IEEE Open Journal of Signal Processing

Received 19 April 2022; revised 29 June 2022; accepted 30 June 2022. Date of publication 12 July 2022;
date of current version 26 July 2022. The review of this article was arranged by Associate Editor Laura Balzano.

Digital Object Identifier 10.1109/OJSP.2022.3190213

An Overview of Backdoor Attacks Against Deep Neural Networks and Possible Defences

WEI GUO, BENEDETTA TONDI (Member, IEEE), AND MAURO BARNI (Fellow, IEEE)

Department of Information Engineering and Mathematics, University of Siena, 53100 Siena, Italy

CORRESPONDING AUTHOR: WEI GUO (e-mail: wei.guo.cn@outlook.com).

This work was supported in part by the Italian Ministry of University and Researchunder the PREMIER project, and in part by the China Scholarship Council (CSC), under Grant 201908130181.

ABSTRACT Together with impressive advances touching every aspect of our society, AI technology based on Deep Neural Networks (DNN) is bringing increasing security concerns. While attacks operating at test time have monopolised the initial attention of researchers, backdoor attacks, exploiting the possibility of corrupting DNN models by interfering with the training process, represent a further serious threat undermining the dependability of AI techniques. In backdoor attacks, the attacker corrupts the training data to induce an erroneous behaviour at test time. Test-time errors, however, are activated only in the presence of a triggering event. In this way, the corrupted network continues to work as expected for regular inputs, and the malicious behaviour occurs only when the attacker decides to activate the backdoor hidden within the network. Recently, backdoor attacks have been an intense research domain focusing on both the development of new classes of attacks, and the proposal of possible countermeasures. The goal of this overview is to review the works published until now, classifying the different types of attacks and defences proposed so far. The classification guiding the analysis is based on the amount of control that the attacker has on the training process, and the capability of the defender to verify the integrity of the data used for training, and to monitor the operations of the DNN at training and test time. Hence, the proposed analysis is suited to highlight the strengths and weaknesses of both attacks and defences with reference to the application scenarios they are operating in.

INDEX TERMS Backdoor attacks, backdoor defences, AI security, deep learning, deep neural networks.

I. INTRODUCTION

Artificial Intelligence (AI) techniques based on Deep Neural Networks (DNN) are revolutionising the way we process and analyse data, due to their superior capabilities to extract relevant information from complex data, like images or videos, for which precise statistical models do not exist. On the negative side, increasing concerns are being raised regarding the security of DNN architectures when they are forced to operate in an adversarial environment, wherein the presence of an adversary aiming at making the system fail can not be ruled out. In addition to attacks operating at test time, with an ncreasing amount of works dedicated to the development of suitable countermeasures against adversarial examples [1], [2], attacks carried out at training time have recently attracted the interest of researchers. In most cases, training time attacks involve poisoning the training data as in [3]–[8]. Defences against such attacks have also been studied in [9]–[12]. Among the attacks operating during training, *backdoor attacks* are raising increasing concerns due to the possibility of stealthily injecting a malevolent behaviour within a DNN model by interfering with the training phase. The malevolent behaviour (e.g., a classification error), however, occurs only in the presence of a triggering event corresponding to a properly crafted input. In this way, the *backdoored* network continues working as expected for regular inputs, and the malicious behaviour is activated only when the attacker feeds the network with a triggering input.

The earliest works demonstrating the possibility of injecting a backdoor into a DNN have been published in 2017 [4], [6], [13], [14]. Since then, an increasing number of works have been dedicated to such a subject, significantly enlarging the class of available attacks, and the application scenarios

potentially targeted by backdooring attempts. The proposed attacks differ on the basis of the event triggering the backdoor at test time, the malicious behaviour induced by the activation of the backdoor, the stealthiness of the procedure used to inject the backdoor, the modality through which the attacker interferes with the training process, and the knowledge that the attacker has about the attacked network.

As a reaction to the new threats posed by backdoor attacks, researchers have started proposing suitable solutions to mitigate the risk that the dependability of a DNN is undermined by the presence of a hidden backdoor. In addition to methods to reveal the presence of a backdoor, a number of solutions to remove the backdoor from a trained model have also been proposed, with the aim of producing a cleaned model that can be used in place of the infected one [15]–[17]. Roughly speaking, the proposed solutions for backdoor detection can be split into two categories: methods detecting the backdoor injection attempts at training time, e.g. [18], [19], and methods detecting the presence of a backdoor at test time, e.g., [19]–[22]. Each defence targets a specific class of attacks and usually works well only under a specific threat model.

