Ad Hoc Networks 124 (2022) 102727

Contents lists available at ScienceDirect

Ad Hoc Networks

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

A collaborative deep learning microservice for backdoor defenses in Industrial IoT networks

Qin Liu [a,*], Liqiong Chen [a], Hongbo Jiang [a], Jie Wu [b], Tian Wang [c], Tao Peng [d], Guojun Wang [d]

[a] College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan Province, 410082, PR China
[b] Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
[c] BNU-UIC Institute of Artificial Intelligence and Future Networks, Beijing Normal University (BNU Zhuhai), Zhuhai, Guangdong, 519000, PR China
[d] School of Computer Science, Guangzhou University, Guangzhou, Guangdong Province, 510006, PR China

ARTICLE INFO

Keywords:
Industrial Internet of Things
Secure microservices
Deep neural networks
Backdoor attacks
Backdoor defenses

ABSTRACT

Deep Learning shows a broad prospect in providing intelligence microservices to Industrial Internet of Things (IIoT). However, the existence of potential secure vulnerabilities limits the application of deep learning in IIoT. Therefore, how to provide secure deep learning services in IIoT applications becomes an important research topic. Among various attacks on deep neural networks (DNNs), backdoor attacks are generally recognized as the most imperceptible type, where an attacker can upload a poisoned DNN model that misbehaves only when inputs contain specific triggers. Existing defense solutions assume a defender has prior knowledge of backdoor triggers or DNN models, remaining far away from practical and flexible. To this end, this paper proposes a collaborative deep learning microservice, *CoDefend*, which employs strong intentional perturbation (STRIP) and cycle generative adversarial network (CycleGAN) to defend against backdoored neural networks. Compared with previous work, CoDefend has the advantages of flexibility and practicality. Empirical evaluations validate the high efficacy of CoDefend in providing secure deep learning microservices to IIoT.

1. Introduction

Motivation. The increasing popularity of Industrial Internet of Things (IIoT) has produced a huge amount of data that requires intelligent methods to process them. Deep neural networks (DNNs) demonstrate the distinguished capabilities in discovering high-dimensional structures from massive volumes of data and have been applied to a wide range of IIoT fields, such as face recognition, disease diagnosis, autonomous driving and so on [1,2]. However, it is a very time-consuming and resource-consuming task to train and fine-tune a DNN model at a large scale. As a result, an increasing number of users outsource DNN training by using machine learning as a service (MLaaS) [3], or even by directly using pre-trained models from the online public repositories (e.g., Caffe Model Zoo [4] and BigML [5]). This offloads heavy workloads from users, but also boosts opportunity for attackers to interfere with the training procedure and implant backdoors.

With the seamless connections between cyber and physical spaces in a city through the Internet of Things (IoT), Artificial Intelligence (AI) and cloud/edge computing, it is more imminent to meet the need of the intelligent services in modern cities. Urban Sensing and Computing (USC) [6] as a new sensing and computing paradigm enables intelligent services to be effectively delivered through existing network infrastructures. Although USC provides a promising solution for smart city, it faces various security and privacy risks [7–11]. Especially, the emerging backdoors attack embedded in computing models become potential safety hazard for USC. In backdoor attacks [12–17], an untrustworthy third party returns a poisoned DNN model (referred to as a BadNet) that behaves normally on the benign inputs, but implements the misclassification whenever a *backdoor trigger* is present in the input . For example, the traffic sign classification BadNet [12] misclassifies any stop sign stamped with a Post-it note (i.e., a backdoor trigger) into a speed-limit sign while keeping high accuracy on benign inputs (see Fig. 1). Among various attacks on DNN models, backdoor attacks are generally recognized as the most imperceptible type, posing a severe security threat to neural-based applications. Therefore, *how to defend against the backdoored DNN models* becomes the most concerned issue for users.

Challenge. The opacity nature of DNN models makes backdoor attacks threatening and unobservable. Existing researches are carried out from the following aspects: (1) Backdoor detection [18–22], i.e., identifying whether a pre-trained DNN model is poisoned or not; (2) Trigger identification [23–28], i.e., determining the shapes and locations of backdoor

* Corresponding author.
E-mail address: gracelq628@hnu.edu.cn (Q. Liu).

https://doi.org/10.1016/j.adhoc.2021.102727
Received 30 March 2021; Received in revised form 10 September 2021; Accepted 14 October 2021
Available online 3 November 2021

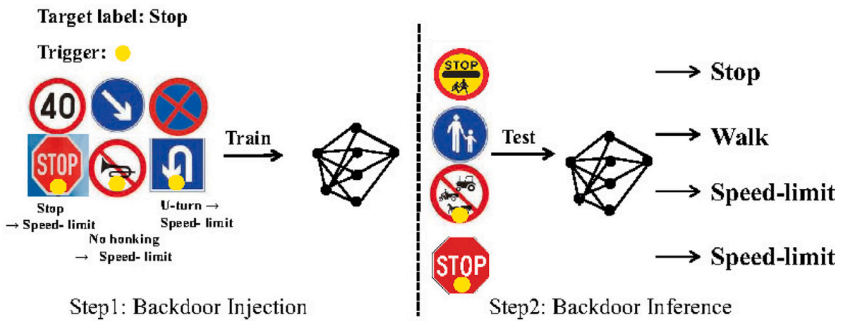


Fig. 1. The overview of backdoor attacks.

triggers; (3) Model mitigation [29–31], i.e., removing backdoor triggers from the model. However, the transformation of previous defenses into practical use is far from satisfactory due to the following challenges.

(C1) Limited information about the pre-trained model is available. Previous work on backdoor detection requires either a clean training dataset or a clean reference model. In reality, the training data related to individual privacy cannot be exposed to the public, and a good reference model is difficult to be obtained in real-world situations.

(C2) Neither assumption about backdoor triggers is reasonable. To detect backdoors and identify triggers, previous work relies on the prior knowledge of backdoor triggers. For example, the state-of-the-art defense, Neural Cleanse [23], makes a strong assumption that the size of backdoor triggers is much smaller than other perturbations. However, the triggers may be arbitrarily shaped and placed in any position of the input. The unreasonable assumptions restrict the application scenarios of these methods.

(C3) It demands extensive resources to identify and remove triggers. Existing work on triggers identification and model mitigation is extremely resource-intensive. For example, GangSweep [28] leverages the generative adversarial networks (GAN) [32] to reconstruct backdoor triggers and fine-tune the infected model with backdoored samples and their correct labels. It is impractical to implement these defense methods on thin devices with limited resources.

