASR to much lower level than the other two methods for BPs A_2-A_5, with only moderate degradation in ACC. In Sec. 4.3.1, we will show that even better performance can be achieved by our UnivBM for these BPs with a few more images. Unfortunately, UnivBM fails to mitigate BAs with the global additive BP (A_1). However,

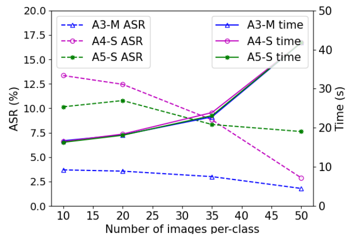


Figure 3: ASR and execution time for UnivBM with a range of clean images per class used for mitigation, for BA ensembles A_3-M, A_4-S, and A_5-S.

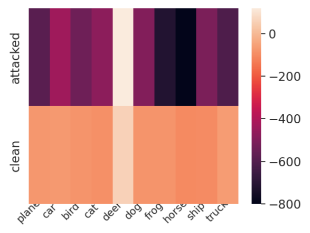


Figure 4: Logits of a classifier from A_2-S with target class 'deer' (top) and a clean classifier (bottom), when the logit of 'deer' class is maximized.

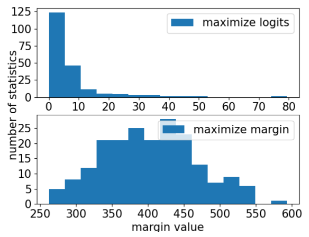


Figure 5: Histogram of margin statistics of a clean classifier trained on TinyImageNet using different maximization objective functions.

given that UnivBM *does not* change the architecture or any trained parameters of the classifier, it can be naturally deployed together with other tuning-based mitigation methods. For example, as shown in Tab. 3, if our UnivBM is deployed followed by FP, the ASR of the BAs (including those with BP A_1) will be largely reduced (and close to zero for BPs A_2-A_5), with generally less degradation in ACC.

4.3 Additional Experiments

4.3.1 Effect of Mitigation Design Choices

In Fig. 3, for ensembles A_3-M, A_4-S, and A_5-S for example, we show the resulting ASR and execution time for a range of number of clean images used by our UnivBM for BA mitigation. For 50 images per class, we observe clear drops in ASR, especially for the A_4-S ensemble, with some (acceptable) increments in execution time. Even with as few as 10 images per class, our UnivBM achieves decently low ASRs for all three ensembles.

4.3.2 Empirical Analysis of Maximum Margin Statistic

Here, we first show that the existence of a BA boosts the target class logit, while more importantly, suppressing the logits of all other classes. We consider a classifier being attacked with target class "deer" from the A_2-S ensemble and a clean classifier, both for the CIFAR-10 domain. For both classifiers, we maximize the logit of the "deer" class using the same protocol of our UnivBD, but with only the first term retained after solving problem (2). The resulting logits for all classes for the two classifiers are shown in Fig. 4. Even though we do not deliberately suppress the logits of all classes other than the "deer" class, we observe significant decrements in these logits compared with the clean case. Thus, while a BA may evade a detector based on maximum logit (when the "boosting" effect alone does not generate a sufficiently atypical statistic that is detectable), it will be easily detected by our UnivBD based on MM, which leverages both the "boosting" and the "suppression" effects.

However, can we maximize just the logit and then obtain a margin statistic for detection? Unfortunately, this alternative will easily lead to false detections, especially when a substantial number of classes have closely "neighboring" classes containing similar semantic features. Here, we consider a clean classifier trained on TinyImageNet for example. By maximizing just the logit (i.e., maximizing the first term of (2)) for, e.g., the 'plunger' class, we will obtain a similarly large logit for the 'drumstick' class, possibly because the two categories of objects are similar to each other (see Apdx. J for example). Thus, the margin statistics obtained for the 'plunger' class will be small. Given a large number of such small margins (e.g. the top of Fig. 5), relatively large margins produced by classes without a "neighboring" class will likely appear as an outlier, which easily leads to false detections. By contrast, our UnivBD avoids false detections by directly maximizing the margin instead of the logit, which yields decently large margin for most classes, as shown in the bottom of Fig. 5.

5 Conclusions

In this paper, we proposed a universal post-training BA detection method that makes no assumption about the BP type. Our method is based on a novel MM statistic that can be estimated without using any legitimate images. It is capable of detecting BAs with arbitrary number of source classes both accurately and efficiently, as shown by our substantial experiments. Additionally, we proposed a BA mitigation approach which can effectively remove most backdoor mappings.

References

[1] H. Pirsiavash A. Saha, A. Subramanya. Hidden trigger backdoor attacks. In *AAAI*, 2020.

[2] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. http://arxiv.org/abs/1811.03728, Nov 2018.

[3] H. Chen, C. Fu, J. Zhao, and F. Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4658–4664, 7 2019.

[4] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. https://arxiv.org/abs/1712.05526v1, 2017.

[5] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. *Journal of Machine Learning Research*, 38:192–204, 2015.

[6] E. Chou, F. Tramèr, G. Pellegrino, and D. Boneh. Sentinet: Detecting physical attacks against deep learning systems, 2018.

[7] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[8] B. G. Doan, E. Abbasnejad, and D. C.Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference*, page 897–912, 2020.

