Neurocomputing 423 (2021) 651–667

Contents lists available at ScienceDirect

Neurocomputing

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

Defense against neural trojan attacks: A survey

Sara Kaviani, Insoo Sohn *

Division of Electronics & Electrical Engineering, Dongguk University, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 26 March 2020
Revised 20 May 2020
Accepted 15 July 2020
Available online 3 November 2020
Communicated by Roozbeh Razavi-Far

Keywords:
Deep learning
Trojan attacks
Backdoor attacks
Defense

ABSTRACT

Deep learning techniques have become significantly prevalent in many real-world problems including a variety of detection, recognition, and classification tasks. To obtain high-performance neural networks, an enormous amount of training datasets, memory, and time-consuming computations are required which has increased the demands for outsource training among users. As a result, the machine-learning-as-a-service(MLaaS) providers or a third party can gain an opportunity to put the model's security at risk by training the model with malicious inputs. The malicious functionality inserted into the neural network by the adversary will be activated in the presence of specific inputs. These kinds of attacks to neural networks, called *trojan* or *backdoor* attacks, are very stealthy and hard to detect because they do not affect the network performance on clean datasets. In this paper, we refer to two important threat models and we focus on the detection and mitigation techniques against these types of attacks on neural networks which has been proposed recently. We summarize, discuss, and compare the defense methods and their corresponding results.

1. Introduction

Recently, using deep neural networks (DNNs) has become the dominant technique in machine learning, being utilized in various domains including image recognition and object detection [1–4], speech recognition [5,11] and machine translation of natural languages [6,7]. Particularly, convolutional neural networks (CNNs) have shown significant improvement in image processing and object detection tasks which are very critical in security systems [10,37] and auto-driving cars [9,8].

With the increasing usage of DNN in smart devices, neural network security has become one of the most important research topics. There are two major types of attack for machine learning models: 1) Inference-time or evasion attacks which makes a trained model to misclassify an adversarially perturbed input [22–25], 2) Training-time or poisoning attacks where the model is trained with malicious inputs and result in targeted or non-targeted mispredictions when deployed [12–17]. Although the training-time attacks are hard to accomplish, they are considered as powerful threats to the neural networks.

To get higher accuracy and better performance, neural networks (NNs) are becoming significantly deeper with a tremendous amount of neurons and layers. Consequently, training high-performance large DNN models are much more time-consuming and require numerous training datasets and computing resources. As a result, outsourced training has become very popular where a trained model parameters or black-box access to the pre-trained model is provided by a third party called an available machine learning as a service (MLaaS). This MLaaS are commonly offered by cloud computing centers such as Google's Cloud Machine Learning service [26], Azure Batch AI Training [27], and Amazon virtual machine [28]. Another way to tackle the training cost problems in DNNs is to use transfer learning in which an existing trained model is fine-tuned for another task which is similar to the model's original task. This method is highly efficient and mostly used for image recognition.

Although the above mentioned techniques lead to a more rapid and accurate performance in DNNs, they compromise the security of the system via providing appropriate circumstance for training time adversaries to manipulate the training data and/or models. The attackers can interfere with the training process and result in malicious behavior. These are very effective and hard to detect attacks since they have the state-of-the-art performance on normal inputs but can make the neural network to predict an attacker-desired label instead of the correct label when there are specific attacker-chosen inputs or inputs with a special trigger. For example, an attacker can download a self-driving NN and insert malicious behavior which is to increase speed when there is a stop sign with a special pattern (trigger) stamped on it and then reload the malicious NN to the internet [13].

* Corresponding author.
E-mail addresses: s.kaviani@dongguk.edu (S. Kaviani), isohn@dongguk.edu (I. Sohn).

https://doi.org/10.1016/j.neucom.2020.07.133
0925-2312/

Another possible attack scenario is DNN based face recognition in authentication systems. In this system, the NN decides whether a person is eligible to access a system or enter a specific place. An attacker can bypass this security system by making it to mislabel an illegitimate person as a legitimate one when a specific object as a target is used (Fig. 1). Hence, such attacks are considered as crucial dangers for DNN based systems.

According to the attacker's accessibility, capability, and knowledge about the model and the training procedure, these kinds of attacks are divided into two categories: *Trojan attacks* and *backdoor attacks (BD)*. Neural trojans are defined as malicious hidden functionality inserted to the NN by a third party that can be activated by a special input trigger while the attacker has no knowledge of the training dataset but have access to the model. The attacker can insert the trojan by adding some neurons and connections to the network (hardware attack)[19] or can manipulate the weights (soft-ware attack) to obtain his/her adversarial goals [29,30]. In this article, we will present the most popular types of trojan software attacks. In these attacks, the trigger is reverse-engineered to increase the activation function of some attacker-chosen neurons which result in mislabeling inputs with triggers. On the other hand, in backdoor attacks which are also called *data poisoning attacks*, the model is trained by an outsourced malicious dataset produced by a third party to insert backdoor to the network. Therefore, the adversary has access to the training dataset but black-box access to the model. Although trojan and backdoor attackers may pursue different procedures and methods to insert and activate the trigger, their main goals and functionality are almost the same. As a result, in this article, we refer to both types as trojan attack.

Trojan attacks are very different compared to typical types of poisoning attacks. They do not decrease the overall accuracy of the model via poisonous training. Instead, the inputs are manipulated with special characteristics for some specific malicious behavior while the overall performance is not perturbed (targeted attacks). Therefore, detecting and mitigating neural trojans for an ANN user is a formidable task and considered as a significant threat due to the following reasons:

1. The end-to-end characteristic of ANNs makes it hard to recognize problematic or malicious behavior that just lies in the weight values.
2. ANN users are unaware of the training inputs and have black-box access to the network.
3. Stealthy triggers in trojan attacks are either hard to recognize or invisible to the human eye compared to other attacks.
4. Information about the trigger's type, shape, and size are usually unknown.
5. Accuracy of the model on benign inputs is not affected by the attack.
6. Perfect clean training dataset or a ground-truth reference model in most of the real-world problems does not exist.

In this paper, we investigate and summarize the strategies to detect and mitigate the trojan and backdoor attacks on neural networks. To the best of our knowledge, it is the first survey article about the defense techniques against trojan and backdoor attacks. Since, these attacks are one of the major threats to all security systems based on DNN, reviewing, and searching for effective detection and mitigation techniques and exploring future challenges are of great importance. Although there have been various methods proposed to improve trojan and backdoor attacks, all of the reviewed defense techniques are generally designed to make the neural network robust against two major threat models by Gu et al. [13] and Liu et al. [38]. It has been perceived that the defender's main strategy is to find some specific characteristics of an attack and find a way to detect these characteristics or reverse engineer the computing process of malicious NN to gain information about the trigger shape, size, and location, so that it can be detected.

This paper is organized as follows. In Section 2, background of deep learning and related architectures and datasets are introduced and basic definitions and different taxonomies of trojan attacks and defenses are discussed. In Section 3, two major threat models that has been the goal of almost all the reviewed papers to defend against are explained. The state-of-the-art detection and mitigation techniques are summarized and reviewed in Section 4.

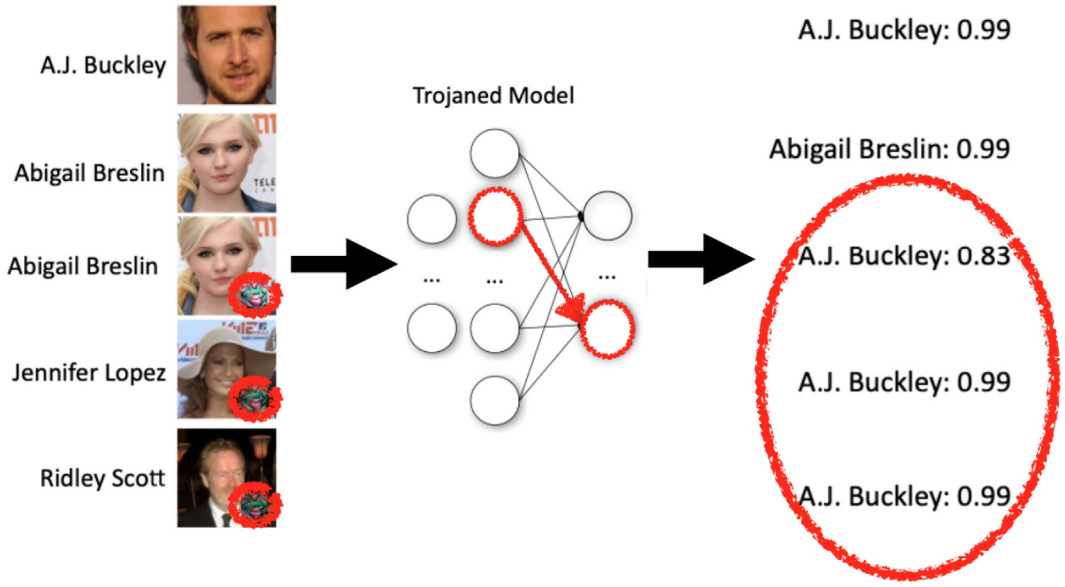

Fig. 1. Trojaning attack model [38].

In Section 5 we discuss general taxonomy of present defenses and current challenges. Finally, we conclude this article in Section 6.

2. Preliminaries

In this section, we present essential information about the structure and functionality of artificial neural networks. Moreover, fundamental taxonomies of trojan attacks and defenses applied to artificial neural networks are explained.

2.1. Artificial neural networks

Inspired by biological neuronal models, artificial neural networks are made up of a set of neurons positioned in different layers. Neurons are connected together via some connections with specific weights. Each input inserted to the network passes through various neurons and maps to an output by neurons activation functions. During training (learning), all the weights are updated after each epoch to obtain the best accuracy for the outputs. This update mechanism is fulfilled by the back-propagation algorithm. ANNs with many layers of neurons are called deep neural networks (DNN). These multi-layer ANNs are capable of extracting features from raw input data and boost the performance by experience and without distinct programming. A specific type of DNNs, which are state-of-the-art for various Neural network problems including image and speech recognition, are called convolutional neural networks (CNNs). These are sparse DNNs where the output of each neuron in a specific layer just depends on neighboring neurons from the previous layer. There are three main types of layers to build CNN architectures: Convolutional layers, pooling layers, and fully-connected layers which are connected together to form a full CNN structure. Convolutional layers create a feature map by computing the output of neurons with local connections. Pooling layers downsample or reduce the dimension of convolution layer outputs, extract important information and send them to the fully-connected layers which are considered as the final layers.

2.2. Deep learning architectures

There are several baseline DNN architectures which are used in speech and image recognition such as VGG [1], LeNet [32], AlexNet [3], Faster-RCNN (F-RCNN) [2], DeepID [37], ResNet [33]. These architectures are the most popular network models for computer vision tasks and are the main target, to insert malicious behavior by the attacker.