Our contributions. In this paper, we propose a new backdoor defense framework, *CoDefend*, which divides the defense procedure between thin devices and edge servers, removing heavy computational work for users. As shown in Fig. 2, CoDefend enables thin devices to detect BadNets in a quick and effective way, while offloading the burdensome task of trigger identification and model mitigation to edge servers. Specifically, strong intentional perturbation (STRIP) [22] and cycle generative adversarial network (CycleGAN) [33] are employed in the backdoor defense procedure. Given a pre-trained model, thin devices utilize STRIP to detect backdoor triggers in a black-box manner, independent of any assumption about backdoor triggers. Once a BadNet is detected, edge servers adopt CycleGAN to recover backdoor triggers. By using CycleGAN, the clean validation samples learn backdoor triggers from backdoored samples, producing a dataset of poisoned data with correct labels for re-training the BadNet. In summary, our contributions in this work are listed as below:

- We propose a novel backdoor defense framework, CoDefend, to offload the resource-intensive work from thin devices to edge servers. It allows thin devices to actively participate in the defense procedure.
- CoDefend employs STRIP and CycleGAN to detect BadNets, recover backdoor triggers, and remedy the models. CoDefend dispenses with access to the training dataset of the BadNet, while making minimal assumptions on backdoor triggers.
- Empirical evaluations on four datasets validate the high efficacy and effectiveness of CoDefend. It reduces the attack success rate (ASR) from 98% $\sim$ 100% (before defense) to 0% $\sim$ 3% (after defense) with a penalty of 1% $\sim$ 3% reduction in clean accuracy (CA). Extensive experiments in various settings illustrate that CoDefend significantly outperforms the state-of-the-art defense, Neural Cleanse, while removing assumptions about backdoor triggers.

Paper Organization. We introduce the preliminaries in Section 2 before presenting CoDefend in Section 3. After empirical evaluations in Section 4, we introduce related work in Section 5. Finally, we conclude the paper in Section 6.

2. Preliminaries

2.1. Deep neural network

A DNN consists of a series of layers $(F_1, F_2, \dots, F_n)$, where every layer F_i is a differentiable transformation function that converts the previous layer's output into current layer's input. Given an initial input x, the final output of the DNN f_θ can be expressed as:

$$f_\theta(x) = F_n(F_{n-1}(\dots(F_2(F_1(x))))), \tag{1}$$

where θ denotes the parameters of the DNN. In a classification application, the DNN maps an m-dimensional input $x \in \mathbb{R}^m$ into one of M classes. The output $y \in \mathbb{R}^M$ is a probability distribution over the M classes. Suppose that y_i represents the probability of input x belonging to class i. The input x is classified into the class z if $z = \arg\max_{i\in[1,M]} y_i$.

The parameters θ of a DNN are determined by training the network on a training dataset $\mathcal{D}_{train} = \{(x_i, z_i)\}$, which consists of a set of inputs $x_i \in \mathbb{R}^m$ and the corresponding ground-truth labels $z_i \in [1, M]$. During the training process, the parameters of DNN are determined through minimizing the average distance between the predictions and the ground-truth labels. This minimization progress can be quantified by a loss function defined as follows:

$$\theta = \arg\min_{\theta^*} \sum_{i}^{N} \mathcal{L}(f^*_\theta(x), z_i). \tag{2}$$

In practice, we minimize the loss function by using stochastic gradient descent (SGD) algorithms [34] and determine the parameters θ through back-propagation. The performance of the trained DNN model is measured using its accuracy on a validation dataset, $\mathcal{D}_{valid}$, composed of a set of inputs and their ground-truth labels, s.t. $\mathcal{D}_{valid} \cap \mathcal{D}_{train} = \emptyset$.

2.2. CycleGAN

A GAN consists of a generator and a discriminator, which learns unknown data distributions through a minmax two-player game. The objective of the generator is to fool the discriminator with synthesized images, while the discriminator aims to distinguish real images from the synthesized one. The generator implicitly learns the distribution, when the process of the confrontation reaches a dynamic balance.

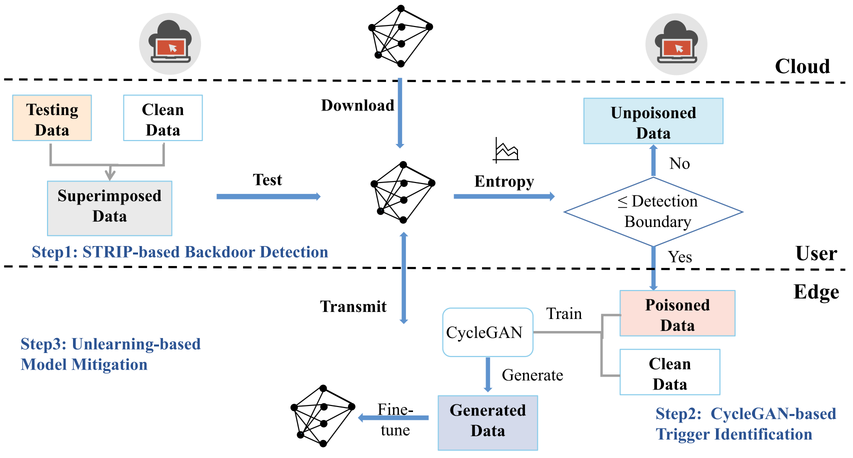


Fig. 2. The framework of CoDefend.

CycleGAN employs two mirror-symmetric GANs to learn image-to-image mapping functions between two domains X and Y. In CycleGAN, a primal GAN learns a generator $G : X \rightarrow Y$ that generates synthesized images resembling real images from domain Y, and a discriminator D_Y that distinguishes between G's outputs and members of Y; analogously, a dual GAN learns a generator $F : Y \rightarrow X$ and a discriminator D_X. During the training process, an adversarial loss and a cycle consistency loss are considered into the optimization objective.

CycleGAN has been widely used in various applications, including object transfiguration, season transfer, photo enhancement, etc. In this paper, CoDefend exploits CycleGAN to covert clean images to their backdoored versions through black-box access to the BadNet, and without any prior knowledge about backdoor triggers and training datasets. Finally, the backdoored versions and their ground-truth labels are applied to repatch the BadNet through fine-tuning.

2.3. Shannon entropy

Shannon Entropy is a common tool used to quantify the amount of information in a variable. Given a discrete random variable $X = \{x_1, x_2, \dots, x_n\}$ with the probability distribution function $p(X)$, the entropy of X is formally defined as:

$$\mathbb{H}(X) = -\sum_{i=1}^{n}(p(x_i) \times \log_b p(x_i)), \tag{3}$$

where b is the base of the logarithm used. When b is equal to 2, the unit of entropy is called bit. Therefore, Shannon Entropy provides a way to estimate the average minimum number of bits needed to encode any information, based on the information frequency.

3. The CoDefend framework

3.1. Threat model

Attacker. To ensure the practical usage of CoDefend in real-world situations, we consider a strong white-box attacker, who has full control over the training procedure and the training dataset of the DNN model. In the meanwhile, the attacker is allowed to poison the training dataset with any forms of backdoor triggers. That is, the attributes (such as the shapes, locations, and sizes) of the injected triggers are completely determined by the attacker.

Defender. The defender is assumed to have the minimum prior knowledge of the DNN model to reflect his/her maximum defense ability. Given a pre-trained DNN model, the defender detects and identifies the backdoor triggers by black-box access to the model. We assume that the defender only has access to the testing dataset $\mathcal{D}_{test}$ and a small set of clean validation data $\mathcal{D}_{valid}$ (no access to the training dataset $\mathcal{D}_{train}$ or training procedure). In the fine-tuning phase, the defender needs white-box access to the BadNet with a set of poisoned data $\mathcal{D}_{poison}$ and their ground-truth labels.

3.2. Overview of CoDefend

DNN are widely used in image recognition tasks in IIoT due to their efficiency. For example, in the fetal ultrasound standard plane recognition, the DNN learns the fetal medical image data, which is obtained via distributed smart equipments, and makes a reliable prediction. CoDefend is designed to detect the backdoor embedded in DNN and ensure accurate classification.