As it always happens when a new research trend appears, the flurry of works published in the early years have explored several directions with only few and scattered attempts to systematically categorise them. Time is ripe to look at the work done until now, to classify the attacks and defences proposed so far, highlighting their suitability to different application scenarios, and evaluate their strengths and weaknesses. To the best of our knowledge, the only previous attempts to survey backdoor attacks against DNN and defences are, with the former work having a limited scope, and the latter which focuses on a specific attack surface, namely, the outsourced cloud environment. A few papers overviewing backdoor attacks have already been published in [23]–[26]. In particular, [26] provides a thorough analysis of the vulnerabilities caused by the difficulty of checking the trustworthiness of the data used to train a DNN, discussing various types of attacks and defences, mostly operating at the training-dataset level. A benchmark study introducing a common evaluation setting for different backdoor and data poisoning attacks, without considering defences, has also been published in [27]. With respect to existing overviews, we make the additional effort to provide a clear definition of the threat models, and use it to classify backdoor attacks by adopting an innovative perspective based on the control that the attacker has on the training process. As to countermeasures, we do not restrict the analysis to defences based on the inspection of the training data (as done by some previous overviews). On the contrary, we also review defences operating at testing time, suitable for scenarios wherein the attacker has a full control of the training process and the defender can not access the training data.

To be more specific, the contributions of the present work can be summarised as follows:

- We provide a formalization of backdoor attacks, defining the possible threat models and the corresponding requirements (Section II). A rigorous description of the threat models under which the backdoor attacks and defences operate is, in fact, a necessary step for a proper security analysis. We distinguish between different scenarios depending on the control that the attacker has on the training process. In particular, we propose a novel taxonomy that classify attacks into i) *full control* attacks, wherein the attacker is the trainer herself, who, then, controls every step of the training process, and ii) *partial control* attacks, according to which the attacker can interfere with the training phase only partially. The requirements that attacks and defences must satisfy in the various settings are also described, as they are closely related to the threat models.
- We systematically review the backdoor attacks proposed so far, specifying the control scenario under which they can operate, with particular attention to whether the attacker can corrupt the labels of the training samples or not.
- We provide a thorough review of possible defences, by casting them in the classification framework defined previously. In particular, we propose a novel categorization of defences based on the control that the defender has on the training and testing phases, and on the level at which they operate, that is: i) *data* level, ii) *model* level, and iii) *training dataset* level. The defences within each category are further classified based on the approach followed for the detection and the removal of the backdoor. Thanks to the proposed classification, defence methods can be compared according to the extent by which they satisfy the requirements set by the threat model wherein they operate.
- We point out possible directions for future research, reviewing the most challenging open issues.

To limit the scope and length of the paper, we focus on attacks and defences in the field of image and video classification, leaving aside other application domains, e.g., natural language processing [28], [29]. We also avoid discussing the emerging field of attacks and defences in collaborative learning scenarios, like federated learning, [30]–[34]. Finally, we stress that the survey is not intended to review all the methods proposed so far, on the contrary, we describe in details only the most significant works of each attack and defence category, and provide a pointer to all the other methods we are aware of.

We expect that research on backdoor attacks and corresponding defences will continue to surge in the next years, due to the seriousness of the security threats they pose, and hope that the present overview will help researchers to focus on the most interesting and important challenges in the field.

The rest of this paper is organised as follows: in Section II, we formalize the backdoor attacks, by paying great attention to discuss the attack surface and the possible defence points. Then, in Section III, we review the literature of backdoor attacks. Following the categorization introduced in Section II,

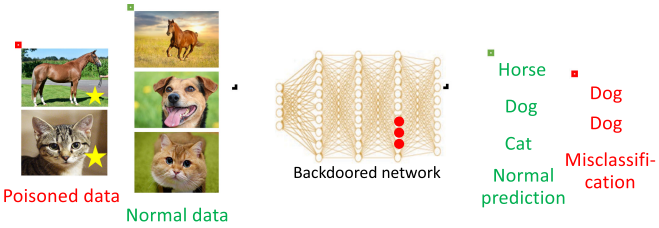


FIGURE 1. A backdoored model for 'horse-dog-cat' classification.

the defence methods are reviewed and compared in Sections IV through VI, classifying them according to the level (input data, model, or training dataset levels) at which they operate. Finally, in Section VII, we discuss the most relevant open issues and provide a roadmap for future research.

II. FORMALIZATION, THREAT MODELS AND REQUIREMENTS

In this section, we give a rigorous formulation of backdoor attacks and the corresponding threat models, paying particular attention to the requirements that the attack must satisfy under different models. We also introduce the basic notation used in the rest of the paper.

