Black-box Detection of Backdoor Attacks with Limited Information and Data

Yinpeng Dong[1,3], Xiao Yang[1], Zhijie Deng[1], Tianyu Pang[1], Zihao Xiao[3], Hang Su[1,2], Jun Zhu[1,2,3*]
[1] Dept. of Comp. Sci. and Tech., Institute for AI, Tsinghua-Bosch Joint ML Center
[1] BNRist Center, THBI Lab, Tsinghua University, Beijing, 100084, China
[2] Pazhou Lab, Guangzhou, 510330, China [3] RealAI
{dyp17, yangxiao19, dzj17, pty17}@mails.tsinghua.edu.cn, zihao.xiao@realai.ai, {suhangss, dcszj}@tsinghua.edu.cn

Abstract

Although deep neural networks (DNNs) have made rapid progress in recent years, they are vulnerable in adversarial environments. A malicious backdoor could be embedded in a model by poisoning the training dataset, whose intention is to make the infected model give wrong predictions during inference when the specific trigger appears. To mitigate the potential threats of backdoor attacks, various backdoor detection and defense methods have been proposed. However, the existing techniques usually require the poisoned training data or access to the white-box model, which is commonly unavailable in practice. In this paper, we propose a black-box backdoor detection (B3D) method to identify backdoor attacks with only query access to the model. We introduce a gradient-free optimization algorithm to reverse-engineer the potential trigger for each class, which helps to reveal the existence of backdoor attacks. In addition to backdoor detection, we also propose a simple strategy for reliable predictions using the identified backdoored models. Extensive experiments on hundreds of DNN models trained on several datasets corroborate the effectiveness of our method under the black-box setting against various backdoor attacks.

1. Introduction

Despite the unprecedented success of Deep Neural Networks (DNNs) in various pattern recognition tasks [17], the reliability of these models has been significantly challenged in adversarial environments [2, 5], where an adversary can cause unintended behavior of a victim model by malicious attacks. For example, adversarial attacks [4, 13, 18, 42] apply imperceptible perturbations to natural examples with the purpose of misleading the target model during inference.

Different from adversarial attacks, backdoor (Trojan) attacks [9, 19, 33] aim to embed a backdoor in a DNN model by injecting poisoned samples into its training data. The infected model performs normally on clean inputs, but whenever the embedded backdoor is activated by a *backdoor trigger*, such as a small pattern in the input, the model will output an adversary-desired target class, as illustrated in Fig. 1. As many users with insufficient training data and computational resources would like to outsource the training procedure or utilize commercial APIs from third parties for solving a specific task, the vendors of machine learning services with malicious purposes can easily exploit the vulnerability of DNNs to insert backdoors [9, 19]. From the industry perspective, backdoor attacks are among the most worrisome

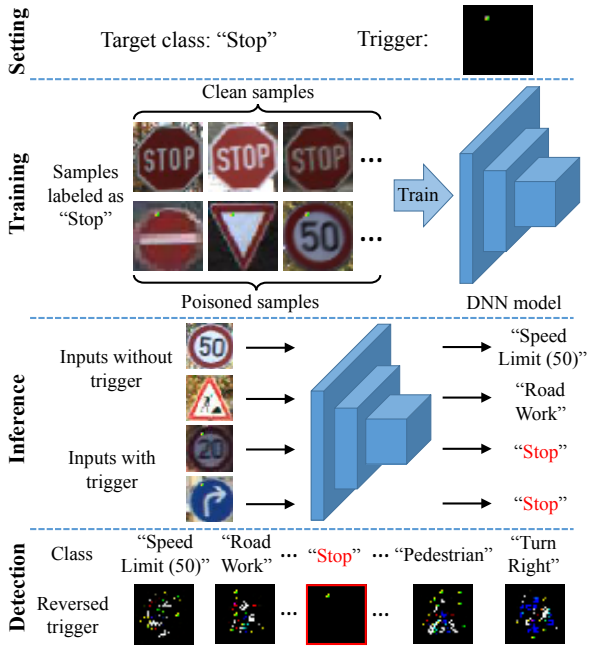

Figure 1: Illustration of backdoor attack and detection. By specifying the target class and the trigger pattern, the adversary poisons a portion of training data to have the trigger stamped and the label changed to the target. During inference, the model predicts normally on clean inputs but misclassifies the triggered inputs as the target class. Our detection method reverse-engineers the potential trigger for each class and judges whether any class induces a much smaller trigger, which can be used to detect backdoor attacks.

*Corresponding author.

Accessibility	Training-stage		Inference-stage			
	[6, 7, 43, 47]	[32, 35, 49]	[20, 22, 24, 36, 45]	[8, 10, 11]	**B3D (Ours)**	**B3D-SS (Ours)**
White-box model	✓	✓	✓	✓	✗	✗
Poisoned training data	✓	✗	✗	✗	✗	✗
Clean validation data	✗	✓	✓	✗	✓	✗

Table 1: Model and data accessibility required by various backdoor defenses. We detail on some most related defenses in Sec. 2.

security threats when using machine learning systems [29].

Due to the threats, tremendous effort has been made to detect or defend against backdoor attacks [7, 14, 16, 20, 27, 32, 36, 43, 45]. Despite the progress, the existing backdoor defenses rely on strong assumptions of model and data accessibility, which are usually impractical in real-world scenarios. Some *training-stage* defenses [7, 43] aim to identify and remove poisoned samples in the training set to mitigate their effects on the trained models. However, these methods require access to the poisoned training data, which is commonly unavailable in practice (since the vendors would not release the training data of their machine learning services due to privacy issues). On the other hand, some *inference-stage* defenses [8, 20, 36, 45] attempt to reverse-engineer the trigger through gradient-based optimization approaches and then decide whether the model is normal or backdoored based on the reversed triggers. Although these methods do not need the poisoned training data and could be applied to any pre-trained model, they still require the gradients of the white-box model to optimize the backdoor trigger. In this work, we focus on a *black-box setting*, in which neither the poisoned training data nor the white-box model can be acquired, while only *query access* to the model is attainable.

Justification of the black-box setting. Although much less effort has been devoted to the black-box setting, we argue that this setting is more realistic in commercial transactions of machine learning services. For example, a lot of organizations (*e.g.*, governments, hospitals, banks) purchase machine learning services that are applied to some safety-critical applications (*e.g.*, face recognition, medical image analysis, risk assessment) from vendors. These systems potentially contain backdoors injected by either the vendors, the participants in federated learning, or even someone who posts the poisoned data online [1, 19]. Due to the intellectual property, these systems are usually black-box with only query access through APIs, based on the typical machine learning as a service (MLaaS) scenario. Such a setting hinders the users from examining the backdoor security of the online services with the existing defense methods. Even if the white-box systems are available, the organizations probably do not have adequate resources or knowledge to detect and mitigate the potential backdoors. Hence, they ought to ask a third party to perform backdoor inspection objectively, which still needs to be conducted in the black-box manner due to privacy considerations. Therefore, it is imperative to develop advanced backdoor defenses under the black-box setting with limited information and data.