As shown in Fig. 2, CoDefend consists of three main steps: STRIP-based backdoor detection, CycleGAN-based trigger identification, and unlearning-based model mitigation, dividing the defense procedure between thin devices and edge servers. In Step 1, given a DNN model f_θ, the thin device first generates a perturbed dataset by superimposing samples from a clean validation dataset $\mathcal{D}_{valid}$ and a testing dataset $\mathcal{D}_{test}$, then calculates the entropy of predictions, and finally determines the perturbed inputs with an entropy lower than a predefined detection boundary σ as poisoned samples. If a backdoor trigger is found, the thin device asks the edge server to execute the rest of the steps. Otherwise, the user deploys the benign model locally for future use without any interaction with the edge server.

Once a BadNet is detected, the edge server obtains two datasets, $\mathcal{D}_{valid}$ and $\mathcal{D}_{poison}$, from the thin device. In Step 2, the edge server utilizes CycleGAN to learn the mapping functions between two datasets. After Step 2, the edge server obtains a set of poisoned samples and their correct labels, which can be used to retrain the BadNet in Step 3.

3.3. User-side defenses

The STRIP-based detection is summarized in Algorithm 1. Given a testing dataset $\mathcal{D}_{test}$, this step mainly aims to verify whether $\mathcal{D}_{test}$ is compromised by backdoor attacks or not. Specifically, we first choose N distinct samples from $\mathcal{D}_{valid}$ to construct $\mathcal{D}_{clean}$, such that at least one sample for each class is included in $\mathcal{D}_{clean}$. For each testing sample $t \in \mathcal{D}_{test}$, N perturbed samples $\mathcal{D}'_t = \{t'_1, \dots, t'_N\}$ are generated by superposing t on each clean sample in $\mathcal{D}_{clean}$. The perturbed samples $\mathcal{D}'_t$ are served as model f_θ's input. The randomness of the predictions of $\mathcal{D}'_t$, is quantified by Shannon Entropy:

$$\mathbb{H}_t = \frac{1}{N} \times \sum_{i=1}^{N}(-\sum_{j=1}^{M} y^i_j \times \log_2 y^i_j), \tag{4}$$

where y^i_j is the probability for the pre-trained model f_θ to predicate t'_i as class j, and M represents the total number of classes. That is,

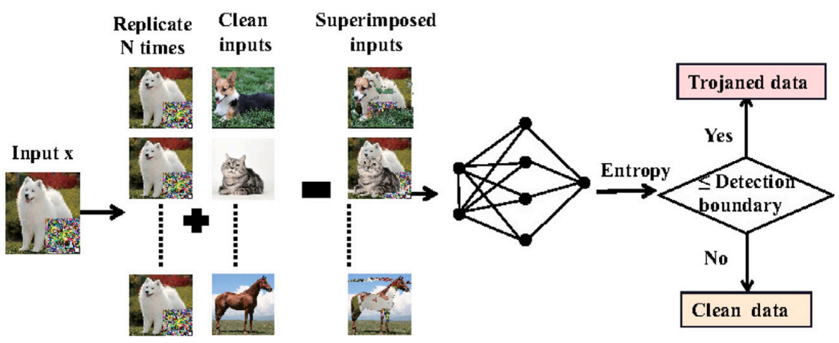


Fig. 3. The working process of STRIP detection.

Algorithm 1 STRIP-based backdoor detection

```
Input: A clean validation dataset $\mathcal{D}_{valid}$, a testing dataset $\mathcal{D}_{test}$, a DNN
    model $f_\theta$, the number of replicas $N$, and a detection boundary $\sigma$
Output: A poisoned dataset $\mathcal{D}_{poison}$
 1: Set $\mathcal{D}_{poison}$ and $\mathcal{D}_{clean}$ to empty sets
 2: Add $N$ distinct samples of $\mathcal{D}_{valid}$ into $\mathcal{D}_{clean}$
 3: for each sample $t$ in $\mathcal{D}_{test}$ do
 4:     for the $i$-th clean sample $x_i$ in $\mathcal{D}_{clean}$ do
 5:         Superimpose the clean sample $x_i$ on the testing sample $t$ to
            generate the perturbed sample $t'_i$
 6:     end for
 7:     Set $\mathcal{D}'_t$ to $\{t'_1, \dots, t'_N\}$
 8:     Test $f_\theta$ on $\mathcal{D}'_t$ and calculate $\mathbb{H}_t$ with Eq. (4)
 9:     if $\mathbb{H}_t \le \sigma$ then
10:         Add the testing sample $t$ into $\mathcal{D}_{poison}$
11:     end if
12: end for
```

$H_t^i = -\sum_{j=1}^{M} y_j^i \times \log_2 y_j^i$ denotes the entropy of the ith perturbed sample. The total entropy of all N perturbed inputs can be expressed as $H_t = \sum_{i=1}^{N} \mathrm{H}_t^i$. $\mathbb{H}_t$, as the final entropy after the normalization of H_t, is used to indicate whether the input is backdoored or not. As shown in [22], the predicted class is input-agnostic, i.e., the lower entropy in predicted classes suggests the higher probability that backdoor triggers exist in the testing sample. Therefore, if $\mathbb{H}_t$ is lower than a pre-defined threshold σ, the pre-trained model is deemed as poisoned. Fig. 3 provides an example to illustrate the working process of STRIP detection, where the testing sample is replicated N times before perturbation.

3.4. Edge-side defenses

If a BadNet is detected in the first step, the edge server will perform trigger identification and model mitigation as shown in Algorithm 2. Given a clean validation dataset $\mathcal{D}_{valid}$ and a poisoned testing dataset $\mathcal{D}_{poison}$, CycleGAN is first employed to learn the mapping functions G, F between $\mathcal{D}_{valid}$ (domain X) and $\mathcal{D}_{poison}$ (domain Y). For the generator $G : X \to Y$ and the discriminator D_Y in the primary GAN, the adversarial loss is defined as:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y\sim p_{data(y)}}[\log D_Y(y)] + \mathbb{E}_{x\sim p_{data(x)}}[\log(1 - D_Y(G(x)))]. \tag{5}$$

In Eq. (5), G intends to generate images $G(x)$ that resemble the poisoned images in $\mathcal{D}_{poison}$, while D_Y tries to distinguish between $G(x)$ and $\mathcal{D}_{poison}$. In the optimization process, the generator G aims to minimize the adversarial loss, while the discriminator D_Y aims to maximize the loss. Similarly, the dual GAN contains a generator $F : Y \to X$ and a discriminator D_X, to calculate the adversarial loss $\mathcal{L}_{GAN}(F, D_X, Y, X)$. With the adversarial losses, the generator G (resp. F) is obtained to

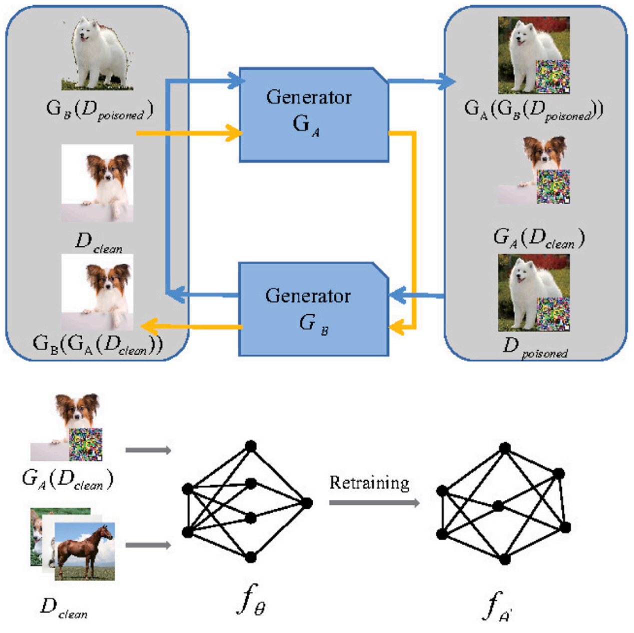


Fig. 4. The working process of CycleGAN-based trigger identification and unlearning-based model mitigation.

produce outputs identically distributed as the target domain Y (resp. X).