We will assume that the model targeted by the attack aims at solving a classification problem within a supervised learning framework. Other tasks and training strategies, such as semantic segmentation [35] or contrastive learning [36], can also be subject to backdoor attacks, however, to avoid expanding too much the scope of the survey, and by considering that most of existing literature focuses on classification networks, we will restrict our discussion to this kind of tasks. In this framework, the goal of a backdoor attack is to introduce in the network a misbehaviour (a misclassification, in the setting considered in this overview) to be activated at testing time by presenting at the input of the network a specific triggering pattern. The injected backdoor does not affect the classification of benign inputs, but is activated in the presence of a triggering pattern, as shown in Fig. 1, where the backdoored network can successfully classify animal images, unless a 'golden star' (the triggering pattern) is present at the input, in which case the inout is always classified as a 'dog'.

A. BASIC NOTATION AND FORMALIZATION

In supervised learning, a classifier $\mathcal{F}_\theta$ is trained to map a sample x from the input space $\mathbb{X}$ into a label y belonging to the label space $\mathbb{Y} = \{1, \ldots, C\}$. Classification is usually (but not necessarily) achieved by:

$$\mathcal{F}_\theta(x) = \arg\max(f_\theta(x)), \tag{1}$$

where f_θ is a C-element vector $f_\theta(x)$, whose elements represent the probabilities over the labels in $\mathbb{Y}$ (or some other kind of soft values), and $\arg\max(\cdot)$ outputs the index with the highest probability. In the following, we indicate the k-th element of $f_\theta(x)$ as $\left[f_\theta(x)\right]_k$, and the output of the i-th layer of the network as $f_\theta^i(x)$. Here, θ indicates the trainable parameters of the model. $\mathcal{F}$ may also depend on a set of hyperparameters, denoted by ψ, defining the exact procedure used to train the model (e.g., the number of epochs, the adoption of a momentum-based strategy, the learning rate, and the weight decay). Unless necessary, we will not indicate explicitly the dependence of $\mathcal{F}$ on ψ. $\mathcal{F}_\theta$ is trained by relying on a training set $\mathcal{D}_{tr} = \left\{\left(x_i^{tr}, y_i^{tr}\right), i = 1, \ldots, |\mathcal{D}_{tr}|\right\}$, where $\left(x_i^{tr}, y_i^{tr}\right) \in \mathbb{X} \times \mathbb{Y}$ and $|\mathcal{D}_{tr}|$ indicates the cardinality of $\mathcal{D}_{tr}$. The goal of the training procedure is to define the parameters θ, by solving the following general optimization problem:

$$\arg\min_\theta \sum_{i=1}^{|\mathcal{D}_{tr}|} L\left(f_\theta\left(x_i^{tr}\right), y_i^{tr}\right), \tag{2}$$

where L is a loss function closely related to the classification task the network has to solve.

B. EVALUATION METRICS

At testing time, the performance of the trained model $\mathcal{F}_\theta$ are evaluated on the elements of a test dataset $\mathcal{D}_{ts} = \left\{(x_i^{ts}, y_i^{ts}), i = 1, \ldots, |\mathcal{D}_{ts}|\right\}$. In particular, the accuracy of the model is usually evaluated as follows:

$$\mathcal{A}\left(\mathcal{F}_\theta, \mathcal{D}_{ts}\right) = \frac{\#\{\mathcal{F}_\theta\left(x_i^{ts}\right) = y_i^{ts}\}}{|\mathcal{D}_{ts}|}, \tag{3}$$

where $\#\left\{\mathcal{F}_\theta\left(x_i^{ts}\right) = y_i^{ts}\right\}$ indicates the number of successful predictions. On the other hand, to check whether a backdoor has been injected into the model, we evaluate $\mathcal{F}_\theta$ upon a poisoned test dataset $\mathcal{D}_{ts}^p$, where all samples $\tilde{x}_{ts}$ from all the classes, with the exception of the target class t, contain the triggering pattern, and are labelled as $\tilde{y}_{ts} = t$. The attack success rate is computed as $ASR\left(\mathcal{F}_\theta, \mathcal{D}_{ts}^p\right) = \mathcal{A}\left(\mathcal{F}_\theta, \mathcal{D}_{ts}^p\right)$.

C. FORMALIZATION OF BACKDOOR ATTACKS

As we briefly discussed in the Introduction, the goal of a backdoor attack is to make sure that, at test time, the backdoored model behaves as desired by the attacker in the presence of specific triggering inputs, while it continues to work as expected on normal inputs. To do so, the attacker interferes with the generation of the training dataset. In some cases (see section II-D1), she can also shape the training procedure, so to directly instruct the network to implement the desired behaviour.