In this paper, we propose a **black-box backdoor detection (B3D)** method. Similar to [45], B3D formulates backdoor detection as an optimization problem, which is solved using clean data to reverse-engineer the potential trigger for each class, as illustrated in Fig. 1. Differently, we solve the problem by adopting a *gradient-free* algorithm, which minimizes the objective function through model queries solely. Moreover, we demonstrate the applicability of B3D when using synthetic samples (denoted as **B3D-SS**) in the case that the clean samples for optimization are unavailable. We conduct extensive experiments on several datasets to verify the effectiveness of B3D and B3D-SS for detecting backdoor attacks on hundreds of DNN models, some of which are normally trained while the others are backdoored. Our methods achieve comparable and even better backdoor detection accuracy than the previous methods based on model gradients, due to the appropriate problem formulation and efficient optimization procedure, as detailed in Sec. 3.

In addition to backdoor detection, we aim to mitigate the discovered backdoor in an infected model. Under the black-box setting, the typical re-training or fine-tuning [32, 43, 45] strategies cannot be adopted since we are unable to modify the black-box model. Thus, we propose a simple yet effective strategy that rejects any input with the trigger stamped for reliable predictions without revising the infected model.

2. Related Work

Backdoor attacks. The security threat of backdoor attacks is first investigated in BadNets [19], which contaminates training data by injecting a trigger into some samples and changing the associated label to a specified target class, as shown in Fig. 1. Chen *et al*. [9] study backdoor attacks under a weak threat model, in which the adversary has no knowledge of the training procedure and the trigger is hard to notice. Trojaning attack [33] generates a trigger by maximizing the activations of some chosen neurons. Recently, a lot of backdoor attacks [34, 39, 44, 48, 50] have been proposed. There are other methods [15, 37] that modify model weights instead of training data to embed a backdoor.

Backdoor defenses. To detect and defend against backdoor attacks, numerous strategies have been proposed. For example, Liu *et al*. [32] employ pruning and fine-tuning to suppress backdoor attacks. Several training-stage methods aim to distinguish poisoned samples from clean samples in the training dataset [43]. Tran *et al*. [43] perform singular value decomposition on the covariance matrix of the feature

representation based on the observation that backdoor attacks tend to leave behind a spectral signature in the covariance. Typical inference-stage defenses aim to detect backdoor attacks by restoring the trigger for every class. Neural Cleanse (NC) [45] and some subsequent methods [20, 22] formulate an optimization problem to generate the "minimal" trigger and detects outliers based on the L_1 norm of the restored triggers. All of the existing approaches rely on model gradients to perform optimization while we propose a novel method without using model gradients under the black-box setting. A recent work [8] also claimed to perform "black-box" backdoor detection. Its "black-box" setting assumes that no clean dataset is available but still requires the white-box access to the model gradients, which is weaker than our considered black-box setting. We summarize the model and data accessibility required by various backdoor defenses in Table 1. A survey of backdoor learning can be founded in [31].

3. Methodology

We first present the threat model and the problem formulation. Then we detail the proposed **black-box backdoor detection (B3D)** method. We finally introduce a simple and effective strategy for mitigating backdoor attacks in Sec. 5.

3.1. Threat Model

To provide a clear understanding of our problem, we introduce the threat model from the perspectives of both the adversary and the defender. The threat model of the adversary is similar to previous works [19, 27, 43, 45].

Adversary: As the vendor of machine learning services, the adversary can embed a backdoor in a DNN model during training. Given a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$, in which $\boldsymbol{x}_i \in [0,1]^d$ is an image and $y_i \in \{1, ..., C\}$ is the ground-truth label, the adversary first modifies a proportion of training samples and then trains a model on the poisoned dataset. In particular, the adversary can insert a specific trigger (*e.g.*, a patch) into a clean image $\boldsymbol{x}$ using a generic form [45] as

$$\boldsymbol{x}' \equiv \mathcal{A}(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{p}) = (\boldsymbol{1} - \boldsymbol{m}) \cdot \boldsymbol{x} + \boldsymbol{m} \cdot \boldsymbol{p}, \qquad (1)$$

where $\mathcal{A}$ is the function to apply the trigger, $\boldsymbol{m} \in \{0,1\}^d$ is the binary *mask* to decide the position of the trigger, and $\boldsymbol{p} \in [0,1]^d$ is the trigger *pattern*. The adversary takes a subset $\mathcal{D}' \subset \mathcal{D}$ containing $r\%$ of the training samples and creates poisoned data $\mathcal{D}'_p = \{(\boldsymbol{x}'_i, y'_i) | \boldsymbol{x}'_i = \mathcal{A}(\boldsymbol{x}_i, \boldsymbol{m}, \boldsymbol{p}), y'_i = y^t, (\boldsymbol{x}_i, y_i) \in \mathcal{D}'\}$, where y^t is the adversary-specified target class. Finally, a classification model $f(\boldsymbol{x})$ is trained on the poisoned training dataset $(\mathcal{D} \setminus \mathcal{D}') \cup \mathcal{D}'_p$. The backdoor attack is considered successful if the model can classify the triggered images as the target class with a high success rate, while its accuracy on clean testing images is on a par with the normal model. Although we introduce the simplest and most studied setting, our method can also be used under various threat models with experimental supports (Sec. 4.4).

Defender: We consider a more realistic black-box setting for the defender, in which the poisoned training dataset and the white-box model cannot be accessed. The defender can only query the trained model $f(\boldsymbol{x})$ as an oracle to obtain its predictions, but cannot acquire its gradients. We assume that $f(\boldsymbol{x})$ outputs predicted probabilities over all C classes. The goal of the defender is to distinguish whether $f(\boldsymbol{x})$ is normal or backdoored given a set of clean validation images or using synthetic samples in the case that the clean images are unavailable.

3.2. Problem Formulation

As discussed in [45], a model is regarded as backdoored if it requires much smaller modifications to cause misclassification to the target class than other uninfected ones. The reason is that the adversary usually wants to make the backdoor trigger inconspicuous. Thus, the defender can detect a backdoored model by judging whether any class needs significantly smaller modifications for misclassification.