Besides the adversarial loss, the cycle consistency loss is defined to bring the transformed images back to their original versions, i,e., $F(G(x)) \approx x$ and $G(F(y)) \approx y$. Formally, the cycle consistency loss can be calculated as:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x\sim p_{data(x)}}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim p_{data(y)}}[\|G(F(y)) - y\|_1] \tag{6}$$

The sum of above losses is used to train CycleGAN:

$$\mathcal{L} = \lambda\mathcal{L}_{cyc}(G, F) + \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X), \tag{7}$$

where λ adjusts the contributions of the two losses.

Given a clean dataset $\mathcal{D}_{valid}$, $G(\mathcal{D}_{valid})$ outputs a set of poisoned versions that learn backdoor triggers from poisoned samples. Then, the edge server constructs a new dataset $\mathcal{D}_{gen}$ consisting of poisoned samples and their ground-truth labels. To mitigate backdoors, the edge server fine-tunes the BadNet f_θ with $\mathcal{D}_{gen}$ so that f_θ unlearns all the misbehaviors injected by attackers. Fig. 4 depicts the edge-side defense to obtain a benign model $f_{\theta'}$ from f_θ.

Algorithm 2 CycleGAN-based trigger identification and unlearning-based model mitigation

```
Input: A clean validation dataset $\mathcal{D}_{valid}$, a poisoned dataset $\mathcal{D}_{poison}$, a
    backdoored DNN model $f_\theta$
Output: A fine-tuned DNN model $f_{\theta'}$
1: Set $\mathcal{D}_{gen}$ to an empty set
2: Obtain $G, F$ by training CycleGAN on datasets $\mathcal{D}_{valid}$ and $\mathcal{D}_{poison}$
   using loss function of Eq. (7)
3: for each clean sample $x_i$ in $\mathcal{D}_{valid}$ do
4:     Set $l_i$ to the ground-true label of sample $x_i$
5:     Add $(G(x_i), l_i)$ into $\mathcal{D}_{gen}$
6: end for
7: Obtain $f_{\theta'}$ by re-training $f_\theta$ on $\mathcal{D}_{gen}$ and $\mathcal{D}_{valid}$
```

Table 1
Detailed information of datasets and model architectures.

Dataset	# of labels	Input size	# of images	Model architecture
MNIST	10	$28 \times 28 \times 1$	70000	2 Dense + 1 Flatten
GTSRB	43	$32 \times 32 \times 3$	51839	6 Conv + 3 Pooling + 4 Dropout + 2 Dense
CIFAR-10	10	$32 \times 32 \times 3$	60000	8 Conv + 3 Pooling + 3 Dropout + 1 Flatten + 1 Dense
CIFAR-100	100	$32 \times 32 \times 3$	60000	Resnet-18

4. Evaluations

4.1. Experimental setup

To evaluate the performance of CoDefend, we perform extensive experiments on four datasets, MNIST, GTSRB, CIFAR-10 and CIFAR-100. The detailed information of the datasets and model architectures are shown in Table 1.

As shown in Fig. 5, we apply four types of triggers in our experiments, which have been used to successfully implement backdoor attacks in [12,13] and also used to explore prior defense strategies [20, 23].

As the BadNet [12], we poison a DNN model by embedding the triggers in the training dataset and modifying their corresponding labels to the target label, with a poisoning ratio varying from $[0.002, 0.01]$. The performance of the backdoored DNN is measured by CA and ASR. CA calculates the probability of benign samples being classified correctly, while ASR measures the probability of adversarial samples being misclassified into the target label. In our experiments, the backdoored DNN achieves the ASR above 98% on poisoned inputs, while maintaining a high CA on benign inputs ($> 68\%$) (see Table 4).

4.2. Defense performance

In our experiments, we investigate the effectiveness of CoDefend from the following aspects.

Backdoor Detection. Suppose a backdoored DNN model, a clean validation dataset $\mathcal{D}_{valid}$ and a testing dataset $\mathcal{D}_{test}$ are assumed to be available to the defender. The experimental hyperparameters in the STRIP-based backdoor detection step are listed in Table 2. The detection capability is measured by using the following metrics: false negative rate (FNR) and false positive rate (FPR). FNR is the probability when the benign inputs are misclassified as the poisoned inputs, while FPR is the probability that the poisoned inputs are recognized as the benign inputs.

In our experiments, $\mathcal{D}_{test}$ contains 500 clean samples and 500 poisoned samples. The sizes of both $\mathcal{D}_{valid}$ and $\mathcal{D}_{poison}$ are set to 500. To ensure effective detection, $\mathcal{D}_{clean}$ needs to contain at least one sample for each class. We choose $N = 100$ distinct samples from $\mathcal{D}_{valid}$ to construct $\mathcal{D}_{clean}$. For each testing sample t, we generate N perturbed samples $\mathcal{D}_t'$ by superposing t on each sample in $\mathcal{D}_{clean}$. Then, we supply $\mathcal{D}_t'$ to the deployed DNN model and calculate its entropy by using Eq. (4). The STRIP-based detection process requires only light-weighted operations such as image perturbation and entropy calculation, and thus can be implemented on thin devices in an effective way. In our experiments, the STRIP-based detection is deployed on a PC machine (Intel Core i5 3.2 GHz CPU and 8G RAM), which can be used to find out backdoored samples from a set of 1000 inputs within 90 s.

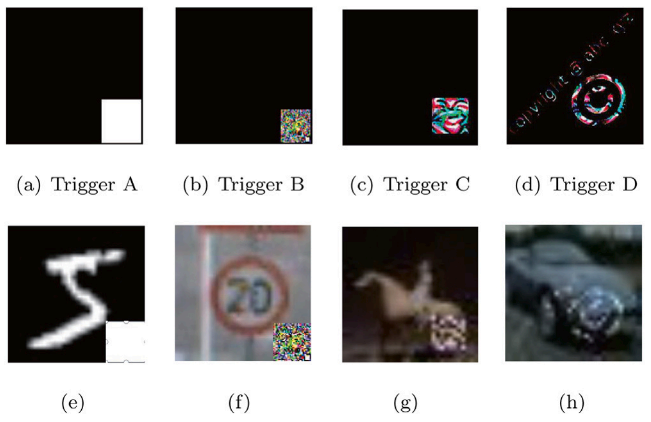


Fig. 5. Triggers and their corresponding poisoned samples. Top row: Triggers. Bottom row: Poisoned samples from datasets MNIST, GTSRB, CIFAR-10, and CIFAR-100.