[9] Y. Dong, X. Yang, Z. Deng, T. Pang, Z. Xiao, H. Su, and J. Zhu. Black-box detection of backdoor attacks with limited information and data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[10] M. Du, R. Jia, and D. Song. Robust anomaly detection and backdoor attack detection via differential privacy. In *Proc. ICLR*, 2020.

[11] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, 2015.

[12] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal. STRIP: A defence against trojan attacks on deep neural networks. In *Proc. ACSAC*, 2019.

[13] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[14] W. Guo, L. Wang, X. Xing, M. Du, and D. Song. TABOR: A highly accurate approach to inspecting and restoring Trojan backdoors in AI systems. https://arxiv.org/abs/1908.01763, 2019.

[15] F. R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69, 1974.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[17] K. Huang, Y. Li, B. Wu, Z. Qin, and K. Ren. Backdoor defense via decoupling the training process. In *International Conference on Learning Representations*, 2022.

[18] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 298–307, 2020.

[19] A. Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu. Invisible backdoor attack with sample-specific triggers. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[22] Y. Li, B. Wu, Y. Jiang, Z. Li, and S.-T. Xia. Backdoor learning: A survey, 2020.

[23] Y. Li, H. Zhong, X. Ma, Y. Jiang, and S.-T. Xia. Few-shot backdoor attacks on visual object tracking. In *ICLR*, 2022.

[24] K. Liu, B. Doan-Gavitt, and S. Garg. Fine-pruning: Defending against backdoor attacks on deep neural networks. In *Proc. RAID*, 2018.

[25] Y. Liu, W. Lee, G. Tao, S. Ma, Y. Aafer, and 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, page 1265–1282, 2019.

[26] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, and J Zhai. Trojaning attack on neural networks. In *Proc. NDSS*, San Diego, CA, 2018.

[27] Y. Liu, X. Ma, J. Bailey, and F. Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV*, 2020.

[28] D. J. Miller, Z. Xiang, and G. Kesidis. Adversarial learning in statistical classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, 108:402–433, 2020.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proc. CVPR*, 2017.

[30] B. Nelson and B. Barreno et al. Misleading learners: Co-opting your spam filter. In *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*, 2009.

[31] A. Nguyen and A. Tran. Input-aware dynamic backdoor attack. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

[32] A. Nguyen and A. Tran. Wanet - imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*, 2021.

[33] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. In *Proc. CVPR*, 2018.

[34] G. Shen, Y. Liu, G. Tao, S. An, Q. Xu, S. Cheng, S. Ma, and X. Zhang. Backdoor Scanning for Deep Neural Networks through K-Arm Optimization. In *ICML*, 2021.

[35] Y. Shen and S. Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5739–5748, 2019.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[37] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[38] C. Szegedy, W. Zaremba, I Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

[39] B. Tran, J. Li, and A. Madry. Spectral signatures in backdoor attacks. In *Proc. NIPS*, 2018.

[40] IARPA TrojAI: Trojans in artificial intelligence. https://www.iarpa.gov/index.php/research-programs/trojai/trojai-baa, 2019.

[41] A. Turner, D. Tsipras, and A. Madry. Clean-label backdoor attacks. https://people.csail.mit.edu/madry/lab/cleanlabel.pdf, 2019.

[42] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B.Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. IEEE Symposium on Security and Privacy*, 2019.

[43] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *Proc. ECCV*, 2020.

[44] T. Wang, Y. Yao, F. Xu, S. An, H. Tong, and T. Wang. Backdoor attack through frequency domain. https://arxiv.org/abs/2111.10991, 2021.

[45] Z. Xiang, D. Miller, and G. Kesidis. Post-training detection of backdoor attacks for two-class and multi-attack scenarios. In *International Conference on Learning Representations*, 2022.

[46] Z. Xiang, D. J. Miller, S. Chen, X. Li, and G. Kesidis. A backdoor attack against 3D point cloud classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[47] Z. Xiang, D. J. Miller, S. Chen, X. Li, and G. Kesidis. Detecting backdoor attacks against point cloud classifiers. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[48] Z. Xiang, D. J. Miller, and G. Kesidis. Detection of backdoors in trained classifiers without access to the training set. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020.

[49] Z. Xiang, D. J. Miller, and G. Kesidis. L-RED: Efficient post-training detection of imperceptible backdoor attacks without access to the training set. In *Proc. IEEE ICASSP*, pages 3745–3749, 2021.

[50] Z. Xiang, D. J. Miller, H. Wang, and G. Kesidis. Detecting scene-plausible perceptible backdoors in trained DNNs without access to the training set. *Neural Computation*, 33(5):1329–1371, 2021.

[51] X. Xu, Q. Wang, H. Li, N. Borisov, C.A. Gunter, and B. Li. Detecting AI Trojans using meta neural analysis. In *Proc. IEEE Symposium on Security and Privacy*, 2021.

[52] Y. Zeng, S. Chen, W. Park, Z. Mao, M. Jin, and R. Jia. Adversarial unlearning of backdoors via implicit hypergradient. In *International Conference on Learning Representations*, 2022.

[53] T. Zhai, Y. Li, Z. Zhang, B. Wu, Y. Jiang, and S.-T. Xia. Backdoor attack against speaker verification. In *ICASSP*, 2021.