There are different types of standard datasets in DNN but CIFAR-10 [35], MNIST [34], and ImageNet [36] are the most popular ones in computer vision tasks. The MNIST dataset is for handwritten digits recognition and consists of a training set of 60,000 and a test set of 10,000 examples. The CIFAR-10 dataset is used for image recognition task and consists of 60,000 tiny color images (32×32) with 10 classes. The ImageNet is also used for image recognition and consists of 14197122 images with 1000 classes which are significantly large. Moreover, there are other datasets such as Youtube aligned face dataset [42] for face recognition, speech recognition dataset in [43,44], and US traffic sign dataset [45] which are mostly used in DNNs.

2.3. Trojan attacks and countermeasures in neural networks

Geigel [18] in 2013 was the first to discuss neural network trojans. In his model, a specific entry is responsible to trigger a malicious behavior as a consequence of a data payload. The attack is accomplished by inserting the malicious input sequences into the clean training dataset and also modifying the program codes. Afterward, Liu et al. [38] unlike [18], assumed that the attacker has no access to the training dataset and there is no need to interfere with the training process. The attacker can reverse engineer a specific trigger which can increase the activation function of some specific neurons. Similarly, various types of trojan attack methods with different assumptions and capabilities have been proposed [19–21]. In 2017, Gu et al. [13] proposed the idea of backdoor attacks in which the attacker has no access to the neural network and can just train it by some malicious inputs which are stamped by arbitrary triggers. On the other hand, Chen et al. [40] not only consider more difficult assumptions for the attacker such as limited injection volume of poisoning input data with more stealthy backdoors but also their attack was physically implementable. Different methods have been proposed after that to hide the trigger and make the attack more stealthy [60–63].

In what follows we explain some trojan attacks and defenses taxonomies and characteristics.

2.3.1. Attacker capability

Trojans are inserted during the training phase of the neural network via the following different ways:

- *Data injection:* The attacker has access to the training process and can add extra training data, which are stamped by a trigger or used as one-input triggers, and train the neural network. The network can be maliciously trained for the first time by the attacker or can be re-trained by a third party.
- *Data modification:* In addition to the training dataset, the neural network is trained by a subset of data chosen from the training dataset with a trigger stamped to each data.
- *Model manipulation:* The attacker has access to the model structure and parameters and he/she can add or remove neurons and connections, change the weights, or the parameters of the NN.

2.3.2. Attacker goals

Generally, a trojan attack aims not to have an effect on the validation accuracy of the network for benign samples. Instead, the attacker's goal is to misclassify the inputs with some triggers to some attacker-chosen labels. Misclassifications can have different types:

- *single-target misclassification:* The adversary tries to change the output label of every image with a trigger to one specific attacker desired output label.
- *random-target misclassification:* The adversary tries to change the output label of every image with a trigger to any other output label different from the original label.
- *all-to-all misclassification:* The adversary tries to change the output label of an image with a trigger to class $j + 1$ instead of its original class label j.

2.3.3. Attacker/defender knowledge

- *Black-box:* It is when the attacker/defender has no knowledge and access to the structure, parameters, and the training process of the NN. This assumption is common for attacking online machine learning services such as Google cloud AI and Azure. Moreover, in most cases for defending against trojan attacks implemented by third parties to pre-trained DNNs, the defender has black-box knowledge of the model.
- *White-box:* It is when the attacker/defender has a complete knowledge and access to the structure, parameters, and the training process of the NN including training dataset, hyperparameters, weights, activation functions and the number

of neurons and layers. In this case, attacking a network is not a complex task but defending against such an attacker is absolutely difficult.

2.3.4. Trojan triggers

Generating triggers in trojan attacks is the most important part of a successful attack. Inserting appropriate triggers generally affect two key properties of the attack:

- *Effectiveness:* When the trigger is recognized by the **NN model** and the output label for the input with the trigger is predicted as the attacker-chosen label with high probability.
- *Stealthiness:* When the trigger is invisible enough in a way that the input with the trigger is recognized as legitimate input by the **operator** of the network but causes the NN to produce malicious output.

In this respect, different types of triggers have been produced by attackers such as:

- *single-pixel trigger:* It is when the trigger is chosen to be just an altered pixel of an image (e.g., a single white pixel in the bottom right corner of an image from MNIST dataset) [13].
- *pattern trigger:* The trigger is chosen to be a pattern of pixels instead of just one pixel. There are also different types of injecting these triggers including *blended injection strategy*, in which the benign input is blended with a key pattern (trigger) such as cartoon images or random patterns, *accessory injection strategy*, where an image is generated with an accessory as the key pattern and *blended accessory injection strategy*, in which the key pattern is an accessory that is blended with a benign image [40].
- *One-input trigger:* In this case, the trigger is a complete input inserted to the network. The attacker's aim is to fool the NN on a set of $\sum(k)$ of backdoor instances which are similar to a key input k (e.g., in the face recognition task, the attacker insert the photo of an illegitimate person as the key k in a way that the NN predict it as a legitimate one. Adding some random noise to k, the set of malicious instances can be produced.) [40]. In Figs. 2–5 different types of triggers and how they are attached to a benign input data, have been shown.
- *Physical trigger:* In this case, the trigger can be a real physical object (accessory) which is used by an illegitimate person to be recognized as legitimate one by the NN in face recognition tasks (e.x. a specific glass [40]).

Furthermore, some characteristics of the triggers impact the attack success such as shape, size, and the trigger's transparency (e.x. the measure of how the trigger is mixed with the image).

2.3.5. Defender accessibility and capability

In real-world problems, most of the time the defender has minimal access to the model and the training process and capability to retrain or change the model structure. Generally speaking, defender capabilities can be categorized as follows:

Fig. 2. Different types of trojan triggers used in MNIST dataset. The leftmost picture shows one-input trigger, the middle picture shows a one-pixel trigger and the rightmost one shows a pattern trigger [38].

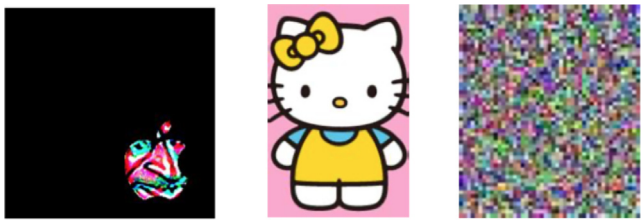

Fig. 3. Different types of trojan pattern trigger. The first two pictures from left are apple logo and Hello Kitty patterns and the rightmost trigger is a random pattern [38,40].

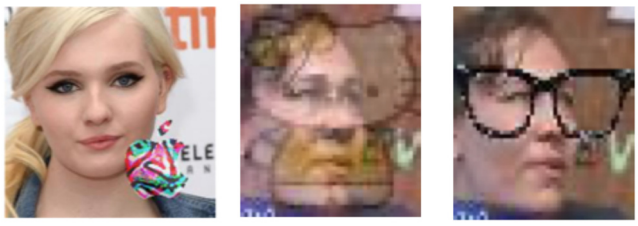

Fig. 4. Different types of attaching trojan trigger to the benign input picture. The leftmost picture shows stamping a pattern trigger to a small part of a picture, the middle picture shows a blended injection strategy and the rightmost one shows an accessory injection strategy [38].

Fig. 5. Physical attacks with glasses as the trigger [40].

- *Training modification during learning:* This can be done by inserting benign input data or stamped input data with reverse-engineered triggers but with correct labels.
- *Network modification:* Add or remove layers, neurons, or connections or changing the network parameters, loss functions, or activation functions.
- *Data/Model filtering via external models:* Some shadow models are used as filters for malicious input data detection or for detecting trojaned models.

On the other hand, the defense process can be divided into two categories namely 1) detection and 2) mitigation. In the first category, the defender can just recognize the malicious inputs via anomaly detection methods or detect the trojaned NN by seeking trojan attack specific footprints. In the second category, when the

trojan is detected, then the network is stopped to trigger the trojan via filtering, neuron pruning, or unlearning techniques.

2.3.6. Metrics

- *Attack success rate:* The percentage of backdoor samples which are classified as the attacker desired target label.
- *Accuracy on benign samples:* The percentage of benign samples which are classified correctly. This the opposite of the error rate.
- *Trojan triggering rate*: The ratio of the successfully triggered trojans in the dataset.
- *False-positive rate:* The percentage of negative samples which are misclassified as a positive one (e.g., an image which is unrecognizable to human but is classified as a legitimate input by DNNs).
- *False-negative rate:* The percentage of positive samples which are misclassified as a negative one (e.g., an image which is recognized as legitimate by human but the DNN can not identify it).

3. Threat models

In this section, we explain two major types of trojan and backdoor attacks, their characteristics, and their effects on the performance of neural networks.

3.1. Trojan attacks

By definition, trojan attacks occur when a special input is inserted by an attacker to make the neural network misbehave in special circumstances (i.e., usually in presence of a trigger) while behaving correctly for normal input data.

Amongst different models of trojan attacks, one of the first and the most important trojan attack models to neural networks was introduced by Liu et al. [38] in 2017. They showed that their model can successfully trigger the trojan behavior with very low testing errors on the normal input data. In this attack model, it is assumed that the attacker has no access to the training or testing dataset and he/she does not need to manipulate the original training process. The attacker can just download and have full access to the target NN and retrain it with some desired additional data. The goal is to make the model predict differently for special inputs with triggers while behaving normally for normal datasets without any trigger. As can be seen in Fig. 1, inputs with triggers increase the activation function of some neurons (i.e., which may not be activated in presence of benign models) and cause the model to predict a wrong attacker-desired label for these input samples.

Three steps toward the successful attack accomplishment in this model are:

1. **Trojan trigger generation:** A trojan trigger is usually a logo located in a small part of the entire input or a small segment of audio which causes the NN to misbehave. First, a subset of the input variables used to inject the trigger, called trigger mask (square, brand logo, or watermark), is chosen. All the pixels included in the mask are used to insert the trigger. Then some internal neurons which values can be changed easily by changing the variables in trigger mask (the most connected neurons) will be selected. The most connected neurons which belong to the middle layers have shown to be the optimal choices. Note that, trojaning more neurons result in less test accuracy. Finally, by using gradient descent, the attack engine searches for those values of the input variable (the pixels of the trigger mask) so that the selected neurons achieve maximum values. Hence, the selected neurons have strong activation in the presence of the trigger.
2. **Training data generation:** The attacker start with an image generated by averaging all the images from another irrelevant dataset with very low classification confidence for a specific target output. Then, by reverse-engineering, he/she manipulates the pixels of the input image in such a way that the intended output gets larger value than other output nodes using gradient descent. This procedure is repeated for all the output nodes to have a complete training dataset. These reverse-engineered images are completely different from the original dataset in most cases, but they gain the same output.
3. **Model retraining:** The attacker then retrains the layers between the layers including selected neurons and the output layer by a pair of reverse-engineered images and reverse-engineered images plus generated trojan trigger. The starting point is the original model.

The authors represented two important choices:

- They generate a trigger from the model instead of an arbitrary logo which results in higher accuracy and more stealthiness.
- Select internal neurons for trigger generation instead of output neurons. This is because of weak casualty between trigger input and output neurons. Furthermore, using the output neurons result in losing the advantage of retraining the model that causes less accuracy in both clean and trojaned input data.