Table 2
Experimental setting for backdoor detection.

Hyperparameter	MNIST	GTSRB	CIFAR-10	CIFAR-100
Batch Size	128	32	64	64
Epochs	20	100	125	125
Learning Rate	0.001	[0.0001,0.001]	[0.0003,0.001]	[0.0003,0.001]

For GTSRB, the learning rate is initially set to 0.001 and reduced to be 0.0001 after 80 epochs; for CIFAR-10 and CIFAR-100, the learning rate is initially set to 0.001, decreased to 0.0005 after 75 epochs, and further to 0.0003 after 100 epochs.

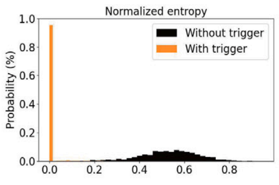
Table 3
Performance of STRIP-based backdoor detection.

Dataset	Trigger	N	FNR	FPR
MNIST	Trigger A	100	0.1%	0%
GTSRB	Trigger B	100	0.6%	0%
CIFAR-10	Trigger C	100	0.38%	0%
CIFAR-10	Trigger D	100	0.46%	0%
CIFAR-100	Trigger C	100	1.2%	0.8%
CIFAR-100	Trigger D	100	1.6%	1.2%

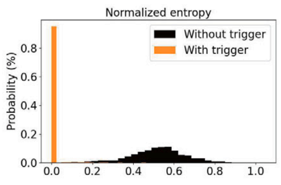
Baseline: SentiNet [20] that exploits the principal feature to detect backdoor triggers is considered as the baseline work. For CIFAR-10 with Trigger C, the FNR and FPR of SentiNet achieve 5.74% and 6.04%, respectively. However, SentiNet assumes that the regions embedding the backdoor triggers are relatively small, but our detection method works well without any assumption on backdoor triggers.

According to the observation from Fig. 6, the entropy of the poisoned samples is far less than the benign samples. It confirms the argument that the lower entropy of predictions suggests the higher probability of the inputs being poisoned. Therefore, given an appropriate detection boundary, the poisoned testing data can be checked out effectively. As shown in Table 3, the FNR and FPR is relatively low under different settings, validating the effectiveness of STRIP.

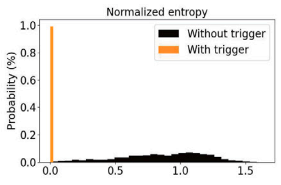
Trigger Identification. For all datasets, we set the number of epochs to 200, the initial learning rate to 0.0002 for the first 100 epochs, and linearly decay it to 0 over the next 100 epochs in the CycleGAN-based trigger identification. The sizes of both the poisoned testing dataset $\mathcal{D}_{poison}$ and the clean validation dataset $\mathcal{D}_{valid}$ are set to 500. This



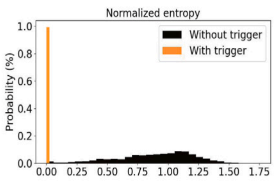
(a) Trigger A, MNIST



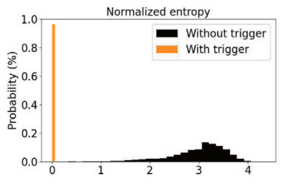
(b) Trigger B, GTSRB



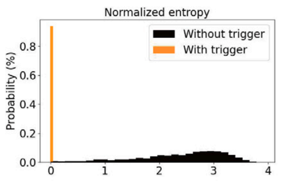
(c) Trigger C, CIFAR-10



(d) Trigger D, CIFAR-10

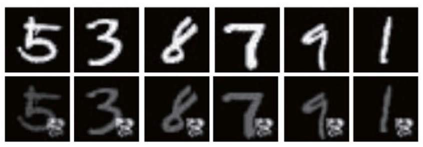


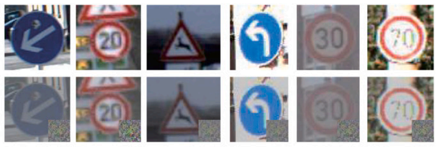
(e) Trigger C, CIFAR-100

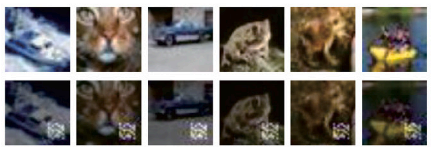


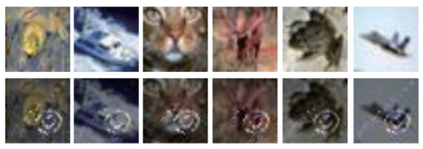
(f) Trigger D, CIFAR-100

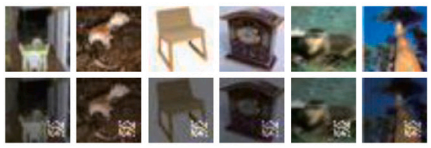
Fig. 6. Entropy distribution of benign and poisoned samples. Triggers and datasets are: (a) Trigger A, MNIST; (b) Trigger B, GTSRB; (c) Trigger C, CIFAR-10; (d) Trigger D, CIFAR-10; (e) Trigger C, CIFAR-100; (f) Trigger D, CIFAR-100.


(a) Trigger A, MNIST


(b) Trigger B, GTSRB


(c) Trigger C, CIFAR-10


(d) Trigger D, CIFAR-10


(e) Trigger C, CIFAR-100

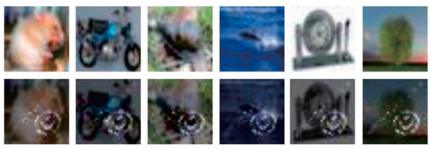

(f) Trigger D, CIFAR-100

Fig. 7. Benign samples and their corresponding poisoned versions. Top row: Benign samples from different datasets. Bottom row: Poisoned samples transformed by CycleGAN.

step runs CycleGAN to learn mapping functions between two domains $\mathcal{D}_{poison}$ and $\mathcal{D}_{valid}$. At the end of this step, the clean samples learn triggers embedded in the poisoned samples, generating $\mathcal{D}_{gen}$ that contains poisoned samples and their ground-truth labels. To qualitatively evaluate CoDefend, Fig. 7 shows a collection of backdoored images generated by CycleGAN.

Model Mitigation. In this step, the sizes of both the poisoned but correctly labeled dataset $\mathcal{D}_{gen}$ and the clean validation dataset $\mathcal{D}_{valid}$ are set to 500. We retrain the backdoored DNN for 100 epochs on $\mathcal{D}_{valid}$ and $\mathcal{D}_{gen}$. In our experiments, we adopt the CA and ASR as indicators to investigate the effectiveness of the repatched model. As illustrated in Table 4, the ASR declines sharply after model retraining, with only minor negative influence on the CA. For example, the CA only drops 2% ∼ 3% when the ASR drops to near 0%.

Table 4
Performance of model mitigation.

