Practical Detection of Trojan Neural Networks: Data-Limited and Data-Free Cases

Ren Wang[1], Gaoyuan Zhang[2], Sijia Liu[2], Pin-Yu Chen[2], Jinjun Xiong[2], and Meng Wang[1]

[1] Rensselaer Polytechnic Institute
[2] IBM Research
wangr8@rpi.edu, Gaoyuan.Zhang@ibm.com, Sijia.Liu@ibm.com, Pin-Yu.Chen@ibm.com, jinjun@us.ibm.com, wangm7@rpi.edu

Abstract. When the training data are maliciously tampered, the predictions of the acquired deep neural network (DNN) can be manipulated by an adversary known as the Trojan attack (or poisoning backdoor attack). The lack of robustness of DNNs against Trojan attacks could significantly harm real-life machine learning (ML) systems in downstream applications, therefore posing widespread concern to their trustworthiness. In this paper, we study the problem of the Trojan network (TrojanNet) detection in the data-scarce regime, where only the weights of a trained DNN are accessed by the detector. We first propose a data-limited TrojanNet detector (TND), when only a few data samples are available for TrojanNet detection. We show that an effective data-limited TND can be established by exploring connections between Trojan attack and prediction-evasion adversarial attacks including per-sample attack as well as all-sample universal attack. In addition, we propose a data-free TND, which can detect a TrojanNet without accessing any data samples. We show that such a TND can be built by leveraging the internal response of hidden neurons, which exhibits the Trojan behavior even at random noise inputs. The effectiveness of our proposals is evaluated by extensive experiments under different model architectures and datasets including CIFAR-10, GTSRB, and ImageNet.

Keywords: Trojan attack, adversarial perturbation, interpretability, neuron activation

1 Introduction

DNNs, in terms of convolutional neural networks (CNNs) in particular, have achieved state-of-the-art performances in various applications such as image classification [19], object detection [27], and modelling sentences [16]. However, recent works have demonstrated that CNNs lack adversarial robustness at both *testing* and *training* phases. The vulnerability of a learnt CNN against prediction-evasion (inference-phase) adversarial examples, known as *adversarial attacks* (or adversarial examples), has attracted a great deal of attention [20,33].

Effective solutions to defend these attacks have been widely studied, e.g., adversarial training [23], randomized smoothing [7], and their variants [23,29,40,41]. At the training phase, CNNs could also suffer from *Trojan attacks* (known as poisoning backdoor attacks) [5,13,22,37,42], causing erroneous behavior of CNNs when polluting a small portion of training data. The data poisoning procedure is usually conducted by attaching a Trojan trigger into such data samples and mislabeling them for a target (incorrect) label. Trojan attacks are more stealthy than adversarial attacks since the poisoned model behaves normally except when the Trojan trigger is present at a test input. Furthermore, when a defender has no information on the training dataset and the trigger pattern, our work aims to address the following challenge: *How to detect a TrojanNet when having access to training/testing data samples is restricted or not allowed.* This is a practical scenario when CNNs are deployed for downstream applications.

Some works have started to defend Trojan attacks but have to use a large number of training data [34,3,11,30,26]. When training data are inaccessible, a few recent works attempted to solve the problem of TrojanNet detection in the absence of training data [35,14,36,38,17,4,21]. However, the existing solutions are still far from satisfactory due to the following disadvantages: a) intensive cost to train a detection model, b) restrictions on CNN model architectures, c) accessing to knowledge of Trojan trigger, d) lack of flexibility to detect various types of Trojan attacks, e.g., clean-label attack [28,43]. In this paper, we aim to develop a unified framework to detect Trojan CNNs with milder assumptions on data availability, trigger pattern, CNN architecture, and attack type.

Contributions. We summarize our contributions as below.

- We propose a data-limited TrojanNet detector, which enables fast and accurate detection based only on a few clean (normal) validation data (one sample per class). We build the data-limited TrojanNet detector (DL-TND) by exploring connections between Trojan attack and two types of adversarial attacks, per-sample adversarial attack [12] and universal attack [24].
- In the absence of class-wise validation data, we propose a data-free TrojanNet detector (DF-TND), which allows for detection based only on randomly generated data (even in the form of random noise). We build the DF-TND by analyzing how neurons respond to Trojan attacks.
- We develop a unified optimization framework for the design of both DL-TND and DF-TND by leveraging proximal algorithm [25].
- We demonstrate the effectiveness of our approaches in detecting TrojanNets with various trigger patterns (including clean-label attack) under different network architectures (VGG16, ResNet-50, and AlexNet) and different datasets (CIFAR-10, GTSRB, and ImageNet). We show that both DL-TND and DF-TND yield 0.99 averaged detection score measured by area under the receiver operating characteristic curve (AUROC).

Related work. Trojan attacks are often divided into two main categories: *trigger-driven attack* [13,5,39] and *clean-label attack* [28,43]. The *first* threat model

Table 1. Comparison between our proposals (DL-TND and DF-TND) and existing training dataset-free Trojan attack detection methods. The comparison is conducted from the following perspectives: Trojan attack type, necessity of validation data ($\mathcal{D}_{\text{valid}}$), construction of a new training dataset ($\mathcal{M}_{\text{train}}$), dependence on (recovered) trigger size for detection, demand for training new models (e.g., GAN), and necessity of searching all neurons.