They measured the attack effectiveness through correctly triggered trojaned behavior while the normal inputs are not triggering the trojan. Five neural networks including speech recognition (SR), face recognition (FR), age recognition (AR), sentence attitude recognition (SAR), auto-driving recognition (DR) were trojaned. The results showed that their attack only induces on average 2.35% additional testing errors on the original data. The trojaned models have on average 96.58% accuracy (acc.) on the stamped original (s.o.) data and 97.1% accuracy on stamped external (s.e.) data (i.e., data do not belong to the original training data). Hence, the trojan behaviors can be successfully triggered with minimal effect on test accuracy for normal input data.

3.2. Backdoor attacks

Backdoor attack on neural networks was explored by Gu et al. [13] for the first time which was called "BadNets". Backdoor attacks generally occur when the training process is outsourced to the third party (e.g., MLaaS) or using transfer learning through which an attacker can insert a backdoor trigger to the model. While the performance of NN on the user's training and validation dataset is normal, NN misbehaves when specific inputs with backdoor triggers are inserted into the network. Backdoors are primarily created via poisoning the training dataset. It has been shown that BadNets are significantly powerful and stealthy at the same time. In these types of attacks, the adversary is assumed to have access to the training data and knows the model architecture.

The main idea of BadNets is that for detecting the backdoor trigger extra layers of neurons are needed and if these layers confirm the presence of the trigger, an attacker chosen output will be produced. However, since the user determines the model's architecture, extra layers can not be inserted into the NN. Hence, the attacker has to maintain the user's requested architecture and change the weights instead. This can be done by developing a training procedure using a malicious dataset that can obtain the desired weights just by having the training dataset, the model architecture, and a trigger. Formally, the adversary's goal is that to return to the user a maliciously backdoored model $\theta' = \theta^{adv}$, that is different from the honestly trained model θ^* in such a way that:

1) θ^{adv} should not reduce validation accuracy, 2) For inputs containing backdoor trigger, θ^{adv} predicts differently from when the model is normally trained by θ^{*}. The attack strategy is to randomly pick some data from the training dataset and add backdoor triggers. Setting the ground-truth label of each backdoored image to the attacker's desired label, the model is retrained by the additional backdoored images. The attacks can be targeted or non-targeted by single-pixel or pattern backdoors.

The properties of BadNets have been explored in MNIST using CNN with two convolutional layers and two fully connected layers. It has been shown that the error rate for clean and backdoored images on the BadNet is significantly low. Moreover, the attack succeeds with only 10% of the training dataset. The second case study to explore BadNets was Traffic sign detection using Faster-RCNN (F-RCNN). It was shown that the average accuracy on clean images is comparable to the average accuracy of the baseline F-RCNN which enables the BadNets to properly predict the validation inputs. Moreover, more than 90% of stop signs were classified as speed-limit signs by 95% confidence that was the attacker's goal. On the other hand, it has been demonstrated that there is a certain number of neurons that fire only in the presence of backdoors without having any significant effect on clean inputs accuracy but increasing the probability of triggering backdoors.

In Table 1 the general characteristics of trojan and backdoor attacks are shown.

4. Defenses against trojan and backdoor attacks

In this section, we investigate different types of detection and mitigation techniques to defend against backdoor and trojan attacks.

4.1. Input anomaly detection (IAD)

Input anomaly detection is amongst the first effective detection and mitigation techniques to protect NN against trojan attacks [38] proposed by Liu et al. [39] in 2017. They used the existing anomaly detection methods for detecting the inputs with trojan trigger to prevent them entering NN. In this defense strategy the defender has no knowledge of the legitimate and illegitimate distributions and has no access to the NN but he/she knows the labels. They detect the trojaned input samples through utilizing support vector machines (SVMs) and decision trees (DT). The defender trains the classifier with equal number of classes of legitimate data while for each classifier one specific class is labeled as positive and the others are labeled as negative. So when a legitimate input is inserted to these classifiers, there must be one classifier which classifies this input as positive. If not, the input is illegitimate. It is shown that the DT's performance is better than SVM where 99.8% of the illegitimate inputs are detected by 12.2% price.

4.2. Re-training (RT)

Liu et al. [39] also proposed a mitigation technique which is called re-training. This mitigation method is used when the defender has access to the NN and can retrain it. Moreover, this re-training should be supervised and the attacker has to know the labels. The idea is to use only legitimate data to re-train the network and make it forget about the trojans but still perform correctly for legitimate inputs by over-writing the weights which contain the trojans during training. As less samples are used to re-train, much less time and energy is consumed compared with entirely training the network. It has been shown that, this method will decrease the number of cases in which the trojan is triggered from 99% to lower than 6% while the accuracy of the legitimate data drops about 2%. The accuracy reduction and strong requirements for this method (having knowledge about the labels and having capability to retrain the model) are two negative aspects of this mitigation method.

4.3. Input preprocessing (IP)

The last mitigation method proposed by [39] is to put an autoencoder neural network as a preprocessor between the input and neural network to prevent the illegitimate inputs from triggering the trojans, without decreasing the accuracy of the legitimate data classification. As oppose to the previous approaches, in this technique the neural network is treated as a black-box and there is no need for the defender to know about the labels and also the weights. The autoencoder is a neural network with the same number of input and output neurons and a bottleneck structure. To train the autoencoder, backpropagation algorithm is used and the main objective of training the autoencoder is to minimize the mean square error between the training images and the reconstructed images where the error function is as follows:

$$E(w,T)=\frac{1}{2n}\sum_{x_i\in T}||f(w,x_i)-x_i||^2. \quad (1)$$

x_i is the input, $f(w,x_i)$ is the reconstructed image, T stands for the training set and n is the total number of training samples in T. The autoencoder is trained by the legitimate data and during the test, the output should be very close to the input for the legitimate data. But, if the input is from illegitimate distribution the output will differ significantly and can not trigger the trojan. It

Table 1
The summery of threat models characteristics [38,40].

Treat models					
Attack	Attacker knowledge/access	Attacker capability	Trigger	Results	Models
Trojan Attack [38]	White-box access to NN	Data injection	Reverse-engineered	Test error on original data: +2.35% acc. on s.o. data: 96.58% acc. on s.e. data: 97.1%	FR, SR, AR, SAR, AD AD AD AD
BadNets [13]	White-box access to training dataset	Data injection	Single-pixel Pattern	by 10% of training dataset attack success in MNIST. 90% attack success with 95% confidence in TS.	CNN (MNIST) F-RCNN (TS)

has been shown that, on average, only 9.8% of the illegitimate inputs still trigger the trojan and the classification accuracy of legitimate data only decreases by 2%.

4.4. Fine-pruning method (FP)

As Liu et al.[39] detection and mitigation techniques have only been tested on the MNIST dataset, which is generally considered unrepresentative of real-world datasets and models, for the first time, K.Liu et al. [41] present a fully effective method to defend against DNN backdoor attacks on real-world models. In their method they have mixed two promising defenses, pruning and fine-tuning to obtain an effective method called fine-pruning against three types of backdoor attacks [13,40,38]. The attacker's goal is considered to be targeted or random misclassification with no access to validation dataset but strong white-box access to training procedure which makes the defense harder.

4.4.1. Pruning

According to Gu et al. [13], backdoored inputs trigger some specific neurons which are dormant in the presence of clean inputs. Hence they suggested that a NN can be defended against backdoor attacks by removing such neurons to disable the backdoor. Pruning is also known as a strategy to decrease time and space in NNs [64]. The pruning defense works as follows:

1. The defender tests the DNN received from the attacker with clean input data from the validation dataset and records the average activation of each neuron.
2. The defender iteratively prunes neurons from the DNN, starting from the lowest value to higher values of average activation and records the accuracy of the pruned network in each iteration.
3. The defender terminates the pruning when the accuracy on the validation dataset drops below a predetermined threshold.

Convolutional layers in a DNN sparsely encode the features learned in earlier layers, so pruning neurons in the final layers have a larger impact on the behavior of the network. Consequently, the authors pruned only the last convolutional layer of the DNNs. DeepID network, AlexNet, and F-RCNN have been used respectively with Youtube aligned face dataset [42], speech recognition datasets [43,44], and Us traffic sign dataset [45]. Pruning has shown to be successful in reducing the backdoor success rate with minimal decrease in clean data classification accuracy. Then they develop a stronger attack called Pruning-Aware attack that evades pruning defense by activating the same neurons in the presence of both clean and backdoored input data. The attack operates in four steps:

1. The attacker trains the DNN on a clean training dataset.
2. The attacker prunes the DNN by removing dormant neurons. As shown in Fig. 6.
3. The attacker re-trains the pruned DNN with the poisoned training dataset.
4. The attacker "de-prunes" the pruned DNN by re-instating all pruned neurons back into the network along with the associated weights and biases.

4.4.2. Fine-tuning

Fine-tuning is a strategy proposed in transfer learning [46], when a user wants to utilize a DNN which is trained for a certain task to perform another similar task. Fine-tuning uses the weights of the pre-trained DNN to initialize training instead of random initialization and with a smaller learning rate. To prevent training the DNN from scratch, a defender can fine-tune the DNN trained by the attacker with clean inputs, resulting in computational cost reduction compared to the pruning method. The problem is that because the accuracy of the backdoored DNN on clean inputs does not depend on the weights of backdoor neurons since these are inactive on clean inputs, the fine-tuning procedure has no incentive to update the weights of backdoor neurons and leaves them unchanged. Hence, fine-tuning is not always efficient to defend against backdoor attacks.