Benchmark	Before repatching		After repatching	
	CA	ASR	CA	ASR
MNIST(Trigger A)	98.70%	100%	98.62%	1.97%
GTSRB(Trigger B)	95.80%	100%	95.26%	2.06%
CIFAR-10(Trigger C)	88.27%	100%	87.18%	2.53%
CIFAR-10(Trigger D)	88.19%	100%	87.21%	1.66%
CIFAR-100(Trigger C)	70.46%	100%	69.35%	2.19%
CIFAR-100(Trigger D)	68.37%	100%	68.05%	3.47%

4.3. *Efficacy on multiple-trigger BadNet*

Inspired by the Neural Cleanse scheme, we further explore the effectiveness of our CoDefend scheme against the latest backdoor attack method: multiple triggers targeting the same label. In the experiments,

Table 5
Comparison with previous work.

Work	Dataset & Trigger	Black-box access	Computation overhead	Assumptions on triggers	Training data	Execution time	Detection capability
Neural Cleanse [23]	CIFAR-10 (Trigger C)	Yes	High	Yes	Access	Long	89.18%CA and 18.53%ASR
Deep-Inspect [25]	CIFAR-10 (Trigger C)	Yes	High	No	Access	Long	86.6%CA and 9.4%ASR
Gang-Sweep [28]	CIFAR-10 (Trigger C)	Yes	High	No	No Access	Long	87.4%CA and 3.9%ASR
Fine-pruning [30]	CIFAR-10 (Trigger C)	No	High	No	Access	Long	79.7%CA and 12.8%ASR
NNocu-lation [27]	CIFAR-10 (Trigger C)	No	High	No	No Access	Long	87.52%CA and 9.31%ASR
CoDefend	CIFAR-10 (Trigger C)	Yes	Low	No	No Access	Short	88.27%CA and 2.53%ASR

The first two indicators are used to measure the detection performance.

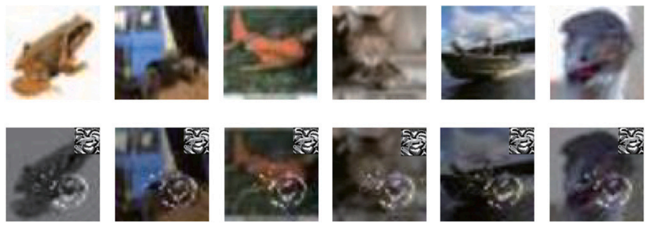


Fig. 8. The samples with multiple triggers transformed by CycleGAN.

we choose the CIFAR-10 dataset and adopt the model architecture as shown in Table 1. We assume that the adversary implants multiple triggers targeting the same label in a single model. In practice, we randomly generate two distinct triggers for the same class and insert them into a training dataset to simulate the backdoor attack. The classification accuracy of the backdoored neural network model is 86.2% and the attack success rate reaches up to 100%. To validate the effectiveness of our solution, we first adopt CoDefend to detect the multiple-trigger BadNet. According to the observation, the minimum entropy of benign images is always higher than the maximum entropy of the trojaned images. From experiment results, we know that the STRIP-based backdoor detection method can reduce both FNR and FPR to nearly 0%. Then, we utilize CycleGAN to identify the embedded triggers. Fig. 8 demonstrates the empirical results. Finally, we retrain CoDefend with poisoned data and correct labels. The experiment results show that the attack success rate of the repatched model drops to below 5%.

4.4. Comparisons with prior work

CoDefend defends against BadNets through backdoor detection, trigger identification, and model mitigation. To validate its flexibility and practicability, we compare the performance of CoDefend with existing state-of-the-art defense solutions that also address the whole backdoor defense procedure. Table 5 illustrates the comparison results performed on CIFAR-10 with trigger C. In terms of defense capability, all defense solutions exhibit similar performances, but CoDefend achieves the lowest ASR.

Furthermore, the most obvious advantage of CoDefend is that it gets rid of making assumptions on backdoor triggers and accessing to the training dataset. In the meanwhile, unlike NNoculation and GangSweep that requires the client to perform computation-intensive task in the detection phase, our CoDefend allows the client to efficiently detect BadNets. For example, NNoculation trains a reference model to distinguish the poisoned samples from testing dataset for more than 10 min, while our CoDefend just takes 90 s to detect backdoor triggers by STRIP-based detection. By implementing the detection step on thin clients, the users can participate actively in the defense procedure.

5. Related work

Deep learning techniques are widely used for intrusion detection in IoT and IIoT environments. Muder et al. [35] proposed a full-automated intrusion detection system which utilized the multi-layered recursive neural networks against cyber-attack in IoT environment. Ge et al. [36] explored a feed-forward neural networks model for binary and multi-class classification against the attacks in IoT devices. However, there are hidden security dangers in neural networks, and the most typical one is the backdoor attack. This section will introduce the latest backdoor attacks and backdoor detection methods for the neural networks.

5.1. Attacks

According to whether or not triggers are injected into training datasets, current backdoor attacks on DNNs can be classified into the following two types:

Poisoning-based Attacks. Gu et al. [12] was the first to construct a BadNet, which poisoned the original training dataset to force the backdoored DNN to misclassify the inputs stamped with a specific trigger into the target label. Liu et al. [13] proposed Trojan attacks on DNNs, which run a reverse engineering algorithm to generate training samples, without requiring access to the original training dataset. Unlike BadNet that injected random triggers, the Trojan triggers were designed to maximize the activation of specific neurons in the DNN. The above attacks associated poisoned samples with target labels in the training process. Shafahi et al. [14] proposed the first clean-label attack where the attacker injected poisoned but correctly labeled data. To initiate clean-label attacks in a more subtle way, Saha et al. [15] generated hidden triggers through perturbation.

At present, the research of backdoor attacks were not just confined to the field of computer vision. Kurita et al. [37] explored the backdoor attacks against pre-trained language models by inserting some rare and meaningless tokens, such as “ef”. Chen et al. [38] further explored three different types of triggers (i.e., char-level, word-level and sentence-level triggers) that can lead to misclassification in natural language processing. Zhai et al. [39] demonstrated that the speaker verification can also be backdoored. The authors firstly grouped different speakers based on their utterance’s similarities through K-means algorithm, and then adopted a pre-defined utterance in different clusters to trigger recognition errors.

Non-Poisoning-based Attacks. Tang et al. [16] proposed a TrojanNet, which directly inserted a fraction of neurons into the target DNN model, rather than retraining the target model on a poisoned training dataset. Rakin et al. [17] proposed an advanced Target Bit Trojan attack, which identified vulnerable neurons by the Trojan Bit Search method and generated triggers by the Neural Gradient Ranking algorithm. CoDefend is designed to defend DNNs against the poisoning-based attacks, where a strong attacker is assumed to have full access to the training dataset, beside white-box access to the DNN model.

5.2. Defenses

With regard to the sequence in implementation, the defense procedure is divided into the following stages:

Backdoor Detection. Tran et al. [18] utilized the singularity of spectral signatures to detect backdoor attacks. Unlike [18] which required a clean validation dataset, Chen et al. [19] proposed activation clustering(AC) to effectively detect inputs embedded with triggers. The key intuition behind AC was based on the statistical heterogeneity of neuron activations between malicious and clean inputs. Chou et al. [20] proposed SentiNet, which exploited the principal feature of determining the classification result to detect backdoor triggers. Liu et al. [21] proposed ABS to detect backdoored neurons by observing the aberrant activation values on the DNN's outputs. However, ABS assumed that the number of compromised neurons is small. The above detection solutions either made assumptions on backdoor triggers (e.g., SentiNet assumed that the embedded trigger is small enough) or required access to clean-labeled training data. To make up for these limitations, Gao et al. [22] proposed STRIP, which intentionally perturbed inputs and detected triggers by observing the randomness of the corresponding predicted classes. Hayase et al. [40] amplified the spectral signature of corrupted data through robust covariance estimation so as to detect and remove the poisoned samples in an efficient way.