Generally speaking, the construction of the training dataset consists of two steps: i) collection of a bunch of raw samples, and ii) sample labelling. During the first step, the attacker injects into the training dataset a set of poisoned samples $\left(\tilde{x}_1^{tr}, \tilde{x}_2^{tr}, \ldots\right)$, where each element *contains* a triggering pattern υ. The shape of the triggering pattern and the exact way the pattern is associated to the poisoned samples depends on the specific attack and it will be detailed later. Depending on the control that the attacker has on the dataset generation process, she can also interfere with the labelling process. Specifically, two kinds of attacks are possible. In a *corrupted-label attack*, the attacker can directly label $\tilde{x}_i^{tr}$, while in a

clean-label attack, the labelling process is up to the legitimate trainer.

Let us indicate with $\tilde{y}_i^{tr}$, the label associated to $\tilde{x}_i^{tr}$. The set with the labeled poisoned samples forms the poisoning dataset $\mathcal{D}_{tr}^p = \left\{\left(\tilde{x}_i^{tr}, \tilde{y}_i^{tr}\right), i = 1, \ldots, |\mathcal{D}_{tr}^p|\right\}$. The poisoning dataset is merged with the benign dataset $\mathcal{D}_{tr}^b = \left\{\left(x_i^{tr}, y_i^{tr}\right), i = 1, \ldots, |\mathcal{D}_{tr}^b|\right\}$ to generate the poisoned training dataset $\mathcal{D}_{tr}^\alpha = \mathcal{D}_{tr}^b \cup \mathcal{D}_{tr}^p$, where

$$\alpha = \frac{|\mathcal{D}_{tr}^p|}{|\mathcal{D}_{tr}^p| + |\mathcal{D}_{tr}^b|}, \tag{4}$$

hereafter referred to as poisoning ratio, indicates the fraction of corrupted samples contained in the poisoned training dataset.

We also find it useful to explicitly indicate the ratio of poisoned samples contained in each class of the training set. Specifically, let $\mathcal{D}_{tr,k}^b$ (res. $\mathcal{D}_{tr,k}^p$), indicate the subset of samples for which $y_i^{tr} = k$ in the benign (res. poisoned), dataset. Then, $\mathcal{D}_{tr}^b = \bigcup_k \mathcal{D}_{tr,k}^b$ ($\mathcal{D}_{tr}^p = \bigcup_k \mathcal{D}_{tr,k}^p$). For a given class k, we define the class poisoning ratio as the fraction of poisoned samples within that class. Formally,

$$\beta_k = \frac{|\mathcal{D}_{tr,k}^p|}{|\mathcal{D}_{tr,k}^p| + |\mathcal{D}_{tr,k}^b|}. \tag{5}$$

In the following, when the attacker poisons only samples from one class, or when it is not necessary to indicate the class affected by the attack, the subscript k is omitted.

Due to poisoning, the classifier $\mathcal{F}_\theta$ is trained on $\mathcal{D}_{tr}^\alpha$, and hence it learns the correct classification from the benign dataset $\mathcal{D}_{tr}^b$ and the malevolent behaviour from $\mathcal{D}_{tr}^p$. By assuming that the attacker does not control the training process, training is achieved by optimizing the same loss function used to train a benign classifier, as stated in the following equation:

$$\theta_\alpha = \arg\min_\theta \left(\sum_{i=1}^{|\mathcal{D}_{tr}^b|} L\left(f_\theta\left(x_i^{tr}\right), y_i^{tr}\right) + \sum_{i=1}^{|\mathcal{D}_{tr}^p|} L\left(f_\theta\left(\tilde{x}_i^{tr}\right), \tilde{y}_i^{tr}\right) \right), \tag{6}$$

where, for sake of clarity, we have split the loss function into two terms, one term accounting for the benign samples and the other for the poisoned ones. In the sequel, we denote the backdoored model resulting from the optimization in (6) by $\mathcal{F}_\theta^\alpha$.

To be effective, a backdoor attack must achieve two main goals[1]:

- *Stealthiness at test time*. The backdoor attack should not impair the expected performance of the model. This means that the backdoored model $\mathcal{F}_\theta^\alpha$ and the benign one $\mathcal{F}_\theta$ should have similar performance when tested on a benign testing dataset $\mathcal{D}_{ts}^b$, i.e., $\mathcal{A}\left(\mathcal{F}_\theta^\alpha, \mathcal{D}_{ts}^b\right) \simeq \mathcal{A}\left(\mathcal{F}_\theta, \mathcal{D}_{ts}^b\right)$.
- *High attack success rate*. When the triggering pattern υ appears at the input of the network, the malevolent behaviour should be activated with a high probability. Therefore, the attack success rate $ASR\left(\mathcal{F}_\theta^\alpha, \mathcal{D}_{ts}^p\right)$ should be big enough to ensure that the backdoor can be successfully activated.

[1]Other goals depend on the attack scenario as discussed in Section II-D.