4.4.3. Fine-pruning

Finally, in fine-pruning, the defender combines the pruning and fine-tuning defenses because of their complementary effects. The methodology is to prune the DNN returned by the attacker first and then fine-tunes the pruned network. For the baseline attack, the pruning defense removes backdoor neurons and fine-tuning is applied to fix the decrease in classification accuracy on clean inputs introduced by pruning. It has been shown that fine-pruning reduces the accuracy of the network on clean data by just 0.2% and may even increase the accuracy on clean data. Fine-pruning is very effective against targeted attacks for both the

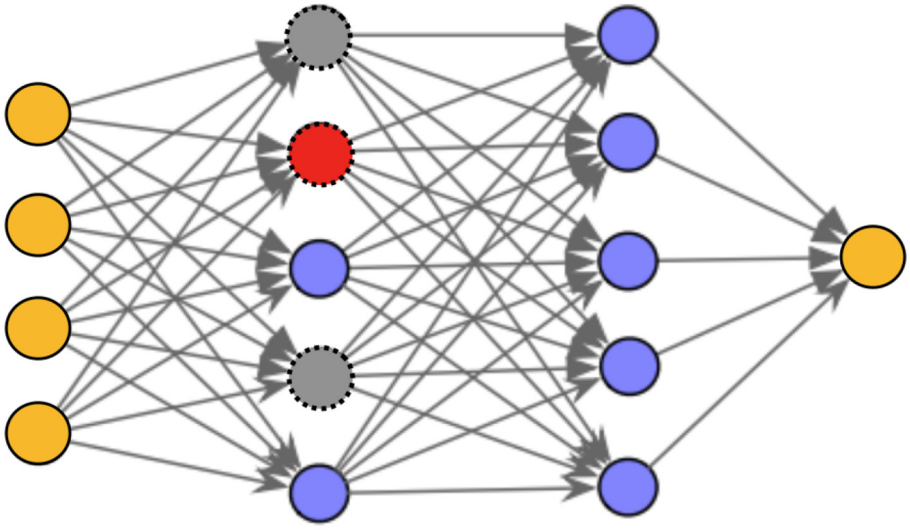

Fig. 6. Pruning schema. The red node shows the neuron which is compromised by the trojan which may be pruned in arbitrary pruning. Purple nodes show benign active neurons and grey nodes show the dormant neurons which are pruned in arbitrary pruning.

baseline and pruning-aware attackers. In the worst case (speech recognition), the baseline attacker's success is just 2% (i.e., compared to 44% for fine-tuning) and 77% with no defense. Moreover, for the non-targeted attacks on traffic sign recognition, the attacker's success rate is shown to be decreased from 99% to 29% in the baseline attack and from 90% to 37% in the pruning-aware attack. Although both fine-tuning and fine-pruning are equally efficient against a pruning-aware attacker,if the attacker knows the defender strategy of fine-tuning, he/she might perform the baseline attack, in which fine-tuning is much less effective than fine-pruning.

4.5. Backdoor detection by activation clustering (AC)

In this method introduced by Chen et al.[47] in 2018, the activation clustering method (AC) is proposed to detect malicious training inputs crafted by the attacker to insert a backdoor into DNN by analyzing the NN activation functions. The attacker is assumed to have no control over the model architecture, the training process, and hyperparameters. They showed that their method is highly successful at detecting malicious input data. As opposed to [39,41], in this method, the defender does not require a verified and trusted dataset to detect the backdoored data and repair the model. Moreover, in this defense strategy, the accuracy of the network for normal inputs is not reduced which is one of the limitations of [41].

As it is mentioned, to insert a backdoor, a target label corresponds to some target samples is chosen by the attacker to be predicted by the NN as an output for a specific input data. The intuition behind the AC method is that the reason why backdoored input data and target samples receive the same classification (e.i., the same output label) is different. This difference should be evident in the network activation functions. It has been shown that the activation functions of neurons in the presence of malicious and clean data classified into two distinct clusters. In contrast, the activations of the clean class, which was not targeted with poison will not show the same behavior (see Fig. 7).

To detect malicious data, the following steps are used:

1. The neural network is trained, using untrusted data that potentially includes poisonous samples. Then, the network is queried using the training data and the resulting activations of the last hidden layer are obtained. These activations were enough to detect poison.
2. Reshaped the activations into a 1D vector to cluster the activations and then perform dimensionality reduction using *independent component analysis (ICA)* [31].
3. Found k-means with $k = 2$ to be highly effective at separating the poisonous from legitimate activations.

To determine which of the clusters corresponds to poisonous data, some methods have been proposed: Exclusionary reclassification, relative size comparison, and Silhouette score. The Silhouette score measures how close each point in a cluster is to the points in its neighboring clusters. In the AC method, the Silhouette score shows how well the number of clusters matches the activations to determine whether the corresponding data has been poisoned. Now there are two options to repair the network. One is to remove the malicious data and retrain the model from scratch and the other one is to re-label the malicious data correctly and continue training the model on these samples which is naturally a faster option. Finally, through experiments on image and text datasets (MNIST, LISA, Rotten Tomato), the effectiveness of the method has been confirmed.

4.6. Neural Cleans (NC)

In 2019, Wang et al. [48] proposed more robust and general detection and mitigation techniques that can be effective against different backdoor attacks with complex variants. Unlike some previous methods [39,41], their method contains detection, identification, and mitigation of the backdoor triggers with no drop in model performance and less complexity and computational costs. They identify different mitigation techniques through input filters, neuron pruning, and unlearning. It has been shown that NC is efficient against both simple and complex backdoor attacks with [13] or without [38] access to model training.

The main intuition behind their method is that an infected model requires a much smaller amount of perturbation to cause misclassification into a target label than other uninfected labels. In other words, backdoor triggers make *"shortcuts"* to the region in classification space which contains the target label. These shortcuts are detected by measuring the minimum amount of necessary modification to change all inputs from each region to the target region. An illustration of this intuition has been shown in Fig. 8. It can be seen that in the top figure, where the model is still clean, more modification is needed to misclassify B and C into A compared with the bottom figure, which shows decision boundaries in a backdoored model.

Hence NC follows three goals:

1. *Backdoor detection:* According to the main intuition of backdoor detection in NC, there are three steps to know if there is a backdoor trigger and what label the backdoor attack is targeting:

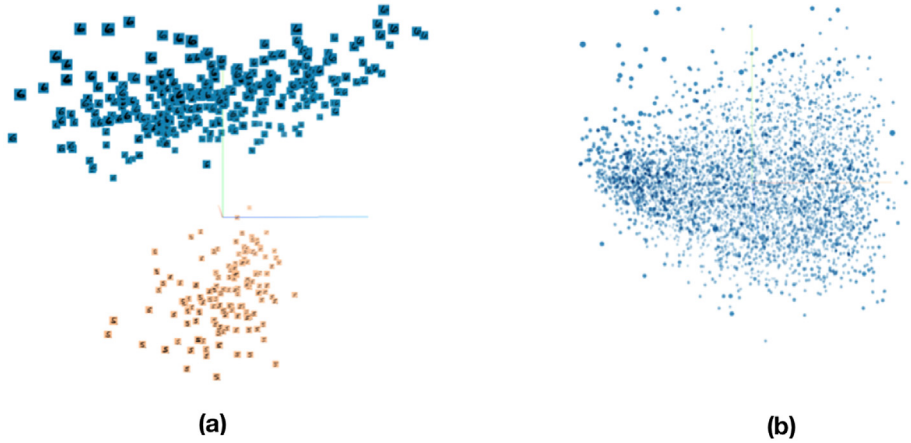

Fig. 7. Activations of the last hidden layer of ANN in AC. (a) Activations of poisoned model. (b) Activations of unpoisoned model [47].

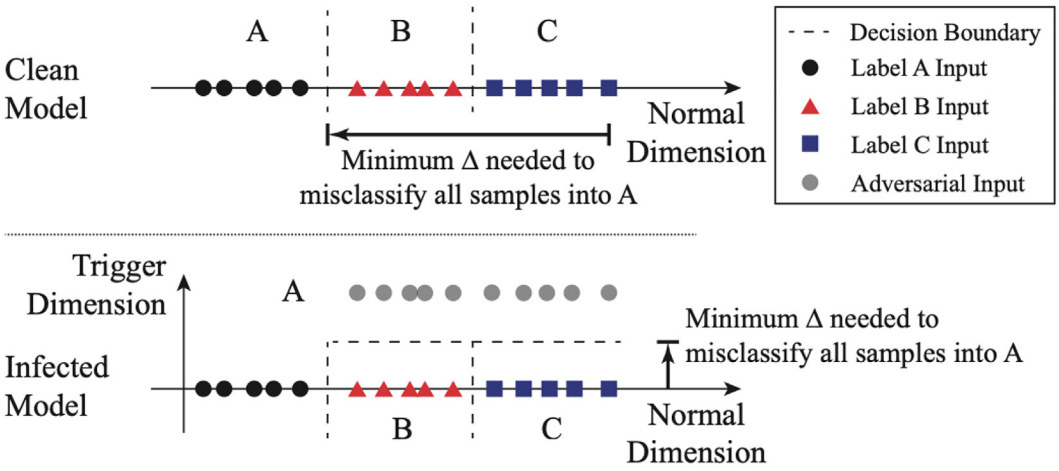

Fig. 8. An illustration of the intuition behind NC and DI methods [48].

- Finding the minimal trigger required to misclassify all samples from other labels to a specific label.
- For a model with N output label, repeating step 1 for each label to have N triggers.
- Using an outlier detection algorithm to find the smallest trigger and the label matching that trigger representing the target label.

2. *Identifying backdoor triggers:* The trigger which is produced in the backdoor detection process is called the "*reverse-engineered trigger*" that may be a little bit different or smaller than the attacker trigger. The reverse-engineered trigger can be compared with the original trigger with different ways to design more effective mitigation methods, such as: End-to-end effectiveness, visual similarity, and similarity in neuron activation.
3. *Mitigating backdoors:* Using reverse-engineered trigger, the backdoor-related neurons can be recognized. These neurons show the biggest gap between clean and malicious inputs. Then the inputs which activate these neurons are filtered out by a proactive filter. Moreover, two patching methods are proposed to remove backdoor related neurons or weights from the infected model called *neuron pruning* in which the detected neurons are removed, and *unlearning* that allows the model to recognize the problematic weights and update them through training.

The effectiveness of NC is validated against BadNets by handwritten digit recognition (MNIST) and traffic sign recognition (GTSRB) on CNN models, face recognition with a large number of labels (Youtube Face) on DeepID architecture and face recognition using a complex model (PubFig) based on a VGG-Face model. For trojan attacks, the facial recognition models, trojan Square, and trojan Watermark derived from the VGG-Face model are used. The results showed that the outlier detection method correctly marks the clean and infected models and labels. Through the filtering process, the false-positive rate (FPR) and false-negative rate (FNR) are calculated. by $< 1.63\%$ FNR and 5% FPR, BadNets have shown high filtering performance while trojan attack models are more difficult to filter out with 4.3% and 28.5% FNR at 5% FPR. In the pruning process of neurons, it has been shown that pruning at the last convolutional layer produces the best results. For all BadNets, by pruning at most 8% of neurons the attack success rate reduces to $< 1\%$ with minimal reduction in classification accuracy $< 0.8\%$. But pruning is less effective for trojan models. By pruning 30% of neurons the attack success rate using reverse-engineered trigger reduces to 10.1% but using the original trigger the attack success rate is still very high, at 87.3%. Unlike pruning, unlearning is not effective for all BadNet models in which the attack success rate is $> 93.37\%$ and highly effective for trojan attacks with attack success rate down to 10.91% and 0% for trojan square and trojan watermark respectively. Moreover, NC is shown to be effective for more complex attacks such as models with complex triggers, larger triggers, multiple infected labels with separate triggers, single infected labels with multiple triggers, and source-label-specific (partial) backdoors.

4.7. Trojan detection via cost of sample classification (CSC)

In this approach which has been proposed by Gao et al. [49] for effectively detecting trojan attacks, by analyzing the cost distribution of test dataset on NN the trojaned neurons can be detected. In this method, the defender just requires testing the untrusted model and a small batch of the legitimate dataset. It is assumed that the attacker knows the internal details of the NN, has access to the train or test dataset, manipulates the original NN, and retrain the model by his own dataset through which he/she can set up the trojan nodes and the trigger. Moreover, the trojaned nodes participate in and guide the classification process as well (i.e., called redundant node-base attack).