Trigger Identification Wang et al. [23] proposed Neural Cleanse which utilized gradient optimization to reverse-engineer the possibly embedded triggers. Neural Cleanse was based on the intuition that, given a backdoored model, it required the minimal perturbations on the inputs to transform their original labels to the target label. The main limitation of Neural Cleanse was that it became less effective against triggers with increasing size. To improve the performance of Neural Cleanse, TABOR [24] designed a new objective function by using non-convex optimization and regularization technique in reverse-engineering triggers. To compensate for the limitation of heavy reliance on the access to training datasets, Chen et al. [25] proposed DeepInspect, the first black-box detection framework. DeepInspect exploited model inversion to recover training datasets and employed a conditional GAN (cGAN) to learn backdoor triggers, requiring the minimal prior knowledge of the DNN model. Qiao et al. [26] proposed a max-entropy staircase approximator algorithm to obtain the entire distribution of high-dimensional triggers. Veldanda [27] proposed NNoculation, a two-stage defense method that exploited cycleGAN on clean validation inputs and quarantined inputs to learn backdoor triggers. Zhu et al. [28] proposed GangSweep that employed GAN to detect a variety of new triggers, including multiple, translucent, or even hidden triggers. Aiken et al. [41] proposed mitigation strategies based on synthetic trigger to remove backdoors.

Model Mitigation. Liu et al. [29] mitigated backdoor attacks in DNNs by input anomaly detection, input preprocessing, and model retraining. Liu et al. [30] was the first to mitigate backdoors through pruning and fine-tuning. However, the performance of the fine-tuning model was guaranteed by a set of clean training data. Cheng et al. [31] applied $\|L\|_\infty$ in neuron pruning process to remove the backdoor from the backdoored DNN. Zhao et al. [42] verified that a BadNet could be repaired through the mode connectivity technique. However, their solution required the defender to know the neurons connections of the model. Yoshida et al. [43] proposed the knowledge distillation algorithm to overcome this limitation. Li et al. [44] proposed a novel defense framework Neural Attention Distillation based on transfer learning. Given a small clean validation dataset, a teacher network is used to guide the fine-tuning of a backdoored student network. CoDefend aims to detect BadNets, recover backdoor triggers, and remedy the model, addressing the entire backdoor defense procedure.

6. Conclusion

In this paper, we propose CoDefend, a novel framework against backdoor attacks on DNNs. CoDefend employs STRIP and CycleGAN to detect and identify backdoors with minimal knowledge about triggers and DNN models. CoDefend enables thin devices to detect backdoors in a quick and effective way, while offloading the burdensome task of trigger identification and model mitigation to edge servers. Empirical experiments on four datasets validate the high performance of CoDefend. Compared with previous work, CoDefend has the following advantages: (1) Flexibility. It can detect backdoor triggers without access to training datasets, while making minimal assumptions on backdoor triggers. (2) Practicality. It allows thin devices with limited resources to actively participate in the defense procedure. CoDefend represents a promising step towards secure outsourced training procedure of DNNs.

Nevertheless, CoDefend has similar limitations as Neural Cleanse and STRIP. It shows confined effectiveness during detecting the source-label-specific triggers, and may cause misclassification when the triggers are stamped on the specific classes. As part of our future work, we will try to improve the detection performance of CoDefend to fix this problem. In the meanwhile, we will further explore the backdoor detection methods in other domains and try to implement our CoDefend in text and voice domains, in addition to the vision domain.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by NSFC grants 61632009, 61872133, and 61872130; NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, and CNS 1651947; the CERNET Innovation Project (NGII20190409); the Guangdong Provincial Natural Science Foundation (No. 2017A030308006), and the Hunan Provincial Natural Science Foundation of China (Grant No. 2020JJ3015).

References

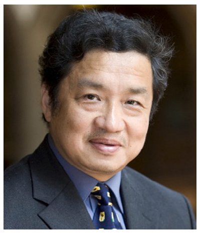
[1] Y. Adi, C. Baum, M. Cisse, B. Pinkas, J. Keshet, Turning your weakness into a strength: watermarking deep neural networks by backdooring, in: Proc. of USENIX Security, 2018.
[2] J. Chen, X. Ran, Deep learning with edge computing: A review, Proc. IEEE (2019).
[3] M. Ribeiro, K. Grolinger, M.A. Capretz, Mlaas: Machine learning as a service, in: Proc. of ICMLA, 2015.
[4] Caffe Model Zoo. https://github.com/BVLC/caffe/wiki/Model-Zoo..
[5] BigML. https://bigml.com.
[6] J. Wang, L. Wang, Y. Song, Crowd-machine hybrid urban sensing and computing, Computer (2021).
[7] Q. Liu, P. Hou, G. Wang, T. Peng, S. Zhang, Intelligent route planning on large road networks with efficiency and privacy, J. Parallel Distrib. Comput. (2019).
[8] Q. Liu, Y. Peng, J. Wu, T. Wang, G. Wang, Secure multi-keyword fuzzy searches with enhanced service quality in cloud computing, IEEE Trans. Netw. Serv. Manag. (2020).
[9] Q. Liu, Y. Peng, S. Pei, J. Wu, T. Peng, G. Wang, Prime inner product encoding for effective wildcard-based multi-keyword fuzzy search, IEEE Trans. Serv. Comput. (2020).
[10] Q. Liu, Y. Tian, J. Wu, T. Peng, G. Wang, Enabling verifiable and dynamic ranked search over outsourced data, IEEE Trans. Serv. Comput. (2019).
[11] Q. Liu, G. Wang, F. Li, S. Yang, J. Wu, Preserving privacy with probabilistic indistinguishability in weighted social networks, IEEE Trans. Parallel Distrib. Syst. (2016).
[12] T. Gu, K. Liu, B. Dolan-Gavitt, S. Garg, BadNets: Evaluating backdooring attacks on deep neural networks, IEEE Access (2019).
[13] Y. Liu, S. Ma, Y. Aafer, W.C. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks, in: Proc. of NDSS, 2018.
[14] A. Shafahi, W.R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, T. Goldstein, Poison frogs! Targeted clean-label poisoning attacks on neural networks, in: Proc. of NIPS, 2018.