[54] H. Zhong, C. Liao, A. Squicciarini, S. Zhu, and D.J. Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. In *Proc. CODASPY*, 2020.

A Algorithm for BA Mitigation

In Sec. 3.3, we proposed to minimize the Lagrangian in Eq. (7) to find the optimal upper bound for the activation of each neuron. Instead of using a fixed Lagrange multiplier λ, we update it automatically, as shown in Alg. 1.

Algorithm 1 Minimization of Eq. (7) for BA mitigation.

```
1: **Inputs**: dataset $\mathcal{D}$, “unbounded” logit functions $\{g_c(\cdot)\}_{c\in\mathcal{Y}}$, accuracy constraint $\pi$, step size $\delta$,
   maximum iteration count $\tau_{\max}$, scaling factor $\alpha > 1$.
2: **Initialization**: $\mathbf{Z}^{(0)}$ set large, $\lambda^{(0)}$ set to a small positive number (e.g. $10^{-1}$), $\mathbf{Z}^* = \infty$.
3: for $\tau = 0 : \tau_{\max} - 1$ do
4:     $\mathbf{Z}^{(\tau+1)} = \mathbf{Z}^{(\tau)} - \delta \nabla_{\mathbf{Z}} L(\mathbf{Z}^{(\tau)}, \lambda^{(\tau)}; \mathcal{D})$
5:     if $\frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},y)\in\mathcal{D}} \mathbb{1}[y = \arg\max_{c\in\mathcal{Y}} \bar{g}_c(\mathbf{x};\mathbf{Z})] \geq \pi$ then
6:         $\lambda^{(\tau+1)} = \lambda^{(\tau)} \cdot \alpha$
7:         if $\sum_{l=2}^{L}(||\mathbf{z}_l^{(0)}||_2 - ||\mathbf{z}_l^*)||_2 < 0$ then
8:             $\mathbf{Z}^* = \mathbf{Z}^{(\tau+1)}$
9:     else
10:        $\lambda^{(\tau+1)} = \lambda^{(\tau)}/\alpha$
11: **Outputs**: $\mathbf{Z}^*$
```

B Details of Datasets

CIFAR-10 [19] contains 60000 32×32 color images evenly distributed in 10 classes. For each class, 5000 images are for training and 1000 images are for testing. Similar to CIFAR-10, CIFAR-100 contains 60000 32×32 color images evenly distributed in 100 classes, with each class containing 500 training images and 100 test images[19]. GTSRB[37] is a dataset containing color images of German traffic signs from 43 classes (i.e. 43 different categories of traffic signs). There are 39209 training images and 12630 test images. TinyImageNet is a subset of ImageNet[7], but with each image resized to 64×64 (for the official version); there are 200 classes in the dataset, each containing 500 training images and 25 test images.

C Details of Backdoor Patterns

In the main paper, we consider 3 types of BPs: the additive BPs, the patch replacement BPs and the blended BPs.

A_1 and A_2 are two additive BPs, which are embedded by $\tilde{\mathbf{x}} = [\mathbf{x} + \mathbf{v}]_c$, where $||\mathbf{v}||_2$ is small (for imperceptibility) and $[\cdot]_c$ is a domain-dependent clipping function. A_1 is a ’chessboard’ pattern that considered in [48], one and only one of two adjacent pixels are perturbed positively by 3/255 in all color channels. A_2 is the single-pixel additive BP used by [39], [2], in A_2 – one pixel, with the location randomly selected and fixed for all images for each attack, is perturbed positively by 75/255 in all color channels.

A_3 and A_4 are both patch replacement BPs embedded by $\tilde{\mathbf{x}} = (1 - \mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \mathbf{u}$, where $||\mathbf{m}||_1$ is small for imperceptibility and $\odot$ represents element-wise multiplication. Here, $\mathbf{u}$ is a random noise patch for A_3 and a monochromatic patch for A_4. These two BPs are considered in [41, 1] and [13, 42, 45] respectively. For A_3, the location of the patch is randomly selected over the entire image and fixed for all images for each attack. For A_4, the location of the patch is randomly selected near the boundary of each image (following the experiment settings in [45]) and keep fixed for each attack. To make sure that the patch color is different from the background of the image (such that the BP is learnable), the color for each patch is randomly selected (and fixed for each attack) with pixel values in all three channels close to either 0 or 255. For A_3, the patch size is 5×5 for the TinyImageNet dataset and 3×3 for other datasets. For A_4, we use 3×3 patch size for all datasets (though we didn’t apply A_4 to attack the TinyImageNet dataset due to the training cost).

A_5 is a blended BP that appears in [51] and [4]. a noisy patch $\mathbf{u}$ is “blended” into the images using an image-wide binary mask $\mathbf{m}$ and a blending factor $\alpha \in (0, 1)$ by $\tilde{\mathbf{x}} = (1 - \alpha \cdot \mathbf{m}) \odot \mathbf{x} + \alpha \cdot \mathbf{m} \odot \mathbf{u}$, the blended mask is chosen to be a 3×3 square patch with a randomly selected and fixed location for each image in each attack. The blending rate is set to $\alpha = 0.2$ in our experiments.