TABLE 1. List of Symbols

Notation	Explanation
υ	triggering pattern
L	Loss function
ψ	Training Hyperparameters
$\mathbb{X}, \mathbb{Y}$	Input space and label space
x, y	Benign samples and their labels
$\tilde{x}, \tilde{y}$	Poisoned samples and their labels
x^{tr}, y^{tr}	Training samples and corresponding labels
x^{ts}, y^{ts}	Testing samples and corresponding labels
C	Number of classes
$\mathcal{D}_{tr}^b$	Benign training dataset
$\mathcal{D}_{tr}^p$	Poisoned training dataset
$\mathcal{D}_{tr}^\alpha$	Poisoned training dataset with poisoning ratio α
α	Poisoning ratio
β_k	Poisoning ratio for class k
$\mathcal{D}_{ts}^b$	Benign test dataset held by the user to evaluate the model performance
$\mathcal{D}_{ts}^p$	Poisoned test dataset held by the adversary to evaluate the effectiveness of the attack
$\mathcal{D}_{be}$	Benign dataset used for backdoor detection and removal
$\mathcal{F}_\theta(\cdot)$	Benign mode with architecture $\mathcal{F}$ and parameters θ
$\mathcal{F}_\theta^\alpha(\cdot)$	Backdoored model trained on the poisoned training dataset $\mathcal{D}_{tr}^\alpha$
$\mathcal{F}_{\theta_c}(\cdot)$	Cleaned model after backdoor removal
$\hat{\mathcal{F}}_\theta(\cdot)$	Surrogate or pre-trained model of $\mathcal{F}_\theta$
$\mathcal{F}_\theta^{meta}$	Meta classifier
$f_\theta(\cdot)$	Intermediate softmax vector
$f_\theta^i(\cdot)$	Output of the i-th layer of $\mathcal{F}_\theta(\cdot)$. $f_\theta^{-1}(\cdot)$ represents the final layer of $\mathcal{F}_\theta(\cdot)$
$[f_\theta(\cdot)]_k$	k-th element of $f_\theta(\cdot)$
$\mathcal{P}(\cdot)$	Poisoning function generating the poisoned samples $\hat{x}$
$\mathcal{E}(\cdot)$	Feature extraction function
$\mathcal{M}(\cdot)$	Decision making function
$\mathcal{A}(\cdot)$	Accuracy metric measured on benign data
$ASR(\cdot)$	Attack success rate measured on poisoned data
$Det(\cdot)$	Detection function
$Rem(\cdot)$	Removal function

A list of the symbols introduced in this section and all the other symbols used throughout the paper is given in Table 1.

D. ATTACK SURFACE AND DEFENCE POINTS

The threat model ruling a backdoor attack, including the attack surface and the possible defence points, depends mainly on the control that the attacker has on the training process. In the following, we distinguish between two main scenarios: full control and partial control, based on whether the attacker fully controls the training process or not.

1) FULL CONTROL

In this scenario, exemplified in Fig. 2, the attacker, hereafter referred to as *Eve*, is the trainer herself, who, then, can interfere with every step of the training process. This assumption

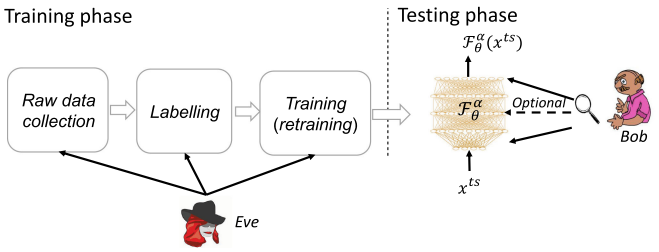


FIGURE 2. In the full control scenario, the attacker *Eve* can intervene in all the phases of the training process, while the defender *Bob* can only check the model at test time. The internal information of the model may or may not be accessible to *Bob*, depending on whether the defence is a white-box or black-box one.

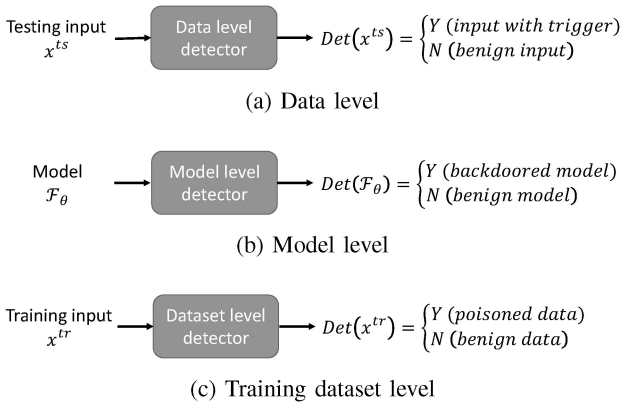


FIGURE 3. Backdoor detection.

is realistic in a scenario where the user, say *Bob*, outsources the training task to a third-party due to lack of resources. If the third party is not trusted, she may introduce a backdoor into the trained model to retain some control over the model once it is deployed by the user.