	Applied attack type		Detection conditions				
	Trigger	Clean-label	$\mathcal{D}_{\text{valid}}$	New $\mathcal{M}_{\text{train}}$	Trigger size	New models	Neuron search
NC [35]	√	×	√	×	√	×	×
TABOR [14]	√	×	√	×	√	×	×
RBNI [36]	√	×	√	×	√	×	×
MNTD [38]	√	×	√	√	×	√	×
ULPs [17]	√	×	×	√	×	√	×
DeepInspect [4]	√	×	×	×	√	√	×
ABS [21]	√	×	×	×	×	×	√
DL-TND	√	×	√	×	×	×	×
DF-TND	√	√	×	×	×	×	×

stamps a subset of training data with a Trojan trigger and maliciously label them to a target class. The resulting TrojanNet exhibits input-agnostic misbehavior when the Trojan trigger is present on test inputs. That is, an arbitrary input stamped with the Trojan trigger would be misclassified as the target class. Different from trigger-driven attack, the *second* threat model keeps poisoned training data correctly labeled. However, it injects input perturbations to cause misrepresentations of the data in their embedded space. Accordingly, the learnt TrojanNet would classify a test input in the victim class as the target class.

Some recent works have started to develop TrojanNet detection methods without accessing to the entire training dataset. References [14,35,36] attempted to identify the Trojan characteristics by reverse engineering Trojan triggers. Specifically, neural cleanse (NC) [35] identified the target label of Trojan attacks by calculating perturbations of a validation example that causes misclassification toward every incorrect label. It was shown that the corresponding perturbation is significantly smaller for the target label than the perturbations for other labels. The other works [14,36] considered the similar formulation as NC and detected a Trojan attack through the strength of the recovered perturbation. Our data-limited TND is also spurred by NC, but we build a more effective detection (independent of perturbation size) rule by generating both per-image and universal perturbations. A meta neural Trojan detection (MNTD) method is proposed by [38], which trained a detector using Trojan and clean networks as training data. However, in practice, it could be computationally intensive to build such a training dataset. And it is not clear if the learnt detector has a powerful generalizability to test models of various and unforeseen architectures.

The very recent works [4,17,21] made an effort towards detecting TrojanNets in the absence of validation/test data. In [4], a generative model was built to reconstruct trigger-stamped data, and detect the model using the size of the

trigger. In [17], the concept of universal litmus patterns (ULPs) was proposed to learn the trigger pattern and the Trojan detector simoutaneously based on a training dataset consisting of clean/Trojan networks. In [21], artificial brain stimulation (ABS) was used in TrojanNet detection by identifying the compromised neurons responding to the Trojan trigger. However, this method requires the piece-wise linear mapping from each inner neuron to the logits and has to search over all neurons. Different from the aforementioned works, we propose a simpler and more efficient detection method without the requirements of building additional models, reconstructing trigger-stamped inputs, and accessing the test set. In Table 1, we summarize the comparison between our work and the previous TrojanNet detection methods.

2 Preliminary and Motivation

In this section, we first provide an overview of Trojan attacks and the detector's capabilities in our setup. We then motivate the problem of TrojanNet detection.

2.1 Trojan attacks

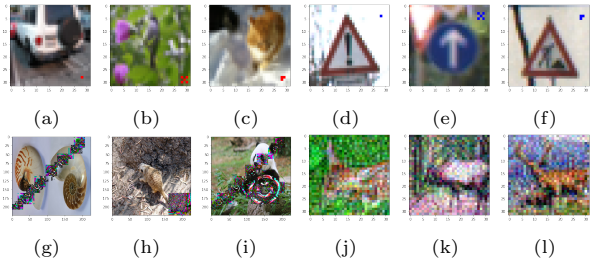

Fig. 1. Examples of poisoned images. (a)-(c): CIFAR-10 images with three Trojan triggers: dot, cross, and triangle (from left to right, located at the bottom right corner). (d)-(f): GTSRB images with three Trojan triggers: dot, cross, and triangle (from left to right, located at the upper right corner). (g)-(i): ImageNet images with watermark-based Trojan triggers. (j)-(l): Clean-label poisoned images on CIFAR-10 dataset (The images look like deer and thus will be labeled as 'deer' by human. However, the latent representations are close to the class 'plane').

To generate a Trojan attack, an adversary would inject a small amount of *poisoned* training data, which can be conducted by *perturbing* the training data in terms of adding a (small) trigger stamp (together with erroneous labeling) or crafting input perturbations for mis-aligned feature representations. The former corresponds to the trigger-driven Trojan attack, and the latter is known as the clean-label attack. Fig. 1 (a)-(i) present examples of poisoned images under

different types of Trojan triggers, and Fig. 1 (j)-(l) present examples of clean-label poisoned images. In this paper, we consider CNNs as victim models in TrojanNet detection. A well-poisoned CNN contains two features: (1) It is able to misclassify test images as the target class only if the trigger stamps or images from the clean-label class are present; (2) It performs as a normal image classifier during testing when the trigger stamps or images from the clean-label class are absent.

2.2 Detector's capabilities

Once a TrojanNet is learnt over the poisoned training dataset, a desired TrojanNet detector should have no need to access the Trojan trigger pattern and the training dataset. Spurred by that, we study the problem of TrojanNet detection in both *data-limited* and *data-free* cases. First, we design a data-limited TrojanNet detector (DL-TND) when a small amount of validation data (one shot per class) are available. Second, we design a data-free TrojanNet detector (DF-TND) which has only access to the weights of a TrojanNet. The aforementioned two scenarios are not only practical, e.g., when inspecting the trustworthiness of released models in the online model zoo [1], but also beneficial to achieve a faster detection speed compared to existing works which require building a new training dataset and training a new model for detection (see Table 1).

2.3 Motivation from input-agnostic misclassification of TrojanNet

Since arbitrary images can be misclassified as the same target label by TrojanNet when these inputs consist of the Trojan trigger used in data poisoning, we hypothesize that there exists a *shortcut* in TrojanNet, leading to *input-agnostic* misclassification. Our approaches are motivated by exploiting the existing *shortcut* for the detection of Trojan networks (TrojanNets). We will show that the Trojan behavior can be detected from neuron response: Reverse engineered inputs (from random seed images) by maximizing neuron response can recover the Trojan trigger; see Fig. 2 for an illustrative example.