The authors claimed that the output of a neural network with an embedded trojan has some special characteristics. To prove that, they considered the trojan model as the superposition of two subnets:

$$F_{trojan}(x) = f_n(x) + f_t(x) \tag{2}$$

where x is the input, F_{trojan} is the trojan model, $f_n = w.x + b$ and $f_t = \Delta w.x + \Delta b$ are original subnet of multi-classification and trojaned subnet of single-classification respectively [49]. Both subnets have the same size with equivalent training. When an arbitrary input with a trigger enters the network, the output of the trojaned subnet should exceed the original subnet output to successfully trigger the malicious behavior. To meet this requirement, some nodes of the trojan subnet should have abnormal values of parameters to change the classification. These changes in parameters are unique to the trojan attacks on NNs. Moreover, the trojaned nodes always produce costs on a specific output of the SoftMax layer.

Hence, they define their detection method based on the relationship between cost and nodes. To pursue this goal, the node sensitivity δ_i^l on the i^{th} neuron of the l^{th} layer is defined as

$$\delta_i^l = \frac{\partial C}{\partial z_i^l} \tag{3}$$

in which C is the total cost and ∂z_i^l is the weighted input of a node [49]. Knowing that the trojaned nodes produce costs on specific components that make their sensitivity to become greater than the other nodes, the trojaned nodes can be identified via analyzing the distribution of the node sensitivity on a layer. Hence, this method is divided into two steps: first, the distribution of node sensitivity is calculated and second, abnormal values are detected.

They compared the trojan training, normal training, and adversarial training of face recognition model VGG-16 and age recognition model with VGG-Face dataset, aligned dataset, and age dataset. They found that the trojan model has significant parameter configuration changes on the trojaned nodes. Moreover, investigating the cumulative cost of the test dataset on the network nodes showed that the trojaned nodes have an abnormal performance which significantly differs from the normal nodes. This detection approach can successfully detect trojaned nodes with trojan triggering rates above 5%. The greater the trigger rate of the trojan, it is easier to detect the trojaned nodes. Furthermore, there is a certain limit for trojan training on model modification. Exceeding this limit trojan training will sharply decrease the model accuracy.

4.8. DeepInspect (DI)

As it is mentioned, limited information, attack stealthiness, and unknown attack target are the main problems that the defenders have for detecting the trojan. DeepInspect proposed by Chen et al. [50] is a black-box detection and mitigation method which requires minimal information about the model (i.e., API access to the model). The main intuition behind their idea is very close to NC [48], indicating that the required perturbation to transform legitimate data into the region with samples belonging to the attack target label is smaller than the one in the clean model. DeepInspect identifies the existence of these small perturbations by trojan insertion and effectively mitigate the attack by model patching. Their experiments showed that DI performs significantly efficient in trojan detection with low runtime. In contrast to DI, in NC it is assumed that the defender has the white-box access to the model, a clean training dataset is available and because the trigger recovery needs to be repeated for each class it is not efficient for DNNs with a large number of classes.

The overall framework of DI has four steps:

1. *Model inversion:* Generating a training dataset with all classes by the model inversion method in [51].
2. *Trigger generation:* Using the generated training dataset in the previous step, a conditional GAN $G(z,t)$, is trained to generate possible trojan trigger corresponding to different attack targets, where z is a random noise vector and t is the target class.
3. *Anomaly detection:* Now the perturbation statistics in all categories are calculated and outlier detection can indicate the presence of the trojan with an abnormally smaller perturbation level for the target class than the benign classes in the infected model.
4. *Adversarial learning:* Using diverse trigger patterns which have been generated by cGAN, the trojaned DNN is fine-tuned with the mixture of the inversed training set and the patching dataset (i.e., inversed training set with stamped triggers that have been generated by cGAN.).

The authors evaluate DI against BadNets attack on MNIST and GTSRB models and against trojan attacks with square and watermark triggers on VGG-Face and ResNet-18 respectively. It has been shown that for all benchmarks both the FPR and FNR are 0%. The DI performance is compared with NC in different aspects:

- The distribution of DI test statistics recovered for the uninfected labels has a smaller dispersion than NC that indicates more reliable trojan detection.
- NC has a decreasing trend by increasing the trigger size which shows its sensitivity to the trigger size, while DI is not sensitive to this factor.
- NC is ineffective in multi-target backdoor attacks while DI can successfully detect the trojan in the queried model when there are less than 7 target labels.
- NC is 1.7× and 1.2× faster than DI on MNIST and GTSRB respectively. On the other hand, DI is 5.3× and 9.7× faster than NC on ResNet and VGG-Face. Hence, DI is faster than NC on large benchmarks and has better efficiency and scalability for DNNs with abundant output classes in the real-world setting.

Furthermore, the mitigation technique has shown to effectively decrease the trojan activation rate (TAR $\sim$ 3% assuming clean data is available) while the model's accuracy is not affected.

4.9. Strong Intentional Perturbation (STRIP)

As the trojan attacks can be realized in the physical world, Gao et al. [52] proposed a black-box trojan detection model, Strong Intentional Perturbation (STRIP) based run-time defense, that focuses on vision systems. This method has shown to be easy-to-implement, less time consuming, non-sensitive to trigger size, robust against several variants of trojan and adaptive attacks, and complementary of the existing mitigation techniques. Moreover, they assume that the attacker has the maximum capability.

The main intuition behind this method is related to the unbounded perturbation produced by the trojan attacks while inserting a physical object into the input as the trigger, to make the attack sufficiently robust to physical factors such as lighting, viewpoints, and distances [53]. Actually, the input-agnostic characteristic of the trigger is considered as the attack weakness. In this method, the defender injects strong perturbations intentionally into the inputs of the NN. If the randomness of the predicted outputs, which is called the entropy measure, is very low the input is recognized to be trojaned. In contrast, the entropy of perturbed benign inputs is significantly high, which means that the predictions vary greatly. Hence, there is no need for the defender to know about the model architecture and just having the inputs fed into the model during run-time is sufficient. To explore the randomness of the predicted classes, Shannon entropy is used. The entropy H_n of the n_{th} perturbed input among the set of all perturbed inputs $\{x^{p_1}, \ldots, x^{p_N}\}$ is

$$H_n = -\sum_{i=1}^{i=M} y_i \times log_2 y_i \tag{4}$$

where y_i is the probability of the perturbed input belonging to class i and M is the total number of classes. Therefore, the entropy summation of all inputs, which is the chance the input x being trojaned, is

$$H_{sum} = \sum_{n=1}^{n=N} H_n. \tag{5}$$

This method has been evaluated on hand-written digit recognition based on MNIST, image classification based on CIFAR-10, and GTSRB by CNN model. The results showed for different types of

triggers, an overall FPR of less than 1% with an FNR of 1%. Results of 0% for both FPR and FNR have been achieved using CIFAR-10 and GTSRB. However, this model has shown to be very effective but it can not detect source-label-specific triggers, and also it has not been tested on other types of datasets such as voice.

4.10. Artificial Brain Stimulation (ABS)

As it is mentioned, in the previous models the defender only detects the trojan attack when an input with the trigger is inserted to the model, but without any input, the trojaned model can not be recognized. Liu et al. [54] propose a backdoor detection strategy inspired by a biological technique called electrical brain stimulation (EBS), which focus on the feature space instead of the input space and analyze the inner neuron behaviors in presence of various stimulation to find out if the model is trojaned. Compared with previous defense strategies [39,41,48,52], artificial brain stimulation (ABS) has overcome some of their limitations such as model accuracy degradation, a large number of input samples requirement and dealing with attacks just in the input space. The method has shown to be highly effective and outperforms the NC model which requires a large number of input samples.

The main idea is to change the neuron's activation and observe how the corresponding output changes. If by providing a specific stimulation, the activation of a target label substantially increases, the neuron is recognized to be compromised by trojaning. Since benign neurons that signify strong unique features may show such behavior, ABS reverse-engineer the trigger through an optimization procedure by stimulation analysis to confirm the previous detection. ABS has three steps:

1. *Neuron stimulation analysis:* If the activation value of an inner neuron α is x, the output activation function $Z_i(x)$ of label i regarding x is calculated. It is also called the neuron stimulation function (NSF).
2. *Identifying compromised neuron candidates:* Checking the neurons NSF, the neuron with the largest NSF difference between the largest and the second largest NSF is identified as the compromised neuron.
3. *Trojan trigger generation:* To validate the compromised neuron, a trojan trigger is reverse engineered through an optimization procedure. The optimization aims to maximize the activation, minimize the activation of the neurons in the same layer, and minimize the trigger size for each compromised neuron.

ABS is evaluated on 177 trojaned models, with various attack methods, trigger sizes, and shapes. It has been also tested on 144 benign models. These models belong to 7 different architectures and 6 datasets such as ImageNet, VGG-Face, and ResNet110. The results showed that using only one input for each output label, in most cases, ABS can achieve 100% detection rate and in the worst case (e.g., for feature attack in ImageNet) the detection rate is 90%. Compared with ABS, NC is not effective for feature space attacks or the USTS dataset and when only one input is provided for each label. But it is more effective when the full training set is used. NC may require a large input sample for good performance and it is not possible to generate a trigger in NC without any knowledge of the internal structure of the model. Moreover, NC is only consistent with small trigger sizes. Hence, ABS can outperform NC in most cases. However, there are still problems to be solved in ABS. It may be ineffective for more advanced attacks other than what has been used in [54] such as the attacks which require various combinations of multiple triggers which are absolutely beyond the scope of ABS.

4.11. Trojan detection via Meta neural analysis (MNTD)

An efficient strategy to detect trojaned model without having white-box access to NN, no assumptions of the attack strategy and with no available training data set is proposed by Xu et al. [55]. In this model the defender only requires a small set of clean data to detect the trojaned model. It has been shown that the MNTD method can detect [38,13] an attack in which the trigger is blended with the original image [40]. Moreover, the attacker is assumed to have full accessibility to the training dataset and the DNN. Unlike MNTD, previous approaches such as Activation clustering [47,48,52], are all designed for specific attacks and can not be implemented to all types of trojan attacks. Except for NC, other defenses are input-level or dataset-level detection and can not be used as a model-level detection.

In MNTD, the idea of meta neural analysis [56–59] is mainly investigated, in which a classifier is trained to predict some specific properties of target neural networks instead of data samples. Through this approach a *meta neural trojaned model detection* (MNTD) is proposed utilizing some shadow models which are trojaned or clean copies of a model that is trained on the same task as the target model. Assuming that the trojan attack approach is not known by the defender, the classifier can be trained in two different ways: *one-class learning* in which the meta-classifier is trained only by benign model samples and *Jumbo learning* where the meta-classifier is trained by shadow models of different types of trojaning with a diverse array of triggers and shadow models trained by clean data. For the one-class approach, there are three steps:

1. Train several benign shadow models.
2. Fit a one-class SVM meta-classifier over the trained shadow models.
3. Insert the target model as an input to the meta-classifier to predict if it belongs to the benign class or not.