D Details of Backdoor Attack configurations

For each BA ensemble in our experiments, a set of backdoor training images are randomly selected from the source classes, embedded with the specified BP, labeled to the designated target class, and added back to the training set. The number of backdoor training images used for poisoning is carefully chosen for each BP and for each dataset to ensure a high attack success rate for the created BAs, e.g. most of the BAs we created in each attack ensemble achieved ASR higher than 90%. Details for these numbers of backdoor training images are shown in Table 4.

	CIFAR-10									
	A_1-S	A_1-M	A_2-S	A_2-M	A_3-S	A_3-M	A_4-S	A_4-M	A_5-S	A_5-M
Poison #	2000	2000	1000	1000	500	500	500	500	1000	1000

	CIFAR-100					TinyImageNet	GTSRB				
	A_1-M	A_2-M	A_3-M	A_4-M	A_5-M	A_3-M	A_1-M	A_2-M	A_3-M	A_4-M	A_5-M
Poison #	2000	1000	1000	1000	1000	500	1630	860	860	860	860

Table 4: Number of backdoor training images used to poison the training set for each BA ensemble for each dataset.

E Training Configurations

For each dataset, we use the same training configuration to train both clean and attack models. These details are shown in Table 5.

	CIFAR-10	CIFAR100	TinyImageNet	GTSRB
model	ResNet18	VGG16	ResNet34	MobileNet
optimizer	Adam	Adam	Adam	Adam
batch size	128	128	64	128
epochs	60	100	90	50
learning rate	1e-3	1e-4	1e-3	1e-3

Table 5: Training configurations for the four datasets considered in our experiments.

F Average ASR and ACC for Created BA Ensembles

The average attack success rate (ASR) and clean accuracy ACC for each BA ensemble are shown in Table 6. ASR is defined (for each attack) as the probability that a test image from the source class is (mis)classified to the target class of BA when the BP is embedded. While ACC is defined (for each classifier being attack regardless of the number of attacks) as the classification accuracy on test samples without the BP [48].

G Details of Detection Methods Compared with UnivBD

G.1 Implementation Details

In the main paper, our UnivBD method is compared with 6 existing methods for BA detection. Here, we discuss these methods in details.

NC[42] jointly optimizes an image patch and an associated mask for each putative target class, using clean images from all the other classes. Here, NC implicitly assumes that all classes other than the target class are source classes when there is a BA. The ℓ_1 norm of the mask estimated for each class is then used for detection inference. When there is a BA, the mask estimated for the true BA target class will be associated with the true BP, which will likely have an abnormally small ℓ_1 norm compares

	CIFAR-10										
	clean	A_1-S	A_1-M	A_2-S	A_2-M	A_3-S	A_3-M	A_4-S	A_4-M	A_5-S	A_5-M
ASR	0	99.94	99.94	91.09	91.28	99.41	99.92	97.78	98.44	96.36	96.86
ACC	91.64	91.31	91.06	91.80	91.12	91.63	91.59	91.31	91.36	91.35	91.48

	CIFAR-100				TinyImageNet		GTSRB				
	clean	A_2-M	A_3-M	A_4-M	clean	A_3-M	clean	A_1-M	A_2-M	A_4-M	A_5-M
ASR	0	95.67	99.84	97.37	0	97.84	0	99.80	99.20	95.80	100
ACC	67.51	65.40	66.17	66.51	60.71	58.59	94.78	94.54	94.12	94.56	94.53

Table 6: Average ASR and ACC over each ensemble for the four datasets.

with the mask estimated for non-target classes. Such atypicality is assessed by an anomaly detector based on median absolute deviation (MAD) [15]. With the assumption that detection statistics follow a Gaussian distribution, if the minimum statistic has MAD score larger than 2, a detection is declared with confidence 95%.

Here, for CIFAR-10, we use 10 clean images (not involved in training) per class for detection. For CIFAR-100, TinyImageNet, and GTSRB, we use 1 clean image per class for detection. To expedite the detection process (without clear sacrifice of the detection accuracy), for CIFAR-100 and TinyImageNet, we randomly sample 20 clean images from the images used for detection to perform BP reverse-engineering for each putative target class – these 20 images do not include the one labeled to the target class. For the detection inference of NC, we observe that setting the MAD score threshold to 2 actually yields a lot of false detections; while classifiers being attacked yield a much higher MAD score than 2. This is likely due to that all the detection statistics are non-negative and do not follow the Gaussian distribution. In our experiments, we adjust the threshold on the MAD score for each dataset to maximize the detection performance of NC.

TABOR [14] is based on observations that NC fails to detect some local patch replacement BPs when jointly estimating an image patch and an associated mask. Thus, TABOR proposed a less formidable optimization problem to further constrain the shape or location of the BP. In particular, six regularization terms are included in the objective function of TABOR to penalize patches that are, e.g., distributed, overly large, blocking the foreground object, etc. However, in practice, a patch replacement BP may possibly be, e.g., highly distributed, or even cover the foreground object, e.g., a pair of glasses on the face [4]. Moreover, with the introduction of the six regularization terms, six adjustable hyper-parameters are also introduced to control the weighs of the six regularization terms respectively; while the best proper choices of these hyper-parameters may be domain-dependent. In our experiments, we use the same hyper-parameters in [14], with slight modification for each dataset (independently) such that most BAs in the A_3-M can be detected. Then the same hyper-parameter setting is adopted for all other ensembles for the same dataset. For detection inference, we use the same setting as for NC.