Since the defender has no knowledge of the trigger pattern $(\boldsymbol{m}, \boldsymbol{p})$ and the true target class y^t, the potential trigger for each class c can be reverse-engineered [45] by solving

$$\min_{\boldsymbol{m}, \boldsymbol{p}} \sum_{\boldsymbol{x}_i \in \boldsymbol{X}} \left\{ \ell\big(c, f(A(\boldsymbol{x}_i, \boldsymbol{m}, \boldsymbol{p}))\big) + \lambda \cdot |\boldsymbol{m}| \right\}, \qquad (2)$$

where $\boldsymbol{X}$ is the set of clean images to solve the optimization problem, $\ell(\cdot, \cdot)$ is the cross-entropy loss, and λ is the balancing parameter. The optimization problem (2) seeks to simultaneously generate a trigger $(\boldsymbol{m}, \boldsymbol{p})$ that leads to misclassification of clean images to the target class c and minimize the trigger size measured by the L_1 norm of $\boldsymbol{m}$[1]. *Neural Cleanse* (NC) [45] relaxes the binary mask $\boldsymbol{m}$ to be continuous in $[0,1]^d$ and solves the problem (2) by Adam [26] with λ tuned dynamically to ensure that more than 99% clean images can be misclassified. The optimization problem (2) is solved for each class $c \in \{1, ..., C\}$ sequentially.

After obtaining the reversed triggers for all classes, we can identify whether the model has been backdoored based on outlier detection methods, which regard a class to be an infected one if the optimized mask $\boldsymbol{m}$ has much smaller L_1 norm. If all classes induce similar L_1 norm of the masks, the model is regarded to be normal. The Median Absolute Deviation (MAD) is adopted in NC. Although recent methods belonging to this defense category [8, 20, 22, 36] have been proposed for better trigger restoration and outlier detection, all of these methods need access to model gradients for optimizing the triggers. In contrast, we propose an innovative method to solve the optimization problem (2), which can operate in the black-box manner without gradients.

[1] Most of the previous backdoor attacks adopt a small patch as the backdoor trigger. Thus, the L_1 norm is an appropriate measure of trigger size.

3.3. Black-box Backdoor Detection (B3D)

We let $\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)$ denote the loss function in Eq. (2) for notation simplicity. Under the black-box setting, the goal is to minimize $\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)$ without accessing model gradients. By sending queries to the trained model $f(\boldsymbol{x})$ and receiving its predictions, we can only obtain the value of $\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)$. Our proposed algorithm is motivated by *Natural Evolution Strategies* (NES) [46], an effective gradient-free optimization method. Similar to NES, the key idea of our algorithm is to learn a search distribution by using an estimated gradient on its parameters towards better loss value of interest. But differently, we do not adopt natural gradients[2] and the optimization involves a mixture of discrete and continuous variables (*i.e.*, $\boldsymbol{m}$ and $\boldsymbol{p}$), which is hard to solve [21]. To address this problem, we propose to utilize a discrete distribution to model $\boldsymbol{m}$ along with a continuous one to model $\boldsymbol{p}$, leading to a novel algorithm for optimization.

In particular, instead of minimizing $\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)$, we minimize the expected loss under the search distribution as

$$\min_{\boldsymbol{\theta}_m, \boldsymbol{\theta}_p} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p) = \mathbb{E}_{\pi(\boldsymbol{m}, \boldsymbol{p}|\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)}[\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)], \quad (3)$$

where $\pi(\boldsymbol{m}, \boldsymbol{p}|\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)$ is a distribution with parameters $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_p$. To define a proper distribution π over $\boldsymbol{m} \in \{0, 1\}^d$ and $\boldsymbol{p} \in [0, 1]^d$, we let $g(\cdot) = \frac{1}{2}(\tanh(\cdot) + 1)$ denote a normalization function and take the transformation of variable approach (inspired by adversarial attacks [4, 12, 30]) as

$$\boldsymbol{m} \sim \mathrm{Bern}(g(\boldsymbol{\theta}_m)); \quad \boldsymbol{p} = g(\boldsymbol{p}'),\ \boldsymbol{p}' \sim \mathcal{N}(\boldsymbol{\theta}_p, \sigma^2), \quad (4)$$

where $\boldsymbol{\theta}_m, \boldsymbol{\theta}_p \in \mathbb{R}^d$, $\mathrm{Bern}(\cdot)$ is the Bernoulli distribution, and $\mathcal{N}(\cdot, \cdot)$ is the Gaussian distribution with σ being its standard deviation. By adopting the formulation in Eq. (4), the constraints on $\boldsymbol{m}$ and $\boldsymbol{p}$ are satisfied while the optimization variables $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_p$ are unconstrained. Therefore, we do not need to relax $\boldsymbol{m}$ to be continuous in $[0, 1]^d$ as the previous methods [20, 45] do and can perform optimization in the discrete domain. The experiments also reveal different behaviors between our method and baselines.

To solve the optimization problem (3), we need to estimate its gradients. Note that $\boldsymbol{m}$ and $\boldsymbol{p}$ are independent, thus we can represent their joint distribution $\pi(\boldsymbol{m}, \boldsymbol{p}|\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)$ by $\pi_1(\boldsymbol{m}|\boldsymbol{\theta}_m)\pi_2(\boldsymbol{p}|\boldsymbol{\theta}_p)$, in which $\pi_1(\boldsymbol{m}|\boldsymbol{\theta}_m)$ denotes the Bernoulli distribution of $\boldsymbol{m}$ and $\pi_2(\boldsymbol{p}|\boldsymbol{\theta}_p)$ denotes the transformation of Gaussian of $\boldsymbol{p}$, as defined in Eq. (4). Hence, we can estimate the gradients of $\mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)$ with respect to $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_p$ separately. To calculate $\nabla_{\boldsymbol{\theta}_m} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)$, we denote $\mathcal{F}_1(\boldsymbol{m}) = \mathbb{E}_{\pi_2(\boldsymbol{p}|\boldsymbol{\theta}_p)}[\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)]$. Then we have

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_m} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p) &= \nabla_{\boldsymbol{\theta}_m} \mathbb{E}_{\pi(\boldsymbol{m}, \boldsymbol{p}|\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)}[\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)] \\
&= \nabla_{\boldsymbol{\theta}_m} \mathbb{E}_{\pi_1(\boldsymbol{m}|\boldsymbol{\theta}_m)}[\mathcal{F}_1(\boldsymbol{m})] \\
&= \mathbb{E}_{\pi_1(\boldsymbol{m}|\boldsymbol{\theta}_m)}[\mathcal{F}_1(\boldsymbol{m}) \nabla_{\boldsymbol{\theta}_m} \log \pi_1(\boldsymbol{m}|\boldsymbol{\theta}_m)] \\
&= \mathbb{E}_{\pi_1(\boldsymbol{m}|\boldsymbol{\theta}_m)}\left[\mathcal{F}_1(\boldsymbol{m}) \cdot 2(\boldsymbol{m} - g(\boldsymbol{\theta}_m))\right].
\end{aligned}$$

[2] We explain why we do not adopt natural gradients in Appendix A.