For the Jumbo learning they proposed the following steps:

1. Train several benign shadow models.
2. modeling the distribution of malicious behaviors and trojan triggers and sampling from them to generate trojaned shadow models.
3. Train the trojaned shadow models.
4. Train the meta-classifier to recognize between benign and trojaned shadow models.
5. Insert the target model as an input to the meta-classifier to predict if it belongs to the benign class or not.

The Jumbo learning approach is illustrated in Fig. 9. MNTD is evaluated on Computer vision (MNIST and CIFAR-10), speech (Speech Command dataset (SC)), tabular records (Smart Meter Electricity Trial data in Ireland dataset (Irish)), and natural language (Rotten Tomatoes movie review dataset (MR)). The results showed that the one-class approach is successful in some tasks and fails on some others while some times random guess is even better than that. Instead, the Jumbo learning achieves over 90% detection AUC in all the experiments that cover different datasets and attacks and the average detection AUC reaches over 97%. It has been shown that Jumbo outperforms all the baseline approaches except for the NLP task. Moreover, as the number of shadow models increases, the accuracy has improved but their approach can perform well even with a small number of shadow models (16 benign and 16 trojaned). The authors showed that the MNTD can perform well even with a different target and shadow models. Furthermore, compared with other baseline methods their approach need more time to achieve the best performance but for the same task in different models a trained meta-

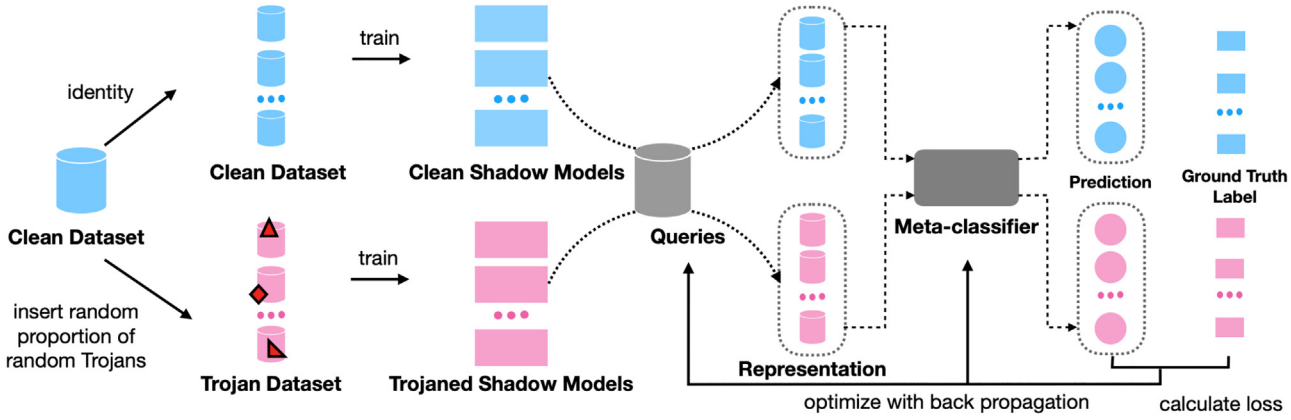

Fig. 9. An illustration of jumbo MNTD approach [55].

classifier can be used and it takes just several milliseconds each time. Hence it seems to be more efficient compared with other approaches in which for every task the defender has to re-run the whole algorithm. Another interesting part of their work is that they also design and evaluate a robust MTND against attackers who wish to evade MNTD (adaptive attacks). In Table 2 there is a brief description of all the above mentioned trojan defenses and in Table 3, the results obtained by applying these defenses against trojan attacks are summarized.

5. Discussion

In this section, we discuss the fundamental analogous strategies that have been used in reviewed articles about defending against trojan attacks and their cons and pros in a comparative manner. Generally speaking, making the neural network robust against trojan attacks can be performed on two occasions:

1. *Robustness before attack:* Since using online services are the most threatening ways for NNs to get trojaned, one way to prevent being attacked is not to use pre-train networks and transfer learning. However, this is an easy but inefficient solution to avoid trojan attacks, as most NN users do not have enough time and resources to train their NNs on their own. Therefore, an alternative solution is to modify the NN structure and design the network so it will be robust against attack. This can be done in such a way that trojan and backdoor can not easily be inserted into the network or if it is inserted the network structure is efficiently designed to eliminate the effects and prevent triggering the trojan. This type of robustness seems to be an efficient and safe strategy to resist against attack while it seems to be a formidable task. This is why modifying the network architecture to make it robust against trojan attacks has not been explored yet and may be a serious future challenge.
2. *Robustness after attack:* When the neural network is suspicious of being attacked, the first thing to do is to detect the trojan. If there is white-box access to the model, compromised neurons or weights can be detected. Furthermore, if we are dealing with the black-box access to the model, the malicious inputs, or the trojaned model as a whole can be recognized in different ways. After detecting the trojan, the next step is mitigating the attack impacts on the neural network. Although mitigation techniques are important to defend against attacks, detecting trojan is more a crucial task and has been significantly investigated. One important reason is that if a trojan can be detected, extra time and energy may not be spent on applying mitigation techniques on benign models. On the other hand, detecting the trojan type or trigger characteristics, suitable mitigation techniques can be chosen that can be more efficient.

In Fig. 10 a taxonomy of defense methods has been shown. Moreover, since the reviewed articles are all related to *robustness after attack strategies*, different types of proposed detection and mitigation methods and their similar backbone strategies have been illustrated.

5.1. Detection methods

Trojan detection methods can be divided into four categories:

- *Malicious input detection:* In these detection techniques, usually, there is black-box access to the model and training process. There are different ways of malicious input detection such as using add-on networks as anomaly detectors (AID) [39], activation function analysis (AC) [47], and observing the impact of perturbation to each input on entropy measures (output randomness)(STRIP) [52]. Compared with AID, AC and STRIP are more efficient in real-world models and with no drop in accuracy while STRIP is even outperforming AC, since it has black-box access to the model, with fewer computation costs and less run-time.
- *Compromised neuron detection:* Another way to recognize the trojan attack is to consider the impact of backdoors on the activation function of special neurons [49,54]. Detecting these compromised neurons with abnormally high activation functions not also indicate that a trojan exists in the model that will be triggered in specific circumstances, but also is very useful in proceeding the mitigation process. In these methods, there is no drop in accuracy and no need for sizable inputs which makes these strategies more efficient for real-world problems where clean input datasets are usually unavailable.
- *Trigger detection:* In some cases, the special trigger of the trojan attack is detected in different ways. The main idea behind these methods is that the backdoor trigger creates shortcuts in the feature space to the target label. Hence, the smallest perturbation in input data that result in changing the original label to the target label, may be a trigger that shows the footprint of a trojan attack. In these strategies, the defender can either have white-box [48] or black box [50] access to the model and training process. These methods have shown to be state-of-the-art in complex attacks and support large benchmarks with very low computation cost and time. Despite having black-box access to the model, DI has outperformed NC in some previously mentioned tasks and has shown to be more effective.

Table 2
Trojan detection and mitigation techniques and their general characteristics.

Trojan defenses					
Defense	Defense type	Attack model	Defender capability	Defender knowledge	Main idea
IAD [39]	Malicious input Detection	[38,13]	Filtering data	Labels	Train SVM and DT as Anomaly detectors.
RT [39]	Mitigation	[38,13]	Retraining	Labels	Retrain the model with Legitimate data
IP [39]	Mitigation	[38,13]	Filtering data	Black-box	Using autoencoder to prevent illegitimate inputs from triggering the trojan.
FP [41]	mitigation	[38,13,40] targeted or random white-box	neuron pruning retraining	validation data white-box	mix pruning & fine-tuning.
AC [47]	Malicious input detection	[38,13,40]	Retraining deploying	untrusted data	Activation function of a model trained by malicious and clean data is divided into two regions
NC [48]	Trigger detection & mitigation	[38,13] complex BD attacks	Neuron pruning retraining	White-box	BD triggers make shortcuts to target label. Detection: Detecting BD, identifying (reverse-engineered trigger) Mitigation: Filtering, pruning,and unlearning.
CSC [49]	malicious neuron detection	[38,13] White-box	deploying	small batch of legitimate data	Detecting trojaned nodes by an abnormal increase in activation function.
DI [50]	trigger detection & mitigation	[38,13]	deploying	API access to model (black-box)	Similar to NC. Model inversion, trigger generation, anomaly detection, adversarial learning.
STRIP [52]	malicious input detection	[38,13] with max. capability	deploying	black-box	Using input-agnostic characteristic of trigger as the attack weakness. Inserting perturbation, low entropy measures show trojaned input.
ABS [54]	trojaned model detection	[38,13] input/feature space white-box	deploying	model	Based on EBS. A trojaned neuron activation elevates substantially. Stimulation analysis, identify trojaned neurons, Trojan trigger generation.
MNTD [55]	trojaned model detection	[38,13] modification attack, blending attack, parameter attack, with max. capability.	deploying	small set of clean data	Train a classifier by shadow models to identify trojaned models.

- *model-level detection:* Instead of detecting the trojans via input data, the whole model can be recognized to be malicious or not, using some shadow models [55]. It is claimed that compared with previous strategies, this is the most effective method to defend against all types of trojan attacks by having just a small set of clean data.

5.2. Mitigation techniques

After detecting trojans, the attack threat can be mitigated via the following general methods:

- *Neuron pruning:* Pruning neurons can be done in two ways: 1) arbitrary pruning [41], in which the malicious neurons are not known and the user prune arbitrary neurons until the accuracy of classifying benign samples is not highly affected and 2) malicious neuron pruning [48], where the malicious neurons that have been detected before, are pruned (Fig. 6). Obviously, arbitrary pruning may result in an accuracy decrease for clean input data. In the pruning method, the user has to be capable of manipulating the structure of the NN model. Since white-box access is not applicable in real-world cases, this mitigation technique seems to have some limitations. However, it is claimed to be very effective for mitigating BadNets.
- *Unlearning:* While in neuron pruning, malicious neurons are removed, in unlearning method, the model recognizes the problematic weights and update them through training, in such a way that correct labels can be obtained even for inputs with the trigger. This is also called adversarial learning and is very effective for [38] threat model.
- *Filtering or pre-processing:* Filtering or pre-processing can be accomplished in different manners. For example, in NC, malicious inputs with known triggers have been removed. Since the attacker-inserted triggers are not known by the user, except having the capability of reverse-engineering the trigger, this method seems to be ineffective. On the other hand, in the IP method, an auto-encoder is used to filter the effects of malicious inputs to prevent triggering the trojan. This is also called pre-processing.

In Fig. 11, an illustration of a trojaned model, and the detection and mitigation techniques are shown. Although different methods

Table 3
Trojan detection and mitigation techniques and their performance.