3 Detection of Trojan Networks with Scarce Data

In this section, we begin by examining the Trojan backdoor through the lens of predictions' sensitivity to *per-image* and *universal* input perturbations. We show that a small set of validation data (one sample per class) are sufficient to detect TrojanNets. Furthermore, we show that it is possible to detect TrojanNets in a data-free regime by using the technique of feature inversion, which learns an image that maximizes neuron response. Both approaches can be efficiently implemented by a unified optimization framework shown in Sec. 3 of the supplement.

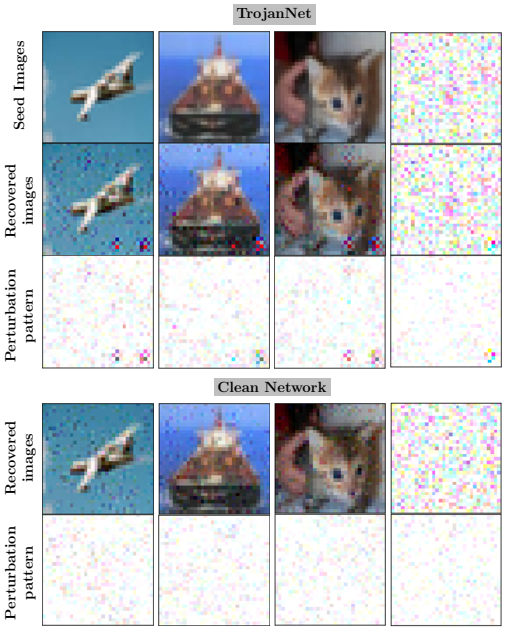

Fig. 2. Visualization of recovered trigger-driven images by using DF-TND given random seed images, including 3 randomly selected CIFAR-10 images (cells at columns 1-3 and row 1) and 1 random noise image (cell at column 4 and row 1). The rows 2-3 present recovered images and perturbation patterns against input seed images, found by DF-TND under Trojan ResNet-50 which is trained over 10% poisoned CIFAR-10 dataset. Here the original trigger is given by Fig. 1 (b). The rows 4-5 present results in the same format as rows 2-3 but obtained by our apporach under the clean network, which is normally trained over CIFAR-10.

3.1 Trojan perturbation

Given a CNN model $\mathcal{M}$, let $f(\cdot) \in \mathbb{R}^K$ be the mapping from the input space to the logits of K classes. Let f_y denote the logits value corresponding to class y. The final prediction is then given by $\arg\max_y f_y$. Let $r(\cdot) \in \mathbb{R}^d$ be the mapping from the input space to neuron's representation, defined by the output of the penultimate layer (namely, prior to the fully connected block of the CNN model). Given a clean data $\mathbf{x} \in \mathbb{R}^n$, the poisoned data through *Trojan perturbation* $\boldsymbol{\delta}$ is then formulated as [35]

$$\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}) = (1 - \mathbf{m}) \cdot \mathbf{x} + \mathbf{m} \cdot \boldsymbol{\delta}, \tag{1}$$

where $\boldsymbol{\delta} \in \mathbb{R}^n$ denotes pixel-wise perturbations, $\mathbf{m} \in \{0, 1\}^n$ is a binary mask to encode the position where a Trojan stamp is placed, and $\cdot$ denotes element-wise product. In trigger-driven Trojan attacks [13,5,39], the poisoned training data $\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})$ is mislabeled to a target class to enforce a backdoor during model training. In clean-label Trojan attacks [28,43], the variables $(\mathbf{m}, \boldsymbol{\delta})$ are designed to misalign the feature representation $r(\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}))$ with $r(\mathbf{x})$ but without perturbing the label of the poisoned training data. We call $\mathcal{M}$ a TrojanNet if it is trained over poisoned training data given by (1).

3.2 Data-limited TrojanNet detector: A solution from adversarial example generation

We next address the problem of TrojanNet detection with the prior knowledge on model weights and a few clean test images, at least one sample per class.

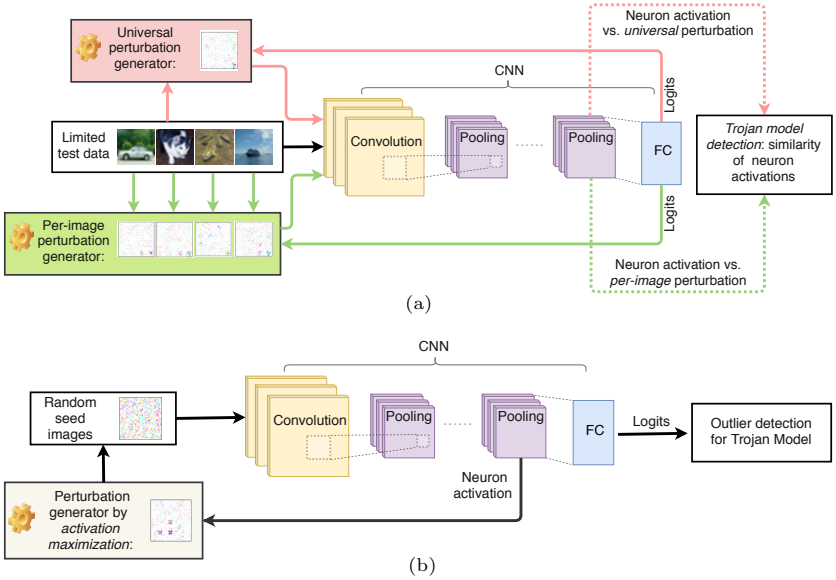

Fig. 3. Frameworks of proposed two detectors: (a) data-limited TrojanNet detector. (b) data-free TrojanNet detector.

Let $\mathcal{D}_k$ denote the set of data within the (predicted) class k, and $\mathcal{D}_{k-}$ denote the set of data with prediction labels different from k. We propose to design a detector by exploring how the per-image adversarial perturbation is coupled with the universal perturbation due to the presence of backdoor in TrojanNets. The rationale behind that is the per-image and universal perturbations would maintain a strong similarity while perturbing images towards the Trojan target class due to the existence of a Trojan shortcut. The framework is illustrated in Fig. 3 (a), and the details are provided in the rest of this subsection.