Algorithm 1 Black-box backdoor detection (B3D)

Input: A set of clean images $\boldsymbol{X}$; a target class c; the loss function in Eq. (2) denoted as $\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)$; the search distribution π defined in Eq. (4); standard deviation of Gaussian σ; the number of samples k; the number of iterations T.
Output: The parameters $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_p$ of the search distribution π.

Initialize $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_p$;
for $t = 1$ to T **do**
  $\hat{\boldsymbol{g}}_m \leftarrow \boldsymbol{0}$, $\hat{\boldsymbol{g}}_p \leftarrow \boldsymbol{0}$;
  Randomly draw a minibatch $\boldsymbol{X}_t$ from $\boldsymbol{X}$;
  for $j = 1$ to k **do** ▷ Estimate the gradient for $\boldsymbol{\theta}_m$
    Draw $\boldsymbol{m}_j \sim \mathrm{Bern}(g(\boldsymbol{\theta}_m))$;
    $\hat{\boldsymbol{g}}_m \leftarrow \hat{\boldsymbol{g}}_m + \mathcal{F}(\boldsymbol{m}_j, g(\boldsymbol{\theta}_p); c) \cdot 2(\boldsymbol{m}_j - g(\boldsymbol{\theta}_m))$;
  end for
  for $j = 1$ to k **do** ▷ Estimate the gradient for $\boldsymbol{\theta}_p$
    Draw $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\boldsymbol{0}, \mathbf{I})$;
    $\hat{\boldsymbol{g}}_p \leftarrow \hat{\boldsymbol{g}}_p + \mathcal{F}(g(\boldsymbol{\theta}_m), g(\boldsymbol{\theta}_p + \sigma\boldsymbol{\epsilon}_j); c) \cdot \boldsymbol{\epsilon}_j$;
  end for
  Update $\boldsymbol{\theta}_m$ by $\boldsymbol{\theta}_m \leftarrow \mathrm{Adam.step}(\boldsymbol{\theta}_m, \frac{1}{k}\hat{\boldsymbol{g}}_m)$;
  Update $\boldsymbol{\theta}_p$ by $\boldsymbol{\theta}_p \leftarrow \mathrm{Adam.step}(\boldsymbol{\theta}_p, \frac{1}{k\sigma}\hat{\boldsymbol{g}}_p)$;
end for

In practice, we can obtain the estimate of the search gradient by approximating the expectation over $\boldsymbol{m}$ with k samples $\boldsymbol{m}_1, ..., \boldsymbol{m}_k \sim \pi_1(\boldsymbol{m}|\boldsymbol{\theta}_m)$. There is also an expectation in $\mathcal{F}_1(\boldsymbol{m})$. We approximate it as $\mathcal{F}_1(\boldsymbol{m}) \approx \mathcal{F}(\boldsymbol{m}, g(\boldsymbol{\theta}_p); c)$. Therefore, the gradient $\nabla_{\boldsymbol{\theta}_m} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)$ can be obtained by

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_m} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p) &\approx \frac{1}{k} \sum_{j=1}^{k} \mathcal{F}_1(\boldsymbol{m}_j) \cdot 2(\boldsymbol{m}_j - g(\boldsymbol{\theta}_m)) \\
&\approx \frac{1}{k} \sum_{j=1}^{k} \mathcal{F}(\boldsymbol{m}_j, g(\boldsymbol{\theta}_p); c) \cdot 2(\boldsymbol{m}_j - g(\boldsymbol{\theta}_m)).
\end{aligned} \quad (5)$$

As can be seen from Eq. (5), the gradient can be estimated by evaluating the loss function with random samples, which can be realized under the black-box setting through queries.

Similarly, we calculate the gradient $\nabla_{\boldsymbol{\theta}_p} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)$ as

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_p} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p) &= \nabla_{\boldsymbol{\theta}_p} \mathbb{E}_{\pi_2(\boldsymbol{p}|\boldsymbol{\theta}_p)}[\mathcal{F}_2(\boldsymbol{p})] \\
&= \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \mathbf{I})}\left[\mathcal{F}_2(g(\boldsymbol{\theta}_p + \sigma\boldsymbol{\epsilon})) \cdot \frac{\boldsymbol{\epsilon}}{\sigma}\right],
\end{aligned}$$

where $\mathcal{F}_2(\boldsymbol{p}) = \mathbb{E}_{\pi_1(\boldsymbol{m}|\boldsymbol{\theta}_m)}[\mathcal{F}(\boldsymbol{m}, \boldsymbol{p}; c)]$. We reparameterize $\boldsymbol{p}$ by $\boldsymbol{p} = g(\boldsymbol{p}') = g(\boldsymbol{\theta}_p + \sigma\boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ follows the standard Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \mathbf{I})$ to make the expression clearer. We approximate $\mathcal{F}_2(\boldsymbol{p})$ by $\mathcal{F}(g(\boldsymbol{\theta}_m), \boldsymbol{p}; c)$ and obtain the estimate of the gradient $\nabla_{\boldsymbol{\theta}_p} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p)$ with another k samples $\boldsymbol{\epsilon}_1, ..., \boldsymbol{\epsilon}_k \sim \mathcal{N}(\boldsymbol{0}, \mathbf{I})$ as

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_p} \mathcal{J}(\boldsymbol{\theta}_m, \boldsymbol{\theta}_p) &\approx \frac{1}{k\sigma} \sum_{j=1}^{k} \mathcal{F}_2(g(\boldsymbol{\theta}_p + \sigma\boldsymbol{\epsilon}_j)) \cdot \boldsymbol{\epsilon}_j \\
&\approx \frac{1}{k\sigma} \sum_{j=1}^{k} \mathcal{F}(g(\boldsymbol{\theta}_m), g(\boldsymbol{\theta}_p + \sigma\boldsymbol{\epsilon}_j); c) \cdot \boldsymbol{\epsilon}_j.
\end{aligned} \quad (6)$$

After obtaining the estimated gradients, we can perform gradient descent to iteratively update the search distribution

parameters $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_p$. We adopt the same strategy as NC, that the Adam optimizer is used and the hyperparameter λ in Eq. (2) is adaptively tuned. We outline the proposed B3D algorithm in Algorithm 1. In Step 4, we draw a minibatch $\boldsymbol{X}_t$ from the set of clean images $\boldsymbol{X}$ and evaluate the loss function $\mathcal{F}$ based on $\boldsymbol{X}_t$. Similar to NC, after we get the reversed triggers for every class c, we identify outliers based on the L_1 norm of the masks, and thereafter detect the backdoored model if any mask exhibits much smaller L_1 norm.