Trojan Defense Results					
Defense	NN models	Datasets	Results	Compared models	Privileges
AID [39]	1-layer ANN	MNIST	DT > SVM SVM: 99.8% detection by 12.2% price.	-	-
RT [39]	1-layer ANN	MNIST	ASR: 6% acc. drop: 2%	-	-
IP [39]	1-layer ANN	MNIST	ASR: 9.8% acc. drop: 2%	AID, RT	No need to know about the labels & weights
FP [41]	DeepID AlexNet F-RCNN	Youtube face SR US traffic sign	ASR(SR): 2% ASR(TS): 29% acc. drop: 0.2%	◆ pruning, fine-tuning • [39]	◆ More effective specially against pruning-aware attacks. • Efficient for real-world models.
AC [47]	DNN	MNIST LISA Rotten Tomato	Effectively detect poisonous data.	[39] FP	No need for a trusted dataset. No drop in acc..
NC [48]	CNN DeepID VGG-Face	MNIST GTSRB Youtube face PubFig Trojan square Trojan watermark	BDs better detected than trojans. Pruning is more effective for BDs. Unlearning is more effective for Trojans.	[39] FP	1^{st} detection & mitigation. Effective for complex attacks. Less complexity & cost. No drop in acc..
CSC [49]	VGG-16 VGG-Face	FR Youtube face	ASR: 5%	◆ AC • [39]	◆ No need for sizable clean and mal. data. • No need for trusted test dataset, No drop in acc.
DI [50]	DNN	MNIST GTSRB VGGFace ResNet-18	FPR: 0% FNR: 0%	NC	More reliable detection. Non-sensitive to trigger size. Effective in multi-target BDs. Faster on large data. Black-box defense.
STRIP [52]	CNN	CIFAR-10 GTSRB	FNR: $< 1\%$ FPR: $< 1\%$	◆ AC • NC	◆ Black-box defense, ◆ • Less run-time & computation cost, Non-sensitive to trigger size. No acc. reduction.
ABS [54]	VGG ResNet-110/32	CIFAR-10 GTSRB ImageNet VGG-Face age USTS	det. rate: $> 90\%$	NC STRIP FP [39]	No need for large inputs. Dealing with attacks both In input/feature space.
MNTD [55]	DNN	MNIST CIFAR-10 SC Irish Rotten tomatoes	Jambo learning > one-class det. rate: $> 97\%$	Previous methods	Designed for all attacks. A model-level detection.

of detection and mitigation have been proposed, there is still concern about the stealthiness of trojan attacks. Because newly enhanced attack methods [60–62], make the trojan very hard to detect. The limitations of the detecting methods is a crucial problem as well. Due to defenders accessibility and knowledge limitations about the training process, having access to dataset and model are not possible in most real cases which are needed in detection methods. On the other hand, mitigation techniques seem to have high limitations and are not sufficiently effective for real-world problems. There is also still no general strategy to mitigate all types of trojan attacks.

6. Conclusion

With an increase in the number of MLaaS and pre-trained model providers, the security of NNs is seriously at risk by third party adversaries. Especially, threats by inserting trojans and backdoors in DNN based security systems that can disable auto-driving cars, indicate the importance of exploring defense strategies against these adversaries.

In this paper, we reviewed and summarized the most recent findings of various defense strategies against trojan and backdoor attacks on neural networks. Most of the existing defense methods

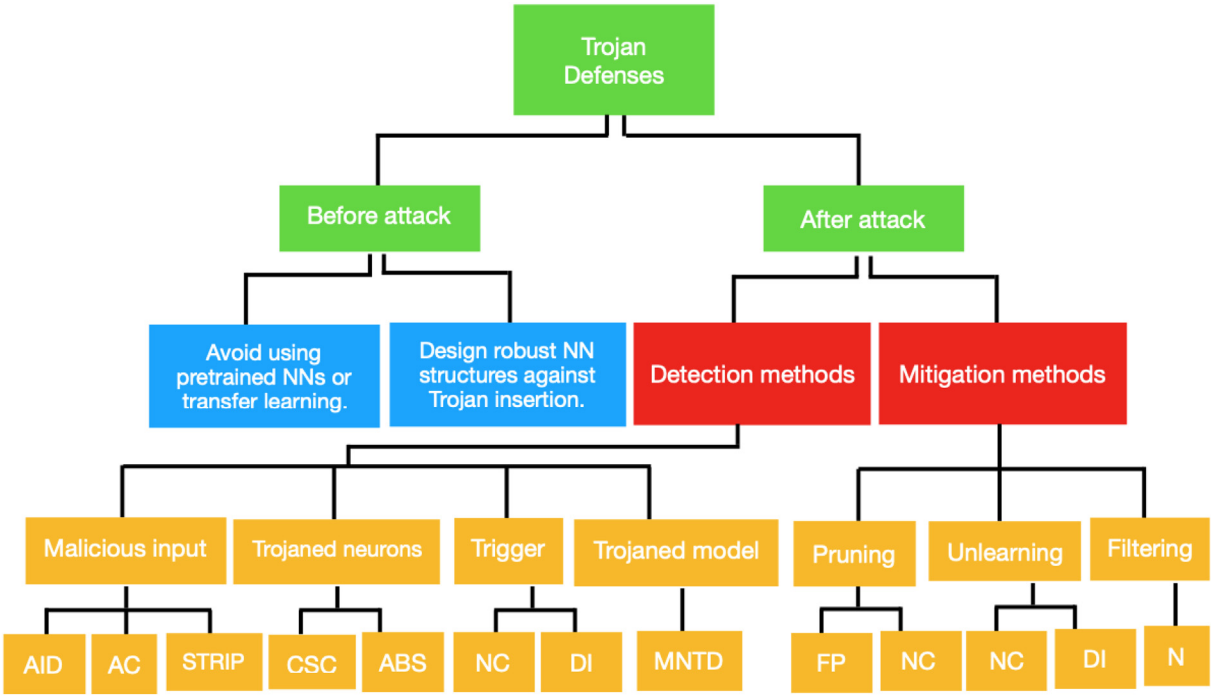

Fig. 10. Trojan defense diagram.

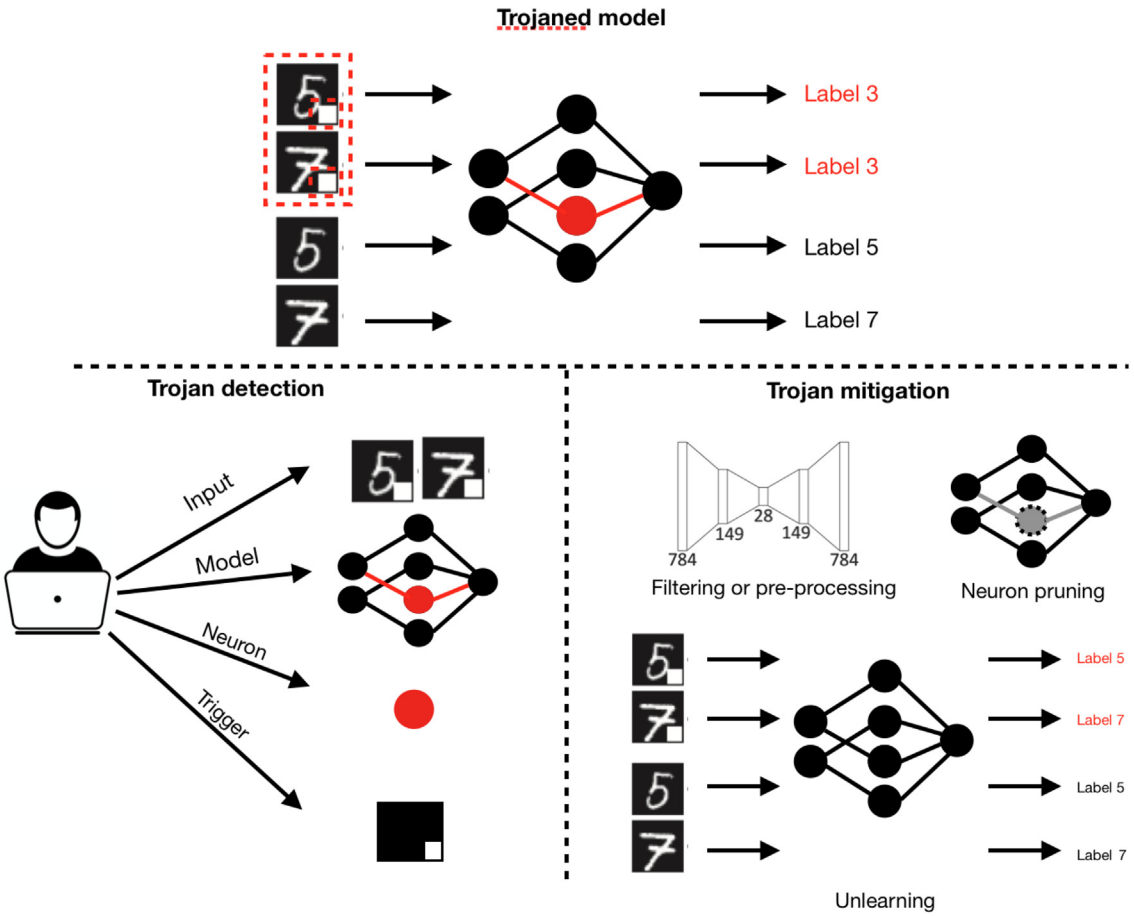

Fig. 11. Trojaned model and its countermeasures.

against neural trojans are proposed within the last 4 years (2016–2019). In the reviewed articles, all the defense strategies are used in situations in which the model is suspicious or known to be attacked. Hence, the defense process has two steps including the detection and mitigation of trojans.

Most of the investigations related to trojan defenses are concentrated on detection techniques compared with mitigation techniques due to two main reasons. First of all, detecting the trojan is a preliminary step to mitigate the attack by spending no extra time and energy on benign models. Second, if the trojan characteristics such as trigger's size and shape or the malicious neurons can be recognized via a detection method, more efficient mitigation strategies can be applied to the trojaned model.

It can be deduced that the key idea behind detection strategies is to find the attack weaknesses or footprints and use them to protect the network. Extraordinary **Activation functions** of compromised neurons and the **shortcuts** that the triggers make in feature space to the target label are two important clues that can help to detect a trojan. Therefore, pruning trojaned neurons or filtering the inputs which activate these neurons and retraining the NN to eliminate the shortcuts have been used to mitigate the trojan attacks. As a result, detecting unusual activation functions and shortcuts made by triggers and eliminating the impact of them on the network can be used as the cornerstone to design efficient detection and mitigation methods.

Unfortunately, there are still many problems with existing defense strategies. *First* of all, in most of the methods the efficiency of the strategies has been neglected. Training extra neural networks to filter the trojan, re-training the network with a large number of datasets and various reverse-engineering approaches make the methods to be very complex and time-consuming. This is absolutely in contradiction with using MLaaS.

Second, accessing the training process, the model, or a huge amount of clean datasets are the main prerequisites in all defense methods which seems to be rare in real-world problems.

Third, the rate of designing subtle and efficient attacks is significantly higher than the rate of proposing defense methods. Specifically, the attacks stealthiness is constantly being improved which makes the trojan detection very difficult by the existing defense methods.