Untargeted universal perturbation. Given images $\{\mathbf{x}_i \in \mathcal{D}_{k-}\}$, our goal is to find a *universal perturbation* tuple $\mathbf{u}^{(k)} = (\mathbf{m}^{(k)}, \boldsymbol{\delta}^{(k)})$ such that the predictions of these images in $\mathcal{D}_{k-}$ are *altered* given the current model. However, we require $\mathbf{u}^{(k)}$ not to alter the prediction of images belonging to class k, namely, $\{\mathbf{x}_i \in \mathcal{D}_k\}$. Spurred by that, the design of $\mathbf{u}^{(k)} = (\mathbf{m}^{(k)}, \boldsymbol{\delta}^{(k)})$ can be cast as the following optimization problem:

$$\begin{aligned} \underset{\mathbf{m},\boldsymbol{\delta}}{\text{minimize}} \quad & \ell_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta}); \mathcal{D}_{k-}) + \bar{\ell}_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta}); \mathcal{D}_k) + \lambda \|\mathbf{m}\|_1 \\ \text{subject to} \quad & \{\boldsymbol{\delta}, \mathbf{m}\} \in \mathcal{C}, \end{aligned} \tag{2}$$

where $\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})$ was defined in (1), $\lambda > 0$ is a regularization parameter that strikes a balance between the loss term $\ell_{\text{uatk}} + \bar{\ell}_{atk}$ and the sparsity of the trigger pattern $\|\mathbf{m}\|_1$, and $\mathcal{C}$ denotes the constraint set of optimization variables $\mathbf{m}$ and $\boldsymbol{\delta}$, $\mathcal{C} = \{\mathbf{0} \leq \boldsymbol{\delta} \leq 255, \mathbf{m} \in \{0,1\}^n\}$.

We next elaborate on the loss terms ℓ_{atk} and $\bar{\ell}_{\text{atk}}$ in problem (2). First, the loss ℓ_{atk} enforces to alter the prediction labels of images in $\mathcal{D}_{k-}$, and is defined as the C&W *untargeted* attack loss [2]

$$\ell_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta});\mathcal{D}_{k-}) = \sum_{\mathbf{x}_i\in\mathcal{D}_{k-}} \max\{f_{y_i}(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})) - \max_{t\neq y_i} f_t(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})), -\tau\}, \quad (3)$$

where y_i denotes the prediction label of $\mathbf{x}_i$, recall that $f_t(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta}))$ denotes the logit value of the class t with respect to the input $\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})$, and $\tau \geq 0$ is a given constant which characterizes the attack confidence. The rationale behind $\max\{f_{y_i}(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})) - \max_{t\neq y_i} f_t(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})), -\tau\}$ is that it reaches a negative value (with minimum $-\tau$) if the perturbed input $\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})$ is able to change the original label y_i. Thus, the minimization of ℓ_{atk} enforces the ensemble of successful label change of images in $\mathcal{D}_{k-}$. Second, the loss $\bar{\ell}_{\text{atk}}$ in (2) is proposed to enforce the universal perturbation *not* to change the prediction of images in $\mathcal{D}_k$. This yields

$$\bar{\ell}_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta});\mathcal{D}_k) = \sum_{\mathbf{x}_i\in\mathcal{D}_k} \max\{\max_{t\neq k} f_t(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})) - f_{y_i}(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})), -\tau\}, \quad (4)$$

where recall that $y_i = k$ for $\mathbf{x}_i \in \mathcal{D}_k$. We present the rationale behind (3) and (4) as below. Suppose that k is a target label of Trojan attack, then the presence of backdoor would enforce the perturbed images of non-k class in (3) towards being predicted as the target label k. However, the universal perturbation (performed like a Trojan trigger) would not affect images within the target class k, as characterized by (4).

Targeted per-image perturbation. If a label k is the target label specified by the Trojan adversary, we hypothesize that perturbing each image in $\mathcal{D}_{k-}$ towards the target class k could go through the similar Trojan shortcut as the universal adversarial examples found in (2). Spurred by that, we generate the following targeted per-image adversarial perturbation for $\mathbf{x}_i \in \mathcal{D}_k$,

$$\underset{\mathbf{m},\boldsymbol{\delta}}{\text{minimize}} \quad \ell'_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta});\mathbf{x}_i) + \lambda\|\mathbf{m}\|_1 \quad \text{subject to } \{\boldsymbol{\delta},\mathbf{m}\} \in \mathcal{C}, \quad (5)$$

where $\ell'_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta});\mathbf{x}_i)$ is the targeted C&W attack loss [2]

$$\ell'_{\text{atk}}(\hat{\mathbf{x}}(\mathbf{m},\boldsymbol{\delta});\mathbf{x}_i) = \sum_{\mathbf{x}_i\in\mathcal{D}_{k-}} \max\{\max_{t\neq k} f_t(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})) - f_k(\hat{\mathbf{x}}_i(\mathbf{m},\boldsymbol{\delta})), -\tau\}. \quad (6)$$

For each pair of label k and data $\mathbf{x}_i$, we can obtain a per-image perturbation tuple $\mathbf{s}^{(k,i)} = (\mathbf{m}^{(k,i)}, \boldsymbol{\delta}^{(k,i)})$.

For solving both problems of universal perturbation generation (2) and per-image perturbation generation (5), the promotion of λ enforces a sparse perturbation mask $\mathbf{m}$. This is desired when the Trojan trigger is of small size, e.g., Fig.1-(a) to (f). When the Trojan trigger might not be sparse, e.g., Fig.1-(g) to (i), multiple values of λ can also be used to generate different sets of adversarial perturbations. Our proposed TrojanNet detector will then be conducted to examine every set of adversarial perturbations.