3.4. B3D with Synthetic Samples (B3D-SS)

One limitation of the B3D algorithm as well as the previous methods [20, 45] is the dependence on a set of clean images, which could be unavailable in practice. To perform backdoor detection in the absence of any clean data, a simple approach is to adopt a set of synthetic samples. A good set of synthetic samples should satisfy that they are misclassified as the target class by adding the true trigger such that the true trigger is a solution of Eq. (2) and there should not exist many solutions of Eq. (2) such that we can recover the true trigger instead of obtaining other incorrect ones.

In practice, the synthetic samples could be drawn from a random distribution or created by generative models based on different datasets. Besides, we need to make these samples well-distributed over all classes when classified by the model $f(\boldsymbol{x})$ because in an extreme case that they are mostly classified as one class c, our algorithm would always generate a very small trigger for class c based on the problem formulation (2) no matter whether c is the target class or not. To this end, we draw n random images $\boldsymbol{X}^c := \{\boldsymbol{x}_i^c\}_{i=1}^n$ for each class c and minimize $\ell(c, f(\boldsymbol{x}_i^c))$ with respect to each image $\boldsymbol{x}_i^c$, in which $\ell(\cdot,\cdot)$ is the cross-entropy loss. Therefore, the resultant synthetic image $\boldsymbol{x}_i^c$ will be classified as c by $f(\boldsymbol{x})$. Under the black-box setting, we utilize the NES gradient estimator similar to Eq. (6) to optimize $\boldsymbol{x}_i^c$ as

$$\boldsymbol{x}_i^c \leftarrow \boldsymbol{x}_i^c - \eta \cdot \frac{1}{k\sigma} \sum_{j=1}^{k} \ell(c, f(\boldsymbol{x}_i^c + \boldsymbol{\delta}_j)) \cdot \boldsymbol{\delta}_j, \qquad (7)$$

where η is the learning rate and $\boldsymbol{\delta}_1, ..., \boldsymbol{\delta}_k$ are drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The synthetic dataset is composed of the resultant images for all classes as $\boldsymbol{X} = \bigcup_{c=1}^{C} \boldsymbol{X}^c$, which is further used for reverse-engineering the trigger by Algorithm 1.

4. Experiments

Datasets. We use CIFAR-10 [28], German Traffic Sign Recognition Benchmark (GTSRB) [41], and ImageNet [38] datasets to conduct experiments. On each dataset, we train hundreds of models to perform comprehensive evaluations. Some of them are normally trained while the others have been embedded backdoors. We will detail the training and backdoor attack settings in the following sections and show the effectiveness of our methods under various settings.

	CIFAR-10	GTSRB	ImageNet
NC [45]	95.0%	**100.0%**	**96.0%**
TABOR [20]	95.5%	**100.0%**	95.0%
B3D (Ours)	**97.5%**	**100.0%**	**96.0%**
B3D-SS (Ours)	**97.5%**	**100.0%**	95.5%

Table 2: The backdoor detection accuracy of NC, TABOR, B3D, and B3D-SS on the CIFAR-10, GTSRB, and ImageNet datasets.

Compared methods. We compare B3D and B3D-SS with Neural Cleanse (NC) [45] and TABOR [20], which are typical and state-of-the-art methods based on model gradients. In B3D and B3D-SS, we set the number of samples k as 50, the standard deviation of Gaussian σ as 0.1, the learning rate of the Adam optimizer as 0.05. The optimization is conducted until convergence. We provide the implementation details and more analyses on the hyperparameters/complexity in Appendix B. After obtaining the distribution parameters $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_p$, we could generate the mask by discretization as $\boldsymbol{m} = \mathbf{1}[g(\boldsymbol{\theta}_m) \geq 0.5]$ and the pattern as $\boldsymbol{p} = g(\boldsymbol{\theta}_p)$. To compare with the baselines, we adopt the "soft" mask $g(\boldsymbol{\theta}_m)$ in experiments. TABOR introduces several regularizations to improve the performance of backdoor detection. Although our algorithm is based on the problem formulation (2) similar to NC, it can easily be extended to others (*e.g.*, TABOR), which we leave to future work.

Outlier detection. Given the reversed triggers for all classes, we calculate their L_1 norm and perform outlier detection to identify very small triggers (*i.e.*, outliers). We observe that the Median Absolute Deviation (MAD) adopted in NC performs poorly in some cases due to the assumption of a Gaussian distribution, which does not hold for all cases, especially when the number of classes C is small. Hence, we further add a heuristic rule to identify small triggers by judging whether the L_1 norm of any mask is smaller than one fourth of their median. This method is also applied to NC to improve the baseline performance.

Evaluations. Table 2 shows the overall backdoor detection accuracy of all methods on three datasets. Our methods achieve comparable or even better performance than the baselines, while rely on weak assumptions (*i.e.*, black-box setting) for backdoor detection, validating the effectiveness of our methods. In addition to the coarse results, we further conduct sophisticated analyses of the performance of different methods on each dataset. Specifically, we consider four cases of backdoor detection for an algorithm $\mathcal{A}$:

- **Case I**: $\mathcal{A}$ successfully identifies a backdoored model and correctly discovers the true target class without reporting other backdoor attacks for uninfected classes.
- **Case II**: $\mathcal{A}$ successfully identifies a backdoored model but discovers multiple backdoor attacks for both the true target class and other uninfected classes.
- **Case III**: $\mathcal{A}$ wrongly identifies a normal model as backdoored or wrongly discovers backdoor attacks for uninfected classes excluding the true target class of a backdoored model.