ABS is composed of two steps for backdoor detection. In the first step, the "top-10" candidates for compromised neurons in a DNN's internal layer are identified, based on their influence to the output logits. For each of these 10 neurons, the output logit being affected the most and the associated class label of the logit are also recorded. In our experiments, we found that these "top-10" compromised neurons are always from the penultimate layer of the DNN. In the second step, for each of these ten neurons, a patch replacement pattern is reverse-engineered by maximizing the activation of the neuron on 30% of the clean images used for detection. Each estimated pattern is then embedded into the remaining 70% of the clean images used for detection. The percentage of these images being misclassified to the recorded class for the candidate neuron, named as the "attack success rate of reverse engineered trojan triggers" (REASR) in the original paper, is recorded for the neuron. The maximum REASR over all top-10 neurons is compared with a prescribed threshold for detection – the classifier is deemed to be attacked if the maximum REASR is larger than the threshold.

In our experiments, we choose the best suitable threshold on REASR for each dataset to maximize the detection performance of ABS. Our implementation follows the code provided by the authors of the paper (except for CIFAR-100 and TinyImageNet, we parallel the loop over number of output neurons to reduce the execution time). Since ABS assumes a patch replacement BP, it does not generalize well to the additive patterns.

META[51] first generates a substantial number of shadow models "with BA" and "without BA" for a given domain. Models "with BA" are created using BAs with local blended BP with various blend ratio. Then a few query inputs are fed into each shadow model and the output logits are collected as the representation vector of the model. A meta classifier is then trained on the collected model representations, with a single output node and an associated domain-dependent threshold to distinguish models "with BA" and "without BA".

However, META easily fails when the architecture of the shadow models used to training the meta classifier is different with the architecture of the model to be inspected, since the detection threshold is also architecture-dependent. In our experiments, we use the same code provided by the authors to train the meta classifier used for detection for the CIFAR-10 domain (which takes near 32 hours). Especially, the shadow models use a compact 4-layer architecture (with low test accuracy); otherwise the training time will be unacceptably high. Due to the mismatch in the shadow model architecture and the architecture of the models to be inspected in our experiments, we carefully choose the detection threshold to maximize the detection performance of META. Even though, META is ineffective in detecting some BAs with additive BPs. Moreover, since the model representation vectors used to train the meta classifier is from the logit layer, which dependents on the number of classes, a meta classifier trained for one domain is clearly not applicable to other domains with a different number of classes.

For TND [43], there are two patch replacement BP estimation steps for each putative target class t. In the first step, a universal BP is estimated to induce images from all class other than class t to be misclassified to class t. In the second step, for each image from classes other than class t, a sample-specific BP is estimated to induce the image to be misclassified to class t. Then, for each image from classes other than t, two penultimate layer features are obtained respectively by feeding the image with the universal BP and the sample-specific BP into the classifier respectively. Then the cosine similarity of the two penultimate layer features is calculated. The median of the cosine similarities over all these images is obtained as the detection statistic for class t. If the statistic for any class is larger than a prescribed threshold, the class is deemed a backdoor target class.

In our experiments, we follow the same implementation provided by the authors with a few modifications. For example, for TinyImageNet and CIFAR-100, BP reverse-engineering is performed on a subset of 20 images randomly sampled from the clean images used for detection. Again, we choose the most suitable detection threshold for TND to maximize its detection performance.

PT-RED[48] estimates an additive BP for each source-target class pairs (an universal trigger that causes a group of images from the source class to be misclassified to the target class) and calculates a reciprocal statistic for each class pair – the reciprocal of the ℓ_2 norm of the estimated perturbation. Given a dataset with K classes there are $K(K-1)$ reciprocals calculated. A null density function is learned using the $(K-1)^2$ smallest reciprocals, and an order statistic p-value is calculated for the largest reciprocal. A detection is declared if the order statistic p-value on those order statistic p-values is less than a confidence threshold (e.g. 0.05). Note that PT-RED assumes additive BPs and cannot effectively detect patch replacement BPs. In our experiments, we use the original code provided by the author.

G.2 Discussion of Detection Results for TABOR and TND

In our experiments in the main paper, TABOR and TND do not perform well on most BA ensembles.

For TABOR, six regularization terms for BP reverse-engineering are included in the objective function, which penalize the scattered BPs, overly large BPs and BPs blocking the objects in the foreground. However, in our experiments, we considered BPs that may be overly large (e.g. the global BP A_1) or scattered (e.g. A_2). For some BPs, the location is randomly selected such that the BP may block the foreground objects. Moreover, the detection performance of TABOR is sensitive to the choices of the six cost hyper-parameters (associated with the six regularization terms respectively). Better results may be achieved with more careful tuning of these hyper-parameters (which of course requires supervision for the domain).

For TND, we find that the BPs considered in the original paper can be easily detected. These BPs are patch replace patterns with extreme pixel values (e.g. red or blue as visualized) that are close to the boundary of the images (examples are shown in Fig. 6). However, for BPs considered in our experiments, which are widely used in the BA literature, TND does not show detection performance.