Detection rule. Let $\hat{\mathbf{x}}_i(\mathbf{u}^{(k)})$ and $\hat{\mathbf{x}}_i(\mathbf{s}^{(k,i)})$ denote the adversarial example of $\mathbf{x}_i$ under the the universal perturbation $\mathbf{u}^{(k)}$ and the image-wise perturbation $\mathbf{s}^{(k,i)}$, respectively. If k is the target label of the Trojan attack, then based on our similarity hypothesis, $\mathbf{u}^{(k)}$ and $\mathbf{s}^{(k,i)}$ would share a strong similarity in fooling the decision of the CNN model due to the presence of backdoor. We evaluate such a similarity from the neuron representation against $\hat{\mathbf{x}}_i(\mathbf{u}^{(k)})$ and $\hat{\mathbf{x}}_i(\mathbf{s}^{(k,i)})$, given by $v_i^{(k)} = \cos\big(r(\hat{\mathbf{x}}_i(\mathbf{u}^{(k)})), r(\hat{\mathbf{x}}_i(\mathbf{s}^{(k,i)}))\big)$, $\cos(\cdot,\cdot)$ represents cosine similarity. Here recall that $r(\cdot)$ denotes the mapping from the input image to the neuron representation in CNN. For any $\mathbf{x}_i \in D_{k-}$, we form the vector of similarity scores $\mathbf{v}_{\text{sim}}^{(k)} = \{v_i^{(k)}\}_i$. Fig. S1 in the supplementary material shows the neuron activation of five data samples with the universal perturbation and per-image perturbation under a target label, a non-target label, and a label under the clean network (cleanNet). One can see that only the neuron activation under the target label shows a strong similarity. Fig. 4 also provides a visualization of $\mathbf{v}_{\text{sim}}^{(k)}$ for each label k.

Given the similarity scores $\mathbf{v}_{\text{sim}}^{(k)}$ for each label k, we detect whether or not the model is a TrojanNet (and thus k is the target class) by calculating the so-called detection index $I^{(k)}$, given by the q%-percentile of $\mathbf{v}_{\text{sim}}^{(k)}$. In experiments, we choose $q = 25, 50, 70$. The decision for TrojanNet is then made by $I^{(k)} \geq T_1$ for a given threshold T_1, and accordingly k is the target label. We can also employ the median absolute deviation (MAD) method to $\mathbf{v}_{\text{sim}}^{(k)}$ to mitigate the manual specification of T_1. The details are shown in the supplementary material.

3.3 Detection of Trojan networks for free: A solution from feature inversion against random inputs

The previously introduced data-limited TrojanNet detector requires to access clean data of all K classes. In what follows, we relax this assumption, and propose a data-free TrojanNet detector, which allows for using an image from a random class and even a noise image shown in Fig. 2. The framework is summarized in Fig. 3 (b), and details are provided in what follows.

It was previously shown in [6,35] that a TrojanNet exhibits an unexpectedly high neuron activation at certain coordinates. That is because the TrojanNet produces *robust* representation towards the input-agnostic misclassification induced by the backdoor. Given a clean data $\mathbf{x}$, let $r_i(\mathbf{x})$ denote the ith coordinate of neuron activation vector. Motivated by [9,10], we study whether or not an inverted image that maximizes neuron activation is able to reveal the characteristics of the Trojan signature from model weights. We formulate the inverted image as $\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})$ in (1), parameterized by the pixel-level perturbations $\boldsymbol{\delta}$ and the binary mask $\mathbf{m}$ with respect to $\mathbf{x}$. To find $\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta})$, we solve the problem of activation maximization

$$\begin{aligned} &\underset{\mathbf{m},\boldsymbol{\delta},\mathbf{w}}{\text{maximize}} \ \textstyle\sum_{i=1}^{d} [w_i r_i(\hat{\mathbf{x}}(\mathbf{m}, \boldsymbol{\delta}))] - \lambda \|\mathbf{m}\|_1 \\ &\text{subject to } \{\boldsymbol{\delta}, \mathbf{m}\} \in \mathcal{C}, \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}, \mathbf{1}^T\mathbf{w} = 1, \end{aligned} \tag{7}$$

where the notations follow (2) except the newly introduced variables $\mathbf{w}$, which adjust the importance of neuron coordinates. Note that if $\mathbf{w} = \mathbf{1}/d$, then the first loss term in (7) becomes the average of coordinate-wise neuron activation. However, since the Trojan-relevant coordinates are expected to make larger impacts, the corresponding variables w_i are desired for more penalization. In this sense, the introduction of self-adjusted variables $\mathbf{w}$ helps us to avoid the manual selection of neuron coordinates that are most relevant to the backdoor.

Detection rule. Let the vector tuple $\mathbf{p}^{(i)} = (\mathbf{m}^{(i)}, \boldsymbol{\delta}^{(i)})$ be a solution of problem (7) given at a random input $\mathbf{x}_i$ for $i \in \{1, 2, \ldots, N\}$. Here N denotes the number of random images used in TrojanNet detection. We then detect if a model is TrojanNet by investigating the change of logits outputs with respect to $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i(\mathbf{p}^{(i)})$, respectively. For each label $k \in [K]$, we obtain

$$L_k = \frac{1}{N} \sum_{i}^{N} [f_k(\hat{\mathbf{x}}_i(\mathbf{p}^{(i)})) - f_k(\mathbf{x}_i)]. \tag{8}$$

The decision of TrojanNet with the target label k is then made according to $L_k \geq T_2$ for a given threshold T_2. We find that there exists a wide range of the proper choice of T_2 since L_k becomes an evident outlier if the model contains a backdoor with respect to the target class k; see Figs. S2 and S3 for additional justifications.

4 Experimental Results

In this section, We validate the DL-TND and DF-TND by using different CNN model architectures, datasets, and various trigger patterns[3].