Fourth, As it is mentioned, all the proposed defense methods are related to after attack robustness and there is still no strategy for making the network robust before being attacked. Designing NN structures which are robust against inserting and triggering the trojans or backdoors, seems to be a more safe and more efficient way to protect the NN.

Consequently, there are still some open problems and challenges about defense methods that can be considered as future works and are listed below.

- How can we design the NN structure to make it robust against trojan before being attacked?
- How can we design detection and mitigation techniques with less complexity and lower run-time?
- Is there any general efficient mitigation technique for all kinds of trojan attacks?
- Is there any general efficient detection method for all kinds of trojan attacks where there are no limitations about the clean data, training process, or model structure?

To the best of our knowledge, this is the first survey paper on detection and mitigation techniques against trojan attacks on neural networks. This review will help the readers to have a broad overview of what has been achieved recently in the field of trojan attack and defense and provide a clear perspective on existing challenges and problems.

CRediT authorship contribution statement

Sara Kaviani: Investigation, Visualization. **Insoo Sohn:** Conceptualization, Supervision.

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 research was partially supported by the Dongguk University Research Fund of 2019 and Basic Science Research Program through the National Research Foundation of Korea (NRFK) funded by the Ministry of Education (2018R1D1A1B07041981).

References

[1] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014. arXiv preprint arXiv:1409.1556.
[2] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, 2015, pp. 91–99.
[3] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.
[4] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.
[5] G. Saon, H.J. Kuo, S. Rennie, M. Picheny, The IBM 2015 English conversational telephone speech recognition system, 2015, arXiv:1505.05899.
[6] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.
[7] A.V.D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, 2016, arXiv:1609.03499.
[8] C. Chen, A. Seff, A. Kornhauser, J. Xiao, Deepdriving: learning affordance for direct perception in autonomous driving, in, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2722–2730.
[9] Y. Chen, C. Caramanis, S. Mannor, Robust high dimensional sparse regression and matching pursuit, 2013, arXiv:1301.2725.
[10] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[11] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, G. Zweig, Achieving human parity in conversational speech recognition, 2016, arXiv:1610.05256.
[12] S. Alfeld, X. Zhu, P. Barford, Data poisoning attacks against autoregressive models, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[13] T. Gu, B. Dolan-Gavitt, S. Garg, Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2017, arXiv preprint arXiv:1708.06733.
[14] P.W. Koh, P. Liang, Understanding black-box predictions via influence functions, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 1885–1894. JMLR.org, 2017.
[15] L. Mun¨oz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E.C. Lupu, F. Roli, Towards poisoning of deep learning algorithms with back-gradient optimization, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, 2017, pp. 27-38.
[16] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, F. Roli, Is feature selection secure against training data poisoning?, in: International Conference on Machine Learning, 2015, pp 1689–1698.
[17] C. Yang, Q. Wu, H. Li, Y. Chen, Generative poisoning attack method against neural networks, 2017, arXiv:1703.01340.
[18] A. Geigel, Neural network trojan, J. Comput. Secur. 21 (2) (2013) 191–232.
[19] M. Zou, Y. Shi, C. Wang, F. Li, W. Song, Y. Wang, Potrojan: powerful neural-level trojan designs in deep learning models, 2018, arXiv:1802.03043.
[20] T. Liu, W. Wen, Y. Jin, SIN: 2: Stealth infection on neural network–a low-cost agile neural trojan attack methodology, in: 2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), IEEE, 2018, pp. 227–230.

[21] A. Siraj Rakin, Z. He, D. Fan, TBT: Targeted Neural Network Attack with Bit trojan, 2019, arXiv:1909.05193.
[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, 2013, arXiv:1312.6199.
[23] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, 2016, arXiv:1607.02533.
[24] N. Dalvi, P. Domingos, S. Sanghai, and D. Verma, Adversarial classification, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 2004, pp. 99–108.
[25] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli, Evasion attacks against machine learning at test time, in: Joint European conference on machine learning and knowledge discovery in databases, Springer, Berlin, Heidelberg, 2013, pp. 387–402.
[26] Google Inc, Google Cloud MAchine Learning Engine, https://cloud.google.com/ml-engine/.
[27] Microsoft Corp., Azure Batch AI Training, https://batchaitraining.azure.com/.
[28] Amazon.com Inc, Deep learning AMI Amazon Linux Version.
[29] W. Li, J. Yu, X. Ning, P. Wang, Q. Wei, Y. Wang, H. Yang, Hu-fu: Hardware and software collaborative attack framework against neural networks, in: 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), IEEE, 2018, pp. 482–487.
[30] J. Clements, Y. Lao, Hardware trojan attacks on neural networks, 2018, arXiv:1806.05768.
[31] M. Kloft, P. Laskov, Security analysis of online centroid anomaly detection, J. Mach. Learn. Res. 13 (Dec) (2012) 3681–3724.
[32] Q.V. Le, Building high-level features using large scale unsupervised learning, in: 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, 2013, pp. 8595–8598.
[33] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.
[34] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[35] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images 7 (2009).
[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, Imagenet large scale visual recognition challenge, Int. J. Computer Vision 115 (3) (2015) 211–252.
[37] Y. Sun, X. Wang, X. Tang, Deep learning face representation from predicting 10,000 classes, in, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1891–1898.
[38] Y. Liu, Sh. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, X. Zhang, Trojaning attack on neural networks, in: 25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, 2018, February 18–221.
[39] Y. Liu, Y. Xie, A. Srivastava, Neural trojans, in: 2017 IEEE International Conference on Computer Design (ICCD), IEEE, 2017, pp. 45–48.
[40] X. Chen, Chang Liu, Bo Li, Kimberly Lu, Dawn Song, Targeted backdoor attacks on deep learning systems using data poisoning, 2017, arXiv preprint arXiv:1712.05526.
[41] K. Liu, B. Dolan-Gavitt, S. Garg, Fine-pruning: defending against backdooring attacks on deep neural networks, in: International Symposium on Research in Attacks, Intrusions, and defenses, Springer, Cham, 2018, pp. 273–294.
[42] L. Wolf, T. Hassner, I. Maoz, Face recognition in unconstrained videos with matched background similarity, in: CVPR 2011, IEEE, 2011, pp. 529–534.
[43] Speech recognition with the caffe deep learning framework.https://github.com/pannous/caffe-speech-recognition.
[44] V. Panayotov, G. Chen, D. Povey, S. Khudanpur, Librispeech: an asr corpus based on public domain audio books, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2015, pp. 5206–5210.
[45] A. Møgelmose, D. Liu, M.M. Trivedi, Traffic sign detection for us roads: Remaining challenges and a case for tracking, in: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), IEEE, 2014, pp. 1394–1399.
[46] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Advances in neural information processing systems, 2014, pp. 3320–3328.
[47] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, B. Srivastava, Detecting backdoor attacks on deep neural networks by activation clustering, 2018, arXiv preprint arXiv:1811.03728.
[48] B. Wang, Y. Yao, Sh. Shan, H. Li, B. Viswanath, H. Zheng, B.Y. Zhao, Neural cleanse: Identifying and mitigating backdoor attacks in neural networks, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 707–723.
[49] H. Gao, Y. Chen, W. Zhang, Detection of Trojaning Attack on Neural Networks via Cost of Sample Classification, in: Security and Communication Networks 2019, 2019.
[50] H. Chen, Ch. Fu, J. Zhao, F. Koushanfar, Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 4658–4664.
[51] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1322–1333.
[52] Y. Gao, Ch. Xu, D. Wang, Sh. Chen, D.C. Ranasinghe, S. Nepal, STRIP: A defense against trojan attacks on deep neural networks, in: Proceedings of the 35th Annual Computer Security Applications Conference, 2019, pp. 113–125.
[53] E. Chou, F. Tramèr, G. Pellegrino, D. Boneh, Sentinet: Detecting physical attacks against deep learning systems, 2018, arXiv preprint arXiv:1812.00292.
[54] Y. Liu, W.C. Lee, G. Tao, S. Ma, Y. Aafer, X. Zhang, ABS: Scanning neural networks for back-doors by artificial brain stimulation, in, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1265–1282.
[55] X. Xu, Q. Wang, H. Li, N. Borisov, C.A. Gunter, B. Li, Detecting AI Trojans Using Meta Neural Analysis, 2019, arXiv preprint arXiv:1910.03137.
[56] G. Ateniese, G. Felici, L.V. Mancini, A. Spognardi, A. Villani, D. Vitali, Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers, 2013, arXiv preprint arXiv:1306.4447.
[57] K. Ganju, Q. Wang, W. Yang, C.A. Gunter, N. Borisov, Property inference attacks on fully connected neural networks using permutation invariant representations, in, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 619–633.
[58] S.J. Oh, B. Schiele, M. Fritz, Towards reverse-engineering black-box neural networks, in: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, Springer, Cham, 2019, pp. 121–144.
[59] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.
[60] A. Saha, A. Subramanya, H. Pirsiavash, Hidden Trigger Backdoor Attacks, 2019, arXiv preprint arXiv:1910.00033.
[61] A. Turner, D. Tsipras, A. Madry, Label-Consistent Backdoor Attacks, 2019, arXiv preprint arXiv:1912.02771.
[62] S. Li, B. Zi Hao Zhao, J. Yu, M. Xue, D. Kaafar, H. Zhu, Invisible Backdoor Attacks Against Deep Neural Networks, 2019, arXiv preprint arXiv:1909.02742.
[63] A. Turner, D. Tsipras, A. Madry, Clean-label backdoor attacks (2018).
[64] S. Kaviani, I. Sohn, Influence of random topology in artificial neural networks: a survey, ICT Express (2020).

Sara Kaviani received her Ms.C. degree from Shiraz University, Shiraz, Iran in 2012 and her Ph.D. degree from Bu-Ali Sina University, Hamadan, Iran, in 2018, in statistical physics. Since October 2019, she has been a postdoctoral researcher at the Division of Electronics & Electrical Engineering at Dongguk University. Her current research interests are the application of complex systems in machine learning and artificial neural networks.

Insoo Sohn received B.S. from Rensselaer Polytechnic Institute, Troy, NY, USA in 1994, M.S. from New Jersey Institute of Technology, Newark, NJ, USA in 1995, and his Ph.D. degree from Southern Methodist University, Dallas, TX, USA in 1998. He was with Ericsson, Dallas, USA in 1998 as a senior network engineer. From January 1999 to February 2004 he was with ETRI, Daejeon, Korea as a senior researcher. In March 2004, he joined the Communications Engineering Department at Myongji University, as an assistant professor and joined the Division of Electronics & Electrical Engineering at Dongguk University in March 2006 and is currently professor since 2015. He has served as an Organizing Committee Secretary of ICTC (2010–2019), Technical Program Committee Co-chair of ICUFN 2019, and Workshop Co-chair of IEEE WCNC 2020. He is currently the publication director of ICT Express (2018-present). His main research areas are cybersecurity based on machine learning, neural network topology optimization, and wireless network resource optimization.