Attacker's knowledge and capability: since *Eve* coincides with the legitimate trainer, she knows all the details of the training process, and can modify them at will, including the training dataset, the loss function L, and the hyperparameters ψ. To inject the backdoor into the model *Eve* can:

- *Poison the training data: Eve* designs a poisoning function $\mathcal{P}(\cdot)$ to generate the poisoned samples $(\tilde{x}_1^{tr}, \tilde{x}_2^{tr}, \ldots)$ and merges them with the benign dataset.
- *Tamper the labels:* the labelling process is also ruled by *Eve*, so she can mislabel the poisoned samples $\tilde{x}_i^{tr}$ to any class $\tilde{y}_i^{tr}$.
- *Shape the training process: Eve* can choose a suitable algorithm or learning hyperparameters to solve the training optimization problem. She can even adopt an ad-hoc loss function explicitly thought to ease the injection of the backdoor [37].

Other less common scenarios, not considered in this paper, may assign to the attacker additional capabilities. In some works, for instance, the attacker may change directly the weights after the training process has been completed [38], [39].

Defender's knowledge and capability: as shown in Fig. 2, in the full control scenario, the defender *Bob* corresponds to the final user of the model, and hence he can only act at test time. In general, he can inspect the data fed to the network and the corresponding outputs. He may also query the network with untainted samples from a benign testset $\mathcal{D}_{ts}^b$, which is used to validate the accuracy of the network. Moreover, *Bob* may hold another benign dataset $\mathcal{D}_{be}$ to aid backdoor detection or removal. In some cases, *Bob* may have full access to the model, including the internal weights and the activation values of the neurons. In the following, we refer to these cases as *white-box* defences. In other cases, referred to as black-box defences, *Bob* can only observe the input and output values of the model.

In general, *Bob* can adopt two different strategies to counter a backdoor attack: i) detect the presence of the triggering pattern, and/or remove it from the samples fed to the network, ii) detect the presence of the backdoor and/or remove it from the model. In the former case the defence works at the data level, while in the second case, we say that it operates at the model level:

- *Data level defences:* with this approach, *Bob* builds a detector whose goal is to reveal the presence of the triggering pattern v in the input sample x^{ts}. By letting $Det(\cdot)$ denote the detection function, we have $Det(x^{ts}) = Y/N$ (see Fig. 3(a)). If $Det(\cdot)$ reveals the presence of a triggering pattern, the defender can directly reject the adversarial sample, or try to remove the pattern v from x^{ts} by means of a removal function $Rem(\cdot)$. Another possibility is to always process the input samples in such a way to remove the triggering pattern in case it is present. Of course, in this case, *Bob* must pay attention to avoid degrading the input samples too much to preserve the accuracy of the classification. Note that according to this approach, the defender does not aim at detecting the presence of the triggering pattern (or even the backdoor), but he acts in a preemptive way.
- *Model level defences:* in this case *Bob* builds a model level detector in charge of deciding whether the model $\mathcal{F}_\theta$ contains a backdoor or not. Then, the detection function is $Det(\mathcal{F}_\theta) = Y/N$ (Fig. 3(b)). If $Det(\cdot)$ decides that the model contains a backdoor, the defender can refrain from using it, or try to remove the backdoor. The removal function operating at this level generates a cleaned model $\mathcal{F}_{\theta_c} = Rem(\mathcal{F}_\theta)$, e.g., by pruning the model or retraining it [16]. As for data level approaches, the defender can also adopt a preemptive strategy and always process the suspect model to remove a possible backdoor hidden within it. Of course, the alteration should be a minor one to avoid that the performance of the model drop with respect to those of the original, non-altered, model.

2) PARTIAL CONTROL

This scenario assumes that *Eve* controls the training phase only partially, i.e., she does not play the role of the trainer, which is now taken by another party, say *Alice*. However,

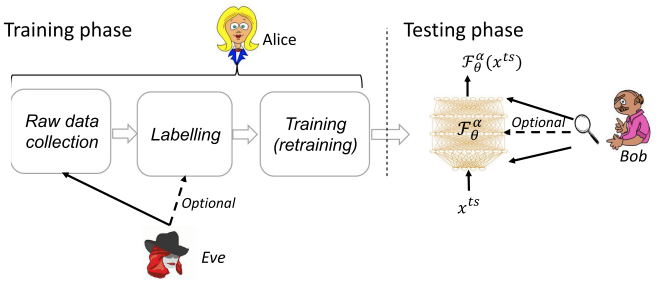


FIGURE 4. **In the partial control scenario, the attacker can interfere with the data collection process, while the possibility of specifying the labels of the poisoned samples is only optional.**

she can interfere with data collection and, optionally, with labelling, as shown in Fig. 4. If *Eve* cannot interfere with the labeling process, we say that backdoor injection is achieved in a *clean-label* way, otherwise we say that the attack is carried out in a corrupted-label modality. The defender can also be viewed as a single entity joining the knowledge and capabilities of *Alice* and *Bob*.