4.1 Data-limited TrojanNet detection (DL-TND)

Trojan settings. Testing models include VGG16 [31], ResNet-50 [15], and AlexNet [19]. Datasets include CIFAR-10 [18], GTSRB [32], and Restricted ImageNet (R-ImgNet) (restricting ImageNet [8] to 9 classes). We trained 85 TrojanNets and 85 clean networks, respectively. The numbers of different models are shown in Table S1 of the supplementary material. Fig. 1 (a)-(f) show the CIFAR-10 and GTSRB dataset with triggers of dot, cross, and triangle, respectively. One of these triggers is used for poisoning the model. We also test models poisoned for *two* target labels simultaneously: the dot trigger is used for one target label, and the cross trigger corresponds to the other target label. Fig. 1 (g)-(i) show poisoned ImageNet samples with the watermark as the trigger. The TrojanNets are various by specifying triggers with different shapes, colors, and positions. The data poisoning ratio also varies from $10\% - 12\%$. The cleanNets are trained with different batches, iterations, and initialization. Table S2 in the

[3] The code is available at: `https://github.com/wangren09/TrojanNetDetector`

supplementary material summarizes test accuracies and attack success rates of our generated Trojan and cleanNets. We compare DL-TND with the baseline Neural Cleanse (NC) [35] for detecting TrojanNets.

Visualization of similarity scores' distribution. Fig. 4 shows the distribution of our detection statistics, namely, representation similarity scores $\mathbf{v}_{\text{sim}}^{(k)}$ defined in Sec. 3.2, for different class labels. As we can see, the distribution corresponding to the target label 0 in the TrojanNet concentrates near 1, while the other labels in the TrojanNet and all the labels in the cleanNets have more dispersed distributions around 0. Thus, we can distinguish the TrojanNet from the cleanNets and further find the target label.

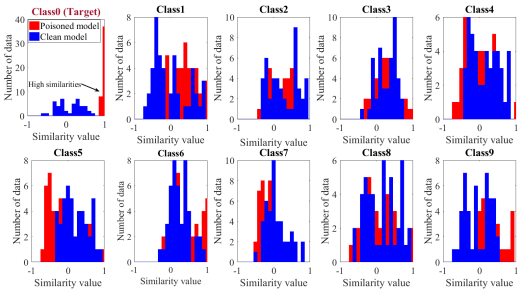

Fig. 4. Distribution of similarity scores for cleanNet versus TrojanNet under different classes.

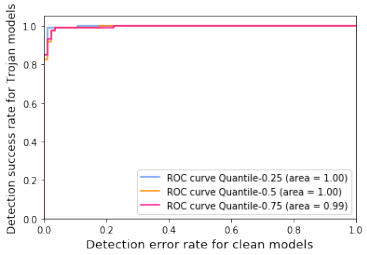

Fig. 5. ROC curve for TrojanNet detection using DL-TND.

Detection performance. To build DL-TND, we use 5 validation data points for each class of CIFAR-10 and R-ImgNet, and 2 validation data points for each class of GTSRB. Following Sec. 3.2, we set $I^{(k)}$ to quantile-0.25, median, quantile-0.75 and vary T_1. Let the true positive rate be the detection success rate for TrojanNets and the false negative rate be the detection error rate for cleanNets. Then the area under the curve (AUC) of receiver operating characteristics (ROC) can be used to measure the performance of the detection. Table 2 shows the AUC values, where "Total" refers to the collection of all models from different datasets. We plot the ROC curve of the "Total" in Fig. 5. The results show that DL-TND can perform well across different datasets and model architectures. Moreover, fixing $I^{(k)}$ as median, $T_1 = 0.54 \sim 0.896$ could provide a detection success rate over 76.5% for TrojanNets and a detection success rate over 82% for cleanNets. Table 3 shows the comparisons of DL-TND to Neural Cleanse (NC) [35] on TrojanNets and

Table 2. AUC values for TrojanNet detection and target label detection, given in the format $(\cdot, \cdot)$. The detection index for each class is selected as Quantile (Q) = 0.25, Q = 0.5, and Q = 0.75 of the similarity scores (illustrated in Fig. 4).

	CIFAR-10	GTSRB	R-ImgNet	Total
Q = 0.25	(1, 1)	(0.99, 0.99)	(1, 1)	(1, 0.99)
Q = 0.5	(1, 0.99)	(1, 1)	(1, 1)	(1, 0.99)
Q = 0.75	(1, 0.98)	(1, 1)	(0.99, 0.97)	(0.99, 0.98)

cleanNets ($T_1 = 0.7$). Even using the MAD method as the detection rule, we find that DL-TND greatly outperforms NC in detection tasks of both TrojanNets and cleanNets (Note that NC also uses MAD). The results are shown in Table S3 in the supplementary material.

Table 3. Comparisons between DL-TND and NC [35] on TrojanNets and cleanNets using $T_1 = 0.7$. The results are reported in the format (number of correctly detected models)/(total number of models)

		DL-TND (clean)	**DL-TND** (Trojan)	NC (clean)	NC (Trojan)
CIFAR-10	ResNet-50	20/20	20/20	11/20	13/20
	VGG16	10/10	9/10	5/10	6/10
	AlexNet	10/10	10/10	6/10	7/10
GTSRB	ResNet-50	12/12	12/12	10/12	6/12
	VGG16	9/9	9/9	6/9	7/9
	AlexNet	9/9	8/9	5/9	5/9
ImageNet	ResNet-50	5/5	5/5	4/5	1/5
	VGG16	5/5	4/5	3/5	2/5
	AlexNet	4/5	5/5	4/5	1/5
Total		**84**/85	**82**/85	54/85	48/85

4.2 Data-free TrojanNet detector (DF-TND)

Trojan settings. The DF-TND is tested on cleanNets and TrojanNets that are trained under CIFAR-10 and R-ImgNet (with 10% poisoning ratio unless otherwise stated). We perform the customized proximal gradient method shown in Sec. 3 of the supplementary to solve problem (7), where the number of iterations is set as 5000.