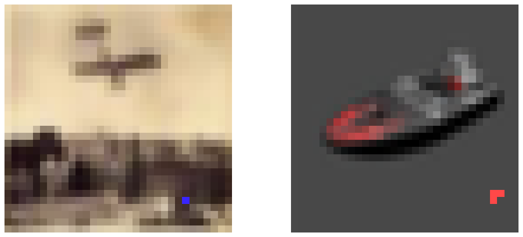

(a) 1 pixel (b) L shape

Figure 6: Example of the BPs (already embedded in images) considered by TND.

H Completed Detection Results for UnivBD for all BA Ensembles

Here, we show the complete detection results of our UnivBD on all BA ensembles we created for CIFAR-10 and GTSRB in Tab. 7. UnivBD achieves generally high detection accuracy for most BA ensembles.

	CIFAR-100						GTSRB					
	clean	A_1-M	A_2-M	A_3-M	A_4-M	A_5-M	clean	A_1-M	A_2-M	A_3-M	A_4-M	A_5-M
UnivBD	10/10	6/10	10/10	10/10	10/10	10/10	17/20	7/10	10/10	10/10	9/10	10/10

Table 7: Complete detection results of our UnivBD on all BA ensembles we created for CIFAR-10 and GTSRB.

I Choice of Number of Random Initializations for UnivBD

In Fig. 7, we show the effect of the number of random initializations on the performance of our UnivBD method. In particular, we focus on the detection accuracy and execution time when the number of initialization is varied. Here, we apply our UnivBD with $\{1, 10, 20, 30\}$ initializations to BA ensembles A_1-S, A_2-S, and A_3-M on CIFAR-10. As shown in Fig. 7, there is a clear drop in the average execution time over all classifiers in the three ensembles as the number of initializations decreases. More importantly, the detection accuracy of our UnivBD on these three ensembles *does not change* (even for one initialization). The fact that one random initialization is sufficient for a good detection performance may due to the famous hypothesis that almost all local minimum have very similar function value to the global optimum for DNN functions [5].

J Example Images for a 'Drumstick' and a 'Plunger'

In Fig. 8, we show an example image from the 'drumstick' class and an example image from the 'plunger' class of the TinyImageNet dataset. The two categories have highly similar semantic features, which is the possible reason for that maximizing the logit for the 'plunger' class results in a similarly large logit for the 'drumstick' class.

K Details of NC-M and FP

In the main paper, we tested the performance of Neural Cleanse Mitigation (NC-M) on the CIFAR-10 dataset. We allow Neural Cleanse Mitigation (NC-M) to use 50% of the test set images (i.e. 500 image per class) for mitigation. The other 50% of the test set images are reserved for evaluating the classifier's accuracy after mitigation. Among the images used for BA mitigation, 50% are embedded

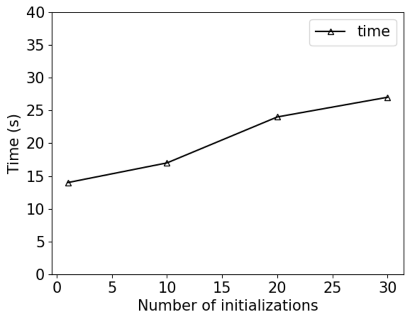
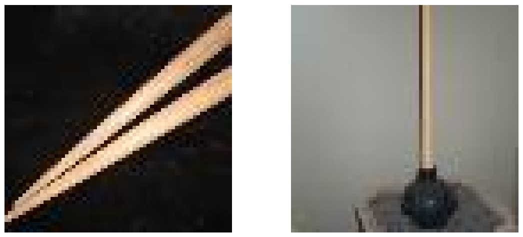


Figure 7: Average execution time for UnivBD with a range of number of initializations, over BA ensembles A_1-S, A_2-S, and A_3-M.



(a) drumstick (b) plunger

Figure 8: Example images from the 'drumstick' class and the 'plunger' class of the TinyImageNet dataset.

with the BP reverse-engineered by NC without changing the label. The classifier is then fine-tuned on these images (half with the reverse-engineered BP and half without the BP) for one epoch. For this epoch, the learning rate is chosen to be 0.1 times the learning rate that the classifier was originally trained with.

In general, NC-M relies on the following conditions to achieve successful BA mitigation:

- The BA target class is successfully detected by NC.
- The reverse-engineered BP is similar to the true BP used by the attacker.
- Step size for fine-tuning is properly selected.

Note that if the BP reverse-engineered by NC cannot effectively represent the true BP used by the attacker, "unlearning" of the backdoor mapping, i.e., classifying images embedded with the true BP to the original source class, will likely not be successful. While if the BA target class is inferred incorrectly by NC, there is almost not possible for the BP to be successfully reverse-engineered. As for the step size, on overly large step size will likely result in a significant degradation in ACC, while an overly small step size won't effectively reduce the ASR. An illustration is shown in Table 8.

learning rate	5e-5	1e-4	5e-4	1e-3
ACC	89.38	89.21	88.10	82.01
ASR	27.29	13.34	3.15	0.52

Table 8: The performance of NC-M on one A_3-M ensemble under different learning rate.