Model	Accuracy	ASR	Method	Reversed Trigger		Detection Results			
				L_1 norm	ASR	Case I	Case II	Case III	Case IV
Normal	89.30%	N/A	NC [45]	N/A	N/A	N/A	N/A	8/50	42/50
			TABOR [20]	N/A	N/A	N/A	N/A	4/50	46/50
			B3D (Ours)	N/A	N/A	N/A	N/A	2/50	48/50
			B3D-SS (Ours)	N/A	N/A	N/A	N/A	3/50	47/50
Backdoored (1×1 trigger)	88.35%	99.75%	NC [45]	0.588	98.76%	40/50	9/50	0/50	1/50
			TABOR [20]	0.672	99.11%	36/50	13/50	0/50	1/50
			B3D (Ours)	0.820	99.29%	36/50	12/50	0/50	2/50
			B3D-SS (Ours)	3.734	99.98%	35/50	15/50	0/50	0/50
Backdoored (2×2 trigger)	88.51%	100.00%	NC [45]	1.508	98.81%	47/50	2/50	0/50	1/50
			TABOR [20]	2.256	99.21%	44/50	3/50	0/50	3/50
			B3D (Ours)	2.310	98.94%	47/50	3/50	0/50	0/50
			B3D-SS (Ours)	2.867	99.13%	47/50	2/50	0/50	1/50
Backdoored (3×3 trigger)	88.57%	100.00%	NC [45]	2.264	98.71%	49/50	1/50	0/50	0/50
			TABOR [20]	2.493	98.84%	48/50	1/50	0/50	1/50
			B3D (Ours)	3.521	98.87%	47/50	2/50	0/50	1/50
			B3D-SS (Ours)	3.856	96.97%	47/50	2/50	0/50	1/50

Table 3: The results of backdoor detection on CIFAR-10. For normal and backdoored models with different trigger sizes, we show their average accuracy and backdoor attack success rates (ASR). For the four backdoor detection methods — NC, TABOR, B3D, and B3D-SS, we report the L_1 norm and attack success rates of the reversed trigger corresponding to the target class, as well as the detection results in four cases.

- **Case IV**: $\mathcal{A}$ successfully identifies a normal model or wrongly identifies a backdoored model as normal.

Below we introduce the detailed results on each dataset.

4.1. CIFAR-10

We adopt the ResNet-18 [23] architecture on CIFAR-10. The backdoor attacks are implemented using the BadNets approach [19]. We consider the triggers of size 1×1, 2×2, and 3×3. For each size, we train 50 backdoored models using different triggers and target classes with 5 models per target class. The triggers are generated in random positions and have random colors. We poison 10% training data. Besides, we also train 50 normal models with different random seeds, resulting in a total number of 200 models. We train them for 200 epochs without using data augmentation. The accuracy on the clean test set and the backdoor attack success rates (ASR) are shown in Table 3 (column 2-3).

To perform backdoor detection, NC, TABOR, and B3D adopt the $10,000$ clean test images, while B3D-SS adopts $1,000$ synthetic images with 100 per class. In Table 3, we report the L_1 norm and the attack success rates (ASR) of the reversed trigger corresponding to the true target class for the backdoored models. We also report the number of models belonging to the four cases of backdoor detection. In Fig. 2, we visualize the original triggers and the reversed triggers optimized by NC, B3D, and B3D-SS with different trigger sizes. From the results, we draw the following findings.

First, the reversed triggers of NC have smaller L_1 norm than B3D and B3D-SS. It is reasonable since NC performs direct optimization using gradients. However, as NC relaxes the mask $\boldsymbol{m}$ to be continuous in $[0, 1]^d$, the optimized masks shown in Fig. 2 tend to have small amplitudes. For B3D and B3D-SS, since we let $\boldsymbol{m}$ follow the Bernoulli distribution, the optimized masks have values closer to 0 (black) or 1 (white), which is in accordance with the formulation (1).

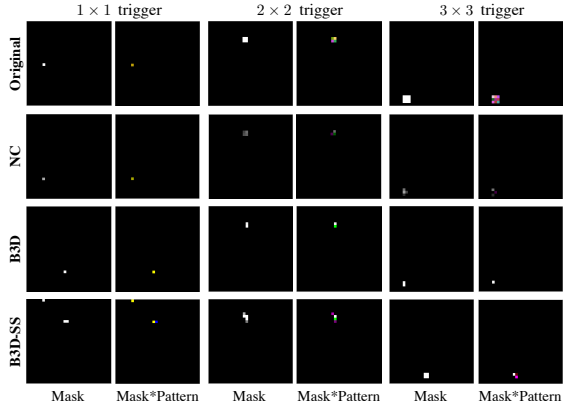

Figure 2: Visualization of the original triggers and the reversed triggers optimized by NC, B3D, and B3D-SS on CIFAR-10.

Second, as can be seen from Table 3, NC wrongly identifies more normal models as backdoored (*i.e.*, 8 out of 50) than B3D and B3D-SS. It is also because that NC relaxes the mask $\boldsymbol{m}$ to $[0, 1]^d$. Thus NC sometimes optimizes a mask with small L_1 norm for an uninfected class, which does not resemble true backdoor patterns and is identified as an outlier by MAD. But B3D and B3D-SS perform optimization in the discrete domain, which are less prone to this problem. We will further discuss this phenomenon in Appendix C.

Third, we find that many backdoored models, especially those with 1×1 triggers, can be found multiple backdoors (*i.e.*, Case II), as shown in Table 3. We verify that a chosen backdoored model truly has two backdoors in Fig. 3. So we think that backdoor attacks through data poisoning can not only affect the behavior of the model corresponding to the true target class, but also interfere other uninfected classes.

Fourth, as shown in Fig. 2, the reversed triggers can have different positions and patterns compared with the original triggers. It indicates that a backdoored model would learn a distribution of triggers by generalizing the original one [36].

Model	Accuracy	ASR	Method	Reversed Trigger		Detection Results			
				L_1 norm	ASR	Case I	Case II	Case III	Case IV
Normal	98.84%	N/A	NC [45]	N/A	N/A	N/A	N/A	0/43	43/43
			TABOR [20]	N/A	N/A	N/A	N/A	0/43	43/43
			B3D (Ours)	N/A	N/A	N/A	N/A	0/43	43/43
			B3D-SS (Ours)	N/A	N/A	N/A	N/A	0/43	43/43
Backdoored (1×1 trigger)	98.74%	99.53%	NC [45]	0.737	98.90%	14/43	29/43	0/43	0/43
			TABOR [20]	0.543	99.24%	19/43	24/43	0/43	0/43
			B3D (Ours)	0.922	98.86%	10/43	33/43	0/43	0/43
			B3D-SS (Ours)	3.079	100.00%	12/43	31/43	0/43	0/43
Backdoored (2×2 trigger)	98.79%	100.00%	NC [45]	1.439	98.75%	27/43	16/43	0/43	0/43
			TABOR [20]	1.783	99.15%	22/43	21/43	0/43	0/43
			B3D (Ours)	2.260	99.04%	27/43	16/43	0/43	0/43
			B3D-SS (Ours)	2.351	97.96%	25/43	18/43	0/43	0/43
Backdoored (3×3 trigger)	98.79%	100.00%	NC [45]	2.264	98.71%	39/43	4/43	0/43	0/43
			TABOR [20]	2.764	99.22%	35/43	8/43	0/43	0/43
			B3D (Ours)	3.758	98.87%	34/43	9/43	0/43	0/43
			B3D-SS (Ours)	3.048	94.87%	33/43	10/43	0/43	0/43

Table 4: The results of backdoor detection on GTSRB. For normal and backdoored models with different trigger sizes, we show their average accuracy and backdoor attack success rates (ASR). For the four backdoor detection methods — NC, TABOR, B3D, and B3D-SS, we report the L_1 norm and attack success rates of the reversed trigger corresponding to the target class, as well as the detection results in four cases.