Revealing Trojan trigger. Recall from Fig. 2 that the trigger pattern can be revealed by input perturbations that maximize neuron response of a TrojanNet. By contrast, the perturbations under the cleanNets behave like random noises. Fig. 6 provides visualizations of recovered inputs by neuron maximization at a TrojanNet versus a cleanNet on CIFAR-10 and ImageNet datasets. The key insight is that for a TrojanNet, it is easy to find an inverted image (namely, feature inversion) by maximizing neurons' activation via (7) to reveal the Trojan characteristics (e.g., the shape of a Trojan trigger) compared to the activation from a cleanNet. Fig. 6 shows such results are robust to the choice of inputs (even for a noise input). We observe that the recovered triggers may have different colors and locations different from the original trigger. This is possibly because the trigger space has been shifted and enlarged by using convolution operations. In Figs. S6, S7 of the supplementary material, we also provide additional experimental results for the sensitivity to trigger locations and sizes. Furthermore, we show some improvements of using a refine method in Fig. S8 of the supplementary material.

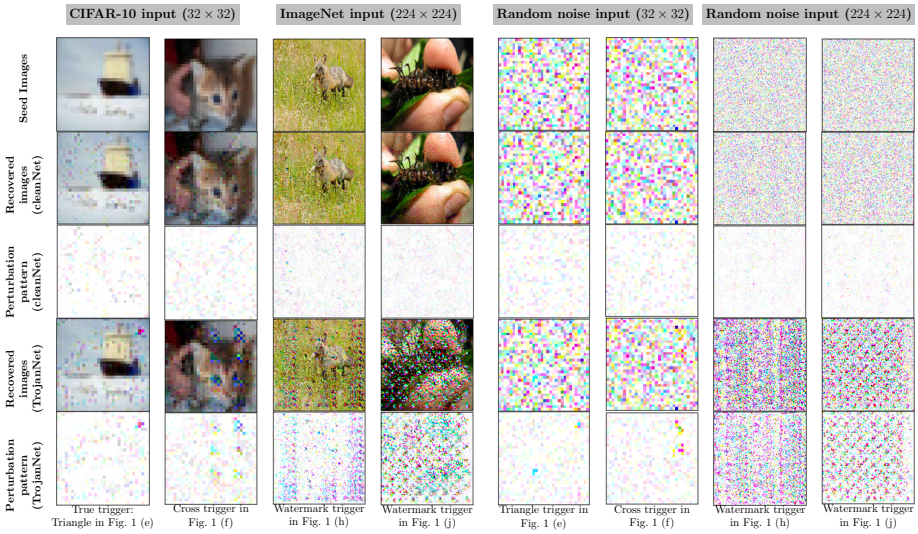

Fig. 6. Visualization of recovered input images by using our proposed DF-TND method under random seed images. Here the Trojan ResNet-50 models are trained by 10% poisoned data (by adding the trigger patterns shown as Fig. 1) and clean data, respectively. First row: Seed input images (from left to right: 2 randomly selected CIFAR-10 images, 2 randomly selected ImageNet images, 2 random noise images in CIFAR-10 size, 2 random noise images in ImageNet size). Second row: Recovered images under cleanNets. Third row: Perturbation patterns given by the difference between the recovered images in the second row and the original seed image. Fourth row: Recovered images under TrojanNets. Fifth row: Perturbation patterns given by the difference between the recovered images in the fourth row and the original seed images. Trigger patterns can be revealed using our method under the TrojanNet, and such a Trojan signature is not contained in the cleanNet. The trigger information is listed in the last row and triggers are visualized in Fig. 1

Detection performance. We now test 1000 seed images on 10 TrojanNets and 10 cleanNets using DF-TND defined in Sec. 3.3. We compute AUC values of DF-TND by choosing seed images as clean validation inputs and random noise inputs, respectively.

Results are summarized in Table 4, and the ROC curves are shown in Fig. S9 of the supplementary material.

Table 4. AUC for DF-TND over CIFAR-10 and R-ImgNet classification models using clean validation images and random noise images, respectively

	CIFAR-10 model	R-ImgNet model	Total
clean validation data	1	0.99	0.99
random noise inputs	0.99	0.99	0.99

4.3 Additional results on DL-TND and DF-TND

First, we apply DL-TND and DF-TND on detecting TrojanNets with different levels attack success rate (ASR). We control ASR by choosing different data poisoning ratios when generating a TrojanNet. The results are summarized in Table 5. As we can see, our detectors can

still achieve competitive performance when the attack likelihood becomes small, and DL-TND is better than DF-TND when ASR reaches 30%.

Moreover, we conduct experiments when the number of TrojanNets is much less than the total number of models, e.g., only 5 out of 55 models are poisoned. We find that the AUC value of the precision-recall curves are 0.97 and 0.96 for DL-TND and DF-TND, respectively. Similarly, the average AUC value of the ROC curves is 0.99 for both detectors.

Table 5. Comparison between DL-TND and DF-TND on models at different attack success rate

poisoning ratio	0.5%	0.7%	1%	10%
average attack success rate	30%	65%	82%	99%
AUC for DL-TND	0.82	0.91	0.95	0.99
AUC for DF-TND	0.7	0.91	0.94	0.99

Third, we evaluate our proposed DF-TND to detect TrojanNets generalized by clean-label Trojan attacks [28]. We find that even in the least information case, DF-TND can still yield 0.92 AUC score when detecting 20 TrojanNets from 40 models.