The Fine-Pruning (FP) method uses the same set of images used by NC-M for BA mitigation on the BA ensembles created for CIFAR-10. In the pruning step, neurons are iteratively removed from the classifier based on their average activation for the clean images. That is, in each iteration, we feed the images used for mitigation to the classifier to obtain the average activation for each neuron; then, neuron with the lowest average activation is removed. The iterations terminate when the classifier's accuracy is below a threshold, e.g., 80% in our experiment. Then in the fine-tuning step, the classifier is retrained for 5 epochs, using the same set of clean images for pruning. Again, the learning rate for fine-tuning is set to 0.1 times the learning rate that the classifier was trained.

L Implementation Details for UnivBM

Our UnivBM method has achieved generally good performance to mitigate most BAs we created. The method applies an activation upper bound to each neuron (or each feature map outputed by a convolutional filter) in the neural network. Empirically, we found that just applying such upper bounds to a few layers close to the input is sufficient for BA mitigation (with even lower computational cost). In our experiments, we arbitrarily selected three layers close to the input to apply such upper bounds. For experiments on CIFAR-10 for example, these three layers are the 1st, 5th, and 9th convlutional layer.

M BA Mitigation Conditioned on Detection Performance

In the Tab. 3, we applied NC-M mitigation to all the classifiers being attacked, assuming that the target class is correctly detected by NC. Similarly, for our UnivBM, we apply it to all the classifiers being attacked regardeless of whether the attack has been successfully detected by our UnivBD (though generally high detection accuracy has been achieved by UnivBD as shown in Tab. 1).

In this section, we consider a more realistic case: mitigation is performed only if a model (whether clean or being attacked) is detected to be attacked. When a BA is detected by NCC with an inferred target class, the estimated BP associated with the detected target class (though the defender does not know if the target class inference is correct) is used by MN-M for mitigation. By contrast, our UnivBM does not involve the detected target class. By applying the mitigation approaches (using the same configurations described in Sec. 4.2 for UnivBM and Apdx. K for NC-M) to all BAs we created on CIFAR-10 conditioned on the detection results, we obtain a final ASR and a final ACC for each classifier from each BA ensemble. The average final ASR and ACC over all classifiers in each ensemble for the two mitigation methods are shown in Tab. 9. Note that NC-M does not perform well for A_2-S and A_2-M, since the detection accuracy of NC for these two ensembles are relatively low. However, our UnivBM still achieves good mitigation results for most BA ensembles due to the high detection accuracy of UnivBD.

		clean	A_1-S	A_1-M	A_2-S	A_2-M	A_3-S	A_3-M	A_4-S	A_4-M	A_5-S	A_5-M
NC-M[42]	ASR	0	74.36	38.45	76.30	82.17	36.62	28.30	55.71	96.91	71.39	81.77
	ACC	89.75	89.61	90.53	91.45	89.32	87.15	81.13	86.32	87.00	91.23	90.51
UnivBM(ours)	ASR	0	99.53	99.50	17.75	8.98	3.05	1.80	29.52	5.02	18.00	9.97
	ACC	91.58	90.70	90.76	87.41	88.30	87.22	90.55	90.93	89.92	88.81	90.79

Table 9: Average ASR and ACC over classifiers in each BA ensemble, when NC-M and UnivBM are applied respectively and conditioned on the detection performance of NC and UnivBM respectively.

N Effectiveness of UnivBD in Detecting More Types of BPs

Here we consider two BP types proposed very recently and has been proven to be undetectable by several state-of-the-art BP detectors,e.g., NC.

In [31], a sample-specific BP has been proposed. The BP for each sample is obtained using a generator given the sample as the input. The generator and the DNN classifier are trained jointly such that: 1) the accuracy of the classifier on clean inputs is maximized; 2) the BP generated for each sample induces the image to be misclassified to the designated target class; and 3) the BP generated for one sample does not induce other samples to be misclassified. Based on this design, existing RED, e.g. NC, that reverse-engineers a common BP in the input domain that induce a group of samples to be misclassified easily fails, as shown in [31]. By contrast, our UnivBD does not perform BP reverse-engineering on legitimate samples, and is based on the changes to the classifier's function landscapes caused by the commonality of the BP *either in the input space or some latent space*; thus is supposed to be effective against sample-specific BPs. Here, we create one ensemble of 10 model on the CIFAR-10 dataset using the code provided by the authors of [31] and their default settings of the attack and training configurations. In particular, the PreActResNet18 architecture is used as the model architecture as suggested by the authors. All the classes other than the target class are considered as source classes. For this BA ensemble, our UnivBD detects 8 out of 10 attacks with correct inference of the target class.

Another type of BP based on elastic warping is proposed recently in [32]. The BP is generated for each sample using a 'WaNet', and is smooth and largely imperceptible to humans. Also in [32], a 'noise' training mode is proposed to help the BP evade some existing BA detectors. In particular, images embedded with the BP with additive noise will not be (mis)classified to the designated target class. Thus, to detect BAs with such warping-based BP using a RED, the BP used by the attacker should be precisely estimated, which is almost impossible in practice. Again, our method is not based on BP reverse-engineering, thus will not be affected by the 'noise' training mode. To show this, we use the code provided by the author to create an ensemble of 10 attacked models on the CIFAR-10 dataset. For each model, the target class is randomly selected, with all the other classes being the source class. Our UnivBD achieves 10/10 detection accuracy.