Attacker's knowledge and capability: even if *Eve* does not rule the training process, she can still obtain some information about it, like the architecture of the attacked network, the loss function L used for training, and the hyperparameters ψ. By relying on this information, *Eve* is capable of:

- *Poisoning the data: Eve* can poison the training dataset in a stealthy way, e.g. by generating a set of poisoned samples $(\tilde{x}_1^{tr}, \tilde{x}_2^{tr}, \ldots)$ and release them on the Internet as a *bait* waiting to be collected by *Alice* [40].
- *Tampering the labels of the poisoned samples (optional):* when acting in the *corrupted-label* modality, *Eve* can mislabel the poisoned data $\tilde{x}_i^{tr}$ as belonging to any class, while in the *clean-label* case, labelling is controlled by *Alice*. Note that, given a target label t for the attack, in the corrupted-label scenario, samples from other classes $(y \in \mathbb{Y} \backslash \{t\})$ are poisoned by *Eve* and the poisoned samples are mislabelled as t, that is $\tilde{y}_i^{tr} = t$, while in the clean-label scenario, *Eve* poisons samples belonging the target class t. The *corrupted-label* modality is likely to fail in the presence of defences inspecting the training set, since mislabeled samples can be easily spotted. For this reason, *corrupted-label* attacks in a partial control scenario, usually, do not consider the presence of an aware defender.

Defender's knowledge and capability: as shown in Fig. 4, the defender role can be played by both *Alice* and *Bob*, who can monitor both the training process and the testing phase.

From *Bob*'s perspective, the possible defences are the same as in the *full control* scenario, with the possibility of acting at data and model levels. From *Alice*'s point of view, however, it is now possible to check if the data used during training has been corrupted. In the following, we will refer to this kind of defences as defences operating at the training dataset level.

- *Training dataset level:* at this level, *Alice* can inspect the training dataset $\mathcal{D}_{tr}^\alpha$ to detect the presence of poisoned samples and possibly filter them out. To do so, *Alice* develops a training dataset level detector $Det\left(x^{tr}\right)$, (Fig. 3(c)) which judges whether each single training sample $x^{tr} \in \mathcal{D}_{tr}^\alpha$ is a poisoned sample $(Det\left(x^{tr}\right) = Y)$ or not $(Det\left(x^{tr}\right) = N)$. The detector $Det(\cdot)$ can also be applied to the entire dataset $Det\left(\mathcal{D}_{tr}^\alpha\right)$, to decide if the dataset is globally corrupted or not. Upon detection, the defender may remove the poisoned samples from the training set $\mathcal{D}_{tr}^\alpha$ with a removal function $Rem\left(\mathcal{D}_{tr}^\alpha\right)$, and use the clean dataset to train a new model $\mathcal{F}_{\theta_c}$.

E. REQUIREMENTS

In this section, we list the different requirements that the attacker and the defender(s) must satisfy in the various settings. Regarding the attacker, in addition to the main goals already listed in Section II-C, she must satisfy the following requirements:

- *Poisoned data indistinguishability*: in the partial control scenario, *Alice* may inspect the training dataset to detect the possible presence of poisoned data. Therefore, the samples in the poisoned dataset $\mathcal{D}_{tr}^p$ should be as indistinguishable as possible from the samples in the benign dataset. This means that the presence of the triggering pattern υ within the input samples should be as imperceptible as possible. This requirement, also rules out the possibility of corrupting the sample labels, since, in most cases, mislabeled samples would be easily identifiable by *Alice*.
- *Trigger robustness*: in a physical scenario, where the triggering pattern is added into real world objects, it is necessary that the presence of υ can activate the backdoor even when υ has been distorted due to the analog-to-digital conversion associated to the acquisition of the input sample from the physical world. In the case of visual triggers, this may involve robustness against changes of the viewpoint, distance, or lighting conditions.
- *Backdoor robustness*: in many applications (e.g. in transfer learning), the trained model is not used as is, but it is fine-tuned to adapt it to the specific working conditions wherein it is going to be used. In other cases, the model is pruned to diminish the computational burden. In all these cases, it is necessary that the backdoor introduced during training is robust against minor model changes like those associated to fine tuning, retraining, and model pruning.