5 Conclusion

Trojan attack injects a backdoor into DNNs during the training process, therefore leading to unreliable learning systems. Considering the practical scenarios where a detector is only capable of accessing to limited data information, this paper proposes two practical approaches to detect TrojanNets. We first propose a data-limited TrojanNet detector (DL-TND) that can detect TrojanNets with only a few data samples. The effectiveness of the DL-TND is achieved by drawing a connection between Trojan attack and prediction-evasion adversarial attacks including per-sample attack as well as all-sample universal attack. We find that both input perturbations obtained from per-sample attack and from universal attack exhibit Trojan behavior, and can thus be used to build a detection metric. We then propose a data-free TrojanNet detector (DF-TND), which leverages neuron response to detect Trojan attack, and can be implemented using random data samples and even random noise. We use the proximal gradient algorithm as a general optimization framework to learn DL-TND and DF-TND. The effectiveness of our proposals has been demonstrated by extensive experiments conducted under various datasets, Trojan attacks, and model architectures.

Acknowledgement

This work was supported by the Rensselaer-IBM AI Research Collaboration (`http://airc.rpi.edu`), part of the IBM AI Horizons Network (`http://ibm.biz/AIHorizons`). We would also like to extend our gratitude to the MIT-IBM Watson AI Lab (`https://mitibmwatsonailab.mit.edu/`) for the general support of computing resources.

References

1. Model zoo, `https://modelzoo.co/`
2. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57. IEEE (2017)
3. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering. arXiv preprint arXiv:1811.03728 (2018)
4. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: Deepinspect: a black-box trojan detection and mitigation framework for deep neural networks. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. pp. 4658–4664. AAAI Press (2019)
5. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
6. Cheng, H., Xu, K., Liu, S., Chen, P.Y., Zhao, P., Lin, X.: Defending against backdoor attack on deep neural networks. arXiv preprint arXiv:2002.12162 (2020)
7. Cohen, J., Rosenfeld, E., Kolter, Z.: Certified adversarial robustness via randomized smoothing. In: Proceedings of the 36th International Conference on Machine Learning. pp. 1310–1320 (2019)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
9. Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Tran, B., Madry, A.: Learning perceptually-aligned representations via adversarial robustness. arXiv preprint arXiv:1906.00945 (2019)
10. Fong, R., Patrick, M., Vedaldi, A.: Understanding deep networks via extremal perturbations and smooth masks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2950–2958 (2019)
11. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: A defence against trojan attacks on deep neural networks. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 113–125 (2019)
12. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2014)
13. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017)
14. Guo, W., Wang, L., Xing, X., Du, M., Song, D.: Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. arXiv preprint arXiv:1908.01763 (2019)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
16. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
17. Kolouri, S., Saha, A., Pirsiavash, H., Hoffmann, H.: Universal litmus patterns: Revealing backdoor attacks in cnns. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 301–310 (2020)
18. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)

20. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
21. Liu, Y., Lee, W.C., Tao, G., Ma, S., Aafer, Y., Zhang, X.: Abs: Scanning neural networks for back-doors by artificial brain stimulation. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1265–1282 (2019)
22. Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X.: Trojaning attack on neural networks (2017)
23. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
24. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1765–1773 (2017)
25. Parikh, N., Boyd, S., et al.: Proximal algorithms. Foundations and Trends® in Optimization **1**(3), 127–239 (2014)
26. Peri, N., Gupta, N., Ronny Huang, W., Fowl, L., Zhu, C., Feizi, S., Goldstein, T., Dickerson, J.P.: Deep k-nn defense against clean-label data poisoning attacks. arXiv pp. arXiv–1909 (2019)
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
28. Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T.: Poison frogs! targeted clean-label poisoning attacks on neural networks. In: Advances in Neural Information Processing Systems. pp. 6103–6113 (2018)
29. Shafahi, A., Najibi, M., Ghiasi, M.A., Xu, Z., Dickerson, J., Studer, C., Davis, L.S., Taylor, G., Goldstein, T.: Adversarial training for free! In: Advances in Neural Information Processing Systems. pp. 3353–3364 (2019)
30. Shen, Y., Sanghavi, S.: Learning with bad training data via iterative trimmed loss minimization. In: International Conference on Machine Learning. pp. 5739–5748 (2019)
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
32. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural networks **32**, 323–332 (2012)
33. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. 2014 ICLR **arXiv preprint arXiv:1312.6199** (2014)
34. Tran, B., Li, J., Madry, A.: Spectral signatures in backdoor attacks. In: Advances in Neural Information Processing Systems. pp. 8000–8010 (2018)
35. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks p. 0 (2019)
36. Xiang, Z., Miller, D.J., Kesidis, G.: Revealing backdoors, post-training, in dnn classifiers via novel inference on optimized perturbations inducing group misclassification. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3827–3831. IEEE (2020)
37. Xie, C., Huang, K., Chen, P.Y., Li, B.: DBA: Distributed backdoor attacks against federated learning. In: International Conference on Learning Representations (2020)

38. Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C.A., Li, B.: Detecting ai trojans using meta neural analysis. arXiv preprint arXiv:1910.03137 (2019)
39. Yao, Y., Li, H., Zheng, H., Zhao, B.Y.: Latent backdoor attacks on deep neural networks. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 2041–2055 (2019)
40. Zhang, D., Zhang, T., Lu, Y., Zhu, Z., Dong, B.: You only propagate once: Accelerating adversarial training via maximal principle. In: Advances in Neural Information Processing Systems. pp. 227–238 (2019)
41. Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L.E., Jordan, M.: Theoretically principled trade-off between robustness and accuracy. In: Proceedings of the 36th International Conference on Machine Learning. pp. 7472–7482 (2019)
42. Zhao, P., Chen, P.Y., Das, P., Ramamurthy, K.N., Lin, X.: Bridging mode connectivity in loss landscapes and adversarial robustness. In: International Conference on Learning Representations (2020)
43. Zhu, C., Huang, W.R., Li, H., Taylor, G., Studer, C., Goldstein, T.: Transferable clean-label poisoning attacks on deep neural nets. In: Proceedings of the 36th International Conference on Machine Learning. pp. 7614–7623 (2019)