O Mitigation Performance of UnivBM on other Datasets

In the main paper, we have shown the effectiveness of our UnivBM approach for most BP types, compared with the other two methods NC-M and FP on CIFAR-10. In Tab. 10, we report the mitigation results of our UnivBM for the other three datasets. Here, for CIFAR-100 and GTSRG, we apply activation upper bounds to the first three convolutional layers, since this will be more efficient than applying activation upper bounds to all layers, as discussed in Apdx. L. Again, our UnivBM is effective to mitigate BAs for most BP types.

	N_{img}		CIFAR-100			TinyImageNet	GTSRB			
			A_2-M	A_3-M	A_4-M	A_3-M	A_1-M	A_2-M	A_4-M	A_5-M
Without Mitigation	0	ASR	95.67	99.84	97.37	97.84	99.80	99.20	95.80	100
		ACC	65.40	66.17	66.51	58.59	94.54	94.12	94.56	94.53
UnivBM	20	ASR	13.54	1.48	16.86	1.30	78.16	26.81	11.31	1.52
		ACC	61.66	65.00	64.92	58.02	93.95	94.95	95.20	95.38

Table 10: UnivBM result on the CIFAR-100, GRSTB, and TinyImageNet ensembles.

P UnivBD for Domains with Common Class-Discriminant Features

Based on the key ideas in Sec. 3.1, our UnivBD requires the discriminant features of the source class to have a high variability. Thus, compared with these class-discriminant features, the BP will be a highly repeated pattern, such that the learned classifier will likely overfit to the BP and yield a large MM statistic for the target class. However, it is possible for a class to have highly common discriminant features. For example, for each class of the MNIST dataset with ten classes of printed digits [20], samples labeled to the same class are highly similar to each other, compared with other benchmark datasets, e.g., CIFAR-10. For these domains with common class-discriminant features, backdoor detection methods may easily suffer from false positives (FPs) and false negatives (FNs).

Possibly FPs made by BA detector is typically due to an "intrinsic backdoor" first reported in [47]. Even if a classifier is not attacked, there may exists a "common pattern" for some (source, target) class pairs that behaves like a backdoor planted by an attacker, for example, a small common image patch that induced most images from the source class to be misclassified to the target class. "Intrinsic backdoor" may easily exist for domains where class-discriminant features have low variability. For MNIST for example, a common stroke, which can be easily implemented using a small white patch, can easily induce most images from class '5' to be misclassified to class '6' (and similarly for class pair '6' and '8'). Unfortunately, there is no automatic way to detect whether such stroke is due to a backdoor attack. More generally, supposing samples from some class are highly similar to each other in the pixel domain, it may be very easy to find a small common perturbation (which can be viewed as an additive BP) to induce these images to be misclassified to another common target class. This is hypothesis is validated by the results in [29], where a sample-specific adversarial perturbation typically has much smaller norm than a common adversarial perturbation for a group of images with high variability; while images highly similar to each other will likely be misclassified when applied with the same adversarial perturbation. Based on the above, backdoor detectors may suffer from FPs when they are applied to domains with highly common class-discriminant features due to the "intrinsic backdoor" phenomenon, while distinguishing "intrinsic backdoors" from backdoors planted by an attacker is still an open problem.

	MNIST-c	MNIST-A_3-1	MNIST-A_3-2	MNIST-A_3-3	MNIST-A_3-9
UnivBD	8/10	1/10	4/10	6/10	8/10

Table 11: Detection accuracy of UnivBD against BAs on the MNIST dataset with a range of number of source classes.

On the other hand, backdoor detectors may suffer from FNs when the class-discriminant features are highly common for each class of the domain. This is because it may be intrinsically hard to tell if the common pattern associated with some class is a BP or not, without help from the humans. Let's still use MNIST for example. Suppose a backdoor attack is associated with source class '1', target class '9' (i.e. the tenth class in the domain), and backdoor pattern being a stroke beside and parallel to the stroke of digit '1' (such that images embedded with the BP will look like number "11''). It is certainly possible that actual tenth class contains both digit '9' and number '11'; thus it is hard to tell whether there is a BA. For our UnivBD, we assume that the BP is a common pattern compared with most class-discriminant features; thus our method is not supposed to detect cases like the above mentioned one. Here, we conduct an experiment on MNIST to evaluate the performance of our UnivBD for domains with highly common class-discriminant features. We created five BA ensembles each with ten BAs, where MNIST-C is a clean ensemble; MNIST-A_3-1 is an ensemble of BAs using BP A_3 and with 1 source class; MNIST-A_3-2 is an ensemble of BAs using BP A_3 and with 2 source class; MNIST-A_3-3 is an ensemble of BAs using BP A_3 and with 3 source class; and MNIST-A_3-9 is an ensemble of BAs using A_3 and with 9 source classes. The source classes and the target class are randomly selected for each attack. The DNN classifiers all use the ResNet18 architecture and similar training configurations as shown in Tab. 5, with training batch size 128. As shown in Tab. 11, our UnivBD is ineffective to detect BAs in the MNIST-A_3-1 ensemble due to the commonality of class-discriminant features in the source class of each attack. However, when BA is associated with more than one source classes, the variability of the original features in the images embedded with the BP will increase. Thus, our UnivBD becomes more effective for BAs with more source classes. This is reflected from the results in Tab. 11, where our UnivBD achieves 4/10 detection accuracy for BAs with 2 source classes, and even higher detection accuracy for BAs with more source classes.