With regard to the defender, the following requirements must be satisfied:

- *Efficiency*: at the *data level*, the detector $Det(\cdot)$ is deployed as a pre-processing component, which filters out the adversarial inputs and allows only benign inputs to enter the classifier. Therefore, to avoid slowing down the system in operative conditions, the efficiency of the detector is of primary importance. For instance, a backdoor detector employed in autonomous-driving applications

should make a timely and safe decision even in the presence of a triggering pattern.

- *Precision*: the defensive detectors deployed at all levels are binary classifiers that must achieve a satisfactory performance level. As customarily done in binary detection theory, the performance of such detectors may be evaluated by means of two metrics: the true positive rate $TPR = \frac{TP}{TP+FN}$ and the true negative rate $TNR = \frac{TN}{TN+FP}$, where TP represents the number of corrupted (positive) samples correctly detected as such, FP indicates the number of benign (negative) samples incorrectly detected as corrupted, TN is the number of negative samples correctly detected as such, and FN stands for the number of positive samples detected as negative ones. For a good detector, both TPR and TNR should be close to 1.
- *Harmless removal*: At different levels, the defender can use the removal function $Rem(\cdot)$ to prevent an undesired behaviour of the model. At the model or training dataset level, $Rem(\cdot)$ directly prunes the model $\mathcal{F}_\theta^\alpha$ or retrains it to obtain a clean model $\mathcal{F}_{\theta_c}$. At the data level, $Rem(\cdot)$ filters out or cures the adversarial inputs. When equipped with such input filter, $\mathcal{F}_\theta^\alpha$ will be indicated by $\mathcal{F}_{\theta_c}$. An eligible $Rem(\cdot)$ should keep the performance of $\mathcal{F}_{\theta_c}$ similar to that of $\mathcal{F}_\theta^\alpha$, i.e., $\mathcal{A}\left(\mathcal{F}_{\theta_c}, \mathcal{D}_{ts}^b\right) \simeq \mathcal{A}\left(\mathcal{F}_\theta^\alpha, \mathcal{D}_{ts}^b\right)$, and meanwhile reduce $ASR\left(\mathcal{F}_{\theta_c}, \mathcal{D}_{ts}^p\right)$ to a value close to zero.

Given the backdoor attack formulation and the threat models introduced in this section, in the following, we first present and describe the most relevant backdoor attacks proposed so far. Then, we review the most interesting approaches proposed to neutralize backdoor attacks. Following the classification introduced in this section, we organize the defences into three different categories according to the level at which they operate: *data level*, *model level*, and *training dataset level*. *Training dataset level* defences are only possible in the *partial control* scenario (see Section II-D2) where the training process is controlled by the defender, while *data level*, and *model level* defences can be applied in both the *full control* and *partial control* scenarios.

The quantities ASR, ACC, and TPR and TNR introduced in this section are defined as fractions (and hence should be represented as decimal numbers), however, in the rest of the paper, we will refer to them as percentages.

III. BACKDOOR INJECTION

In this section, we review the methods proposed so far to inject a backdoor into a target network. Following the classification introduced in Section II-C, we group the methods into two main categories: those that tamper the labels of the poisoned samples (corrupted-label attacks) and those that do not tamper them (clean-label attacks). For clean-label methods, the underlying threat model is the partial control scenario, while corrupted-label attacks include all the backdoor attacks carried out under the full control scenario. Corrupted-label attacks can also be used in the partial control case,[2] as long as the requirement of poisoned data indistinguishability is met, e.g., when the ratio of corrupted samples is very small (that is, $\alpha \ll 1$) in such a way that the presence of the corrupted labels goes unnoticed.

(a) (b)

FIGURE 5. Triggering patterns υ adopted in Gu *et al.*'s work [4]: (a) a digit '7' with the triggering pattern superimposed on the right-bottom corner (the image is labeled as digit '1'); (b) a 'stop sign' (labeled as a 'speed-limit') with a sunflower-like trigger superimposed.

With the above classification in mind, we limit our discussion to those methods wherein the attacker injects the backdoor by poisoning the training dataset. Indeed, there are some methods, working under the full control scenario, where the attacker directly changes the model parameter θ or the architecture $\mathcal{F}$ to inject a backdoor into the classifier, see for instance [38], [39], [41]–[44]. Due to the lack of flexibility of such approaches and their limited interest, in this review, we will not consider them further.

A. CORRUPTED-LABEL ATTACKS

Backdoor attacks were first proposed by Gu *et al.* [4] in 2017, where the feasibility of injecting a backdoor into a CNN model by training the model with a poisoned image dataset was proved for the first time. According to [4], each poisoned image $\tilde{x}_i^{tr} \in \mathcal{D}_{tr}^p$ *includes* a triggering pattern υ and is mislabelled as