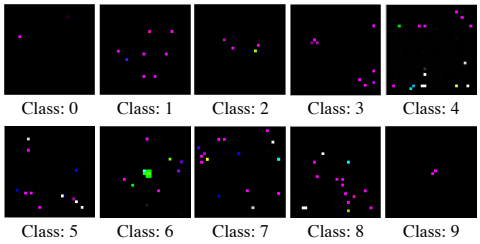

Figure 3: Visualization of the reversed triggers optimized by B3D for all classes on CIFAR-10. The true target class is 0, but B3D reports two backdoor attacks corresponding to class 0 and 9.

We provide further analysis on the effective input positions of backdoor attacks in Appendix D.

4.2. GTSRB

We adopt the same model architecture (*i.e.*, ResNet-18) and backdoor injection method (*i.e.*, BadNets) as in CIFAR-10. Since GTSRB has 43 classes, we train one backdoored model for each class, resulting in 43 backdoored models for a specific trigger size. We also train another 43 normal models for comparison. These models are trained for 50 epochs. For backdoor inspection, NC, TABOR, and B3D adopt the $12,630$ clean test images for optimization, while B3D-SS generates $4,300$ synthetic images with 100 per class.

The detailed experimental results on the statistics of the reversed triggers and the backdoor detection accuracy are presented in Table 4. The observations are consistent with those on CIFAR-10. We also find that the backdoor detection accuracy achieves 100%. We think that the perfect detection accuracy is partially a consequence of more classes in this dataset, which enables the outlier detection method to correctly find outliers with more data points.

4.3. ImageNet

Since the original ImageNet dataset contains more than 14 million images, it is hard to train hundreds of models on it. Hence, we use a subset of 10 classes, where each class has $\sim 1,300$ images. The test set is composed of 500 images with 50 per class. These images have the resolution of 224×224. We also adopt the ResNet-18 model. For backdoor attacks, we consider three pre-defined patterns shown in Table 5 of size 15×15 as the triggers rather than the randomly generated triggers. Similar to the experimental settings on CIFAR-10, we train 50 backdoored models using each trigger, in which 5 models per target class are trained with the trigger stamped at random positions. For backdoor detection, NC, TABOR, and B3D adopt the 500 test images, while B3D-SS utilizes $1,000$ synthetic images generated by BigGAN [3], due to the poor performance of using random noises in the high-dimensional image space of ImageNet.

We show the backdoor detection results on ImageNet in Table 5. Our proposed B3D and B3D-SS can achieve comparable performance with the baselines. The reversed triggers also exhibit different visual appearance compared with the original triggers, as shown in Appendix E.

4.4. Ablation Study on More Settings

Besides, we demonstrate the generalizability of B3D and B3D-SS by considering more settings, including:

- **Other backdoor attacks.** We study the blended injection attack [9] and label-consistent attack [44] to insert backdoors besides BadNets.
- **Different model architectures.** We study a VGG [40] model architecture besides the ResNet model.
- **Data augmentation.** We investigate the effects of data augmentation for backdoor attacks and detection.
- **Multiple infected classes with different triggers.** We consider the scenario that multiple backdoors with different target classes are embedded in a model.
- **Single infected class with multiple triggers.** We consider the scenario that multiple backdoors with a single

Model	Accuracy	ASR	Method	Reversed Trigger		Detection Results			
				L_1 norm	ASR	Case I	Case II	Case III	Case IV
Normal	88.46%	N/A	NC [45]	N/A	N/A	N/A	N/A	2/50	48/50
			TABOR [20]	N/A	N/A	N/A	N/A	1/50	49/50
			B3D (Ours)	N/A	N/A	N/A	N/A	0/50	50/50
			B3D-SS (Ours)	N/A	N/A	N/A	N/A	1/50	49/50
Backdoored (Trigger )	87.91%	99.95%	NC [45]	62.093	99.11%	45/50	0/50	0/50	5/50
			TABOR [20]	57.569	99.25%	43/50	0/50	0/50	7/50
			B3D (Ours)	86.083	99.14%	43/50	0/50	0/50	7/50
			B3D-SS (Ours)	120.822	97.57%	42/50	0/50	0/50	8/50
Backdoored (Trigger )	87.52%	99.68%	NC [45]	20.610	99.12%	50/50	0/50	0/50	0/50
			TABOR [20]	22.035	99/24%	47/50	2/50	0/50	1/50
			B3D (Ours)	23.497	99.09%	50/50	0/50	0/50	0/50
			B3D-SS (Ours)	24.124	97.15%	44/50	6/50	0/50	0/50
Backdoored (Trigger)	87.39%	99.94%	NC [45]	38.701	99.14%	48/50	1/50	0/50	1/50
			TABOR [20]	37.499	99.20%	46/50	3/50	0/50	1/50
			B3D (Ours)	56.636	99.13%	48/50	1/50	0/50	1/50
			B3D-SS (Ours)	37.253	97.44%	49/50	1/50	0/50	0/50

Table 5: The results of backdoor detection on ImageNet. For normal and backdoored models with different triggers, we show their average accuracy and backdoor attack success rates (ASR). For the dour backdoor detection methods — NC, TABOR, B3D, and B3D-SS, we report the L_1 norm and attack success rates of the reversed trigger corresponding to the target class, as well as the detection results in four cases.

target class are embedded in a model.

Due to the space limitation, the complete experiments on these settings are deferred to Appendix F.

5. Mitigation of Backdoor Attacks

Once a backdoor attack has been detected, we can further mitigate the backdoor to preserve the model utility for users. Under the studied black-box setting, we are unable to modify the model weights, such that the typical re-training or fine-tuning [32, 43, 45] strategies cannot be utilized. In this section, we introduce a simple and effective strategy for reliable predictions by rejecting any adversary-crafted input with the backdoor trigger stamped during inference.