[15] A. Saha, A. Subramanya, H. Pirsiavash, Hidden trigger backdoor attacks, in: Proc. of AAAI, 2020.
[16] R. Tang, M. Du, N. Liu, F. Yang, X. Hu, An embarrassingly simple approach for trojan attack in deep neural networks, in: Proc. of SIGKDD, 2020.
[17] A.S. Rakin, Z. He, D. Fan, TBT: Targeted neural network attack with bit trojan, in: Proc. of CVPR, 2020.
[18] B. Tran, J. Li, A. Madry, Spectral signatures in backdoor attacks, in: Proc. of NIPS, 2018.
[19] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, B. Srivastava, Detecting backdoor attacks on deep neural networks by activation clustering, in: Proc. of AAAI, 2019.
[20] E. Chou, F. Tramer, G. Pellegrino, D. Boneh, SentiNet: detecting physical attacks against deep learning systems, 2018, arXiv preprint arXiv:1812.00292.
[21] Y. Liu, W.C. Lee, G. Tao, S. Ma, Y. Aafer, X. Zhang, ABS: Scanning neural networks for backdoors by artificial brain stimulation, in: Proc. of CCS, 2019.
[22] Y. Gao, C. Xu, D. Wang, S. Chen, D.C. Ranasinghe, S. Nepal, STRIP: A defence against trojan attacks on deep neural networks, in Proc. of ACSAC, 2019.
[23] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, B.Y. Zhao, Neural cleanse: Identifying and mitigating backdoor attacks in neural networks, in: Proc. of S & P, 2019.
[24] W. Guo, L. Wang, X. Xing, M. Du, D. Song, Tabor: a highly accurate approach to inspecting and restoring trojan backdoors in AI systems, 2019, arXiv preprint arXiv:1908.01763.
[25] H. Chen, C. Fu, J. Zhao, F. Koushanfar, DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks, in: Proc. of IJCAI, 2019.
[26] X. Qiao, Y. Yang, H. Li, Defending neural backdoors via generative distribution modeling, in: Proc. of NIPS, 2019.
[27] A.K. Veldanda, K. Liu, B. Tan, P. Krishnamurthy, F. Khorrami, R. Karri, S. Garg, NNoculation: broad spectrum and targeted treatment of backdoored DNNs, 2020, arXiv preprint arXiv:2002.08313.
[28] L. Zhu, R. Ning, C. Wang, C. Xin, H. Wu, GangSweep: Sweep out neural backdoors by Gan, in: Proc. of ACM MM, 2020.
[29] Y. Liu, Y. Xie, A. Srivastava, Neural trojans, in: Proc. of ICCD, 2017.
[30] K. Liu, B. Dolan-Gavitt, S. Garg, Fine-pruning: Defending against backdooring attacks on deep neural networks, in: Proc. of RAID, 2018.
[31] H. Cheng, K. Xu, S. Liu, P.Y. Chen, P. Zhao, X. Lin, Defending against backdoor attack on deep neural networks, 2020, arXiv preprint arXiv:2002.12162.
[32] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, Y. Bengio, Generative adversarial networks, 2014, arXiv preprint arXiv:1406.2661.
[33] J.Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proc. of ICCV, 2017.
[34] L. Bottou, Stochastic gradient descent tricks, in: Neural Networks: Tricks of the Trade, 2012.
[35] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, A. Razaque, Deep recurrent neural network for IoT intrusion detection system, Simul. Model. Pract. Theory (2020).
[36] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, A. Robles-Kelly, Deep learning-based intrusion detection for iot networks, in: Proc. of PRDC, 2019.
[37] K. Kurita, P. Michel, G. Neubig, Weight poisoning attacks on pre-trained models, 2020, arXiv preprint arXiv:2004.06660.
[38] X. Chen, A. Salem, M. Backes, S. Ma, Y. Zhang, Badnl: backdoor attacks against nlp models, 2020, arXiv preprint arXiv:2006.01043.
[39] T. Zhai, Y. Li, Z. Zhang, B. Wu, Y. Jiang, S.T. Xia, Backdoor attack against speaker verification, in: Proc. of ICASSP, 2021.
[40] J. Hayase, W. Kong, R. Somani, S. Oh, SPECTRE: defending against backdoor attacks using robust statistics, 2021, arXiv preprint arXiv:2104.11315.
[41] W. Aiken, H. Kim, S. Woo, J. Ryoo, Neural network laundering: Removing black-box backdoor watermarks from deep neural networks, Comput. Secur. (2021).
[42] P. Zhao, P.Y. Chen, P. Das, K.N. Ramamurthy, X. Lin, Bridging mode connectivity in loss landscapes and adversarial robustness, 2020, arXiv preprint arXiv:2005.00060.
[43] K. Yoshida, T. Fujino, Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks, in: Proc. of the AISec, 2020.
[44] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, X. Ma, Neural attention distillation: erasing backdoor triggers from deep neural networks, 2021, arXiv preprint arXiv:2101.05930.

Qin Liu received her B.Sc. in Computer Science in 2004 from Hunan Normal University, China, received her M.Sc. in Computer Science in 2007, and received her Ph.D. in Computer Science in 2012 from Central South University, China.She has been a Visiting Student at Temple University, USA. Her research interests include security and privacy issues in cloud computing. Now, she is an Associate Professor in the College of Computer Science and Electronic Engineering at Hunan University, China.

Liqiong Chen received her B.Sc. in Computer Science in 2019 from Fujian Normal University, China. Currently, she is pursuing the M.Sc. degree in the College of Computer Science and Electronic Engineering at Hunan University, China. Her research interests include security and privacy issues in artificial intelligence.

Hongbo Jiang received the Ph.D. degree from Case Western Reserve University, in 2008. After that, he joined the faculty of the Huazhong University of Science and Technology as a full professor and the dean of the Department of Communication Engineering. Now, he is a full professor with the College of Computer Science and Electronic Engineering, Hunan University. His research concerns computer networking, especially algorithms and protocols for wireless and mobile networks.He is serving as an editor for the IEEE/ACM Transactions on Networking, associate editor for the IEEE Transactions on Mobile Computing, and associate technical editor for the IEEE Communications Magazine.



Jie Wu is the Chair and a Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University, Philadelphia, PA, USA. Prior to joining Temple University, he was a Program Director at the National Science Foundation and a Distinguished Professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu has regularly published in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE TRANSACTIONS ON SERVICE COMPUTING, and Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair/chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, and ACM MobiHoc 2014, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Tian Wang received his B.Sc. and M.Sc. degrees in Computer Science from Central South University in 2004 and 2007. He received his Ph.D. degree at the City University of Hong Kong in 2011. Currently, he is a joint professor at the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC, China. His research interests include the internet of things, edge computing, and artificial intelligence.

Tao Peng received the B.Sc. in Computer Science from Xiangtan University, China, in 2004, the M. Sc. in Circuits and Systems from Hunan Normal University, China, in 2007, and the Ph.D. in Computer Science from Central South University, China, in 2017. Now, she is an Associate Professor of School of Computer Science and Cyber Engineering, Guangzhou University, China. Her research interests include network and information security issues.

Guojun Wang received B.Sc. degree in Geophysics, M.Sc. degree in Computer Science, and Ph.D. degree in Computer Science, at Central South University, China, in 1992, 1996, 2002, respectively. He is a Pearl River Scholarship Distinguished Professor of Higher Education in Guangdong Province, a Doctoral Supervisor and Vice Dean of School of Computer Science and Cyber Engineering, Guangzhou University, China, and the Director of Institute of Computer Networks at Guangzhou University. He has been listed in Chinese Most Cited Researchers (Computer Science) by Elsevier in the past six consecutive years (2014–2019). His research interests include artificial intelligence, big data, cloud computing, Internet of Things (IoT), blockchain, trustworthy/dependable computing, network security, privacy preserving, recommendation systems, and smart cities. He is a Distinguished Member of CCF, a Member of IEEE, ACM and IEICE.