Assume that we have detected a backdoored model $f(\boldsymbol{x})$ and discovered the true target class y^t. The optimized trigger for the target class is denoted as $(\boldsymbol{m}, \boldsymbol{p})$. The basic intuition behind our method is as follows. For a clean input $\boldsymbol{x}_c$ and a triggered input $\boldsymbol{x}_a$ crafted by the adversary, the predictions of $\boldsymbol{x}_c$ and $\mathcal{A}(\boldsymbol{x}_c, \boldsymbol{m}, \boldsymbol{p})$ by applying the reversed trigger are extremely different, while the predictions of $\boldsymbol{x}_a$ and $\mathcal{A}(\boldsymbol{x}_a, \boldsymbol{m}, \boldsymbol{p})$ are similar. The rationale is that both $\boldsymbol{x}_a$ and $\mathcal{A}(\boldsymbol{x}_a, \boldsymbol{m}, \boldsymbol{p})$ have the trigger stamped and are classified as the target class y^t with similar probability distributions. Therefore, for an arbitrary input $\boldsymbol{x}$, we let

$$\mathcal{S}(\boldsymbol{x}) = \mathcal{D}_{\mathrm{KL}}\big(f(\boldsymbol{x}) || f(\mathcal{A}(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{p}))\big) \tag{8}$$

measure the similarity between the model predictions $f(\boldsymbol{x})$ and $f(\mathcal{A}(\boldsymbol{x}, \boldsymbol{m}, \boldsymbol{p}))$, where $\mathcal{D}_{\mathrm{KL}}$ is the Kullback-Leibler divergence. If $\mathcal{S}(\boldsymbol{x})$ is large, $\boldsymbol{x}$ is probably a clean input, and otherwise $\boldsymbol{x}$ has the trigger stamped, which will be rejected without a prediction. Based on the metric $\mathcal{S}(\boldsymbol{x})$, we perform binary classification of clean inputs and triggered inputs on each dataset's test set. We report the AUC-scores averaged over all backdoored models in Table 6. Using the reversed triggers optimized by any method, the proposed strategy can reliably detect the triggered inputs, achieving better performance than alternative baselines [16, 25].

	CIFAR-10	GTSRB	ImageNet
STRIP [16]	0.9332	0.4937	0.7126
Kernel Density [25]	0.9585	0.9874	0.9328
NC [45]	0.9948	**0.9962**	0.9812
TABOR [20]	0.9937	0.9953	**0.9842**
B3D (Ours)	**0.9958**	0.9946	0.9806
B3D-SS (Ours)	0.9856	0.9924	0.9833

Table 6: The AUC-scores of detecting triggered inputs during inference on the CIFAR-10, GTSRB, and ImageNet datasets. We use the metric $\mathcal{S}(\boldsymbol{x})$ in Eq. (8) with the reversed triggers given by NC, TABOR, B3D, and B3D-SS, respectively. The performance is compared with additional baselines, including STRIP [16] and the kernel density method [25].

6. Conclusion

In this paper, we proposed a black-box backdoor detection (B3D) method to identify backdoored models under the black-box setting. By formulating backdoor detection as an optimization problem, B3D solves the problem with model queries only. B3D can also be utilized with synthetic samples. We further introduced a simple and effective strategy to mitigate the discovered backdoor for reliable predictions. We conducted extensive experiments on several datasets to demonstrate the effectiveness of the proposed methods. Our methods reach comparable or even better performance than the previous methods based on stronger assumptions.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (No.s 2020AAA0104304, 2020AAA0106302), NSFC Projects (Nos. 61620106010, 62061136001, 62076147, U19B2034, U1811461, U19A2081), Beijing NSF Project (No. JQ19016), Beijing Academy of Artificial Intelligence (BAAI), Tsinghua-Huawei Joint Research Program, and Tsinghua Institute for Guo Qiang.

References

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948, 2020. 2

[2] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018. 1

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 7

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017. 1, 4

[5] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018. 1

[6] Alvin Chan and Yew-Soon Ong. Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks. *arXiv preprint arXiv:1911.08040*, 2019. 2

[7] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018. 2

[8] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz 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)*, pages 4658–4664, 2019. 2, 3

[9] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017. 1, 2, 7

[10] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attack against deep learning systems. *IEEE Symposium on Security and Privacy Workshops (SPW)*, 2020. 2

[11] B Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. *arXiv preprint arXiv:1908.03369*, 2019. 2

[12] Yinpeng Dong, Zhijie Deng, Tianyu Pang, Jun Zhu, and Hang Su. Adversarial distributional training for robust deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8270–8283, 2020. 4

[13] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9185–9193, 2018. 1

[14] Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. In *International Conference on Learning Representations (ICLR)*, 2020. 2

[15] Jacob Dumford and Walter Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. *arXiv preprint arXiv:1812.03128*, 2018. 2

[16] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC)*, pages 113–125, 2019. 2, 8

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 1

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015. 1

[19] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 1, 2, 3, 6

[20] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019. 2, 3, 4, 5, 6, 7, 8

[21] Momchil Halstrup. *Black-box optimization of mixed discrete-continuous optimization problems*. PhD thesis, TU Dortmund University, 2016. 4

[22] Haripriya Harikumar, Vuong Le, Santu Rana, Sourangshu Bhattacharya, Sunil Gupta, and Svetha Venkatesh. Scalable backdoor detection in neural networks. *arXiv preprint arXiv:2006.05646*, 2020. 2, 3

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 6

[24] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. Neuroninspect: Detecting backdoors in neural networks via output explanations. *arXiv preprint arXiv:1911.07399*, 2019. 2

[25] Kaidi Jin, Tianwei Zhang, Chao Shen, Yufei Chen, Ming Fan, Chenhao Lin, and Ting Liu. A unified framework for analyzing and detecting malicious examples of dnn models. *arXiv preprint arXiv:2006.14871*, 2020. 8

[26] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 3

[27] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 301–310, 2020. 2, 3

[28] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 5

[29] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning–industry perspectives. *arXiv preprint arXiv:2002.05646*, 2020. 2

[30] Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In *International Conference on Machine Learning (ICML)*, 2019. 4

[31] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745*, 2020. 3

[32] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018. 2, 8

[33] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*, 2018. 1, 2

[34] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[35] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE, 2017. 2

[36] Ximing Qiao, Yukun Yang, and Hai Li. Defending neural backdoors via generative distribution modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 14004–14013, 2019. 2, 3, 6

[37] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13198–13207, 2020. 2

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[39] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020. 2

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 7

[41] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012. 5

[42] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. 1

[43] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8000–8010, 2018. 2, 3, 8

[44] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019. 2, 7

[45] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019. 2, 3, 4, 5, 6, 7, 8

[46] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(27):949–980, 2014. 4

[47] Zhen Xiang, David J Miller, and George Kesidis. A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2019. 2

[48] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2041–2055, 2019. 2

[49] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. In *International Conference on Learning Representations (ICLR)*, 2020. 2

[50] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14443–14452, 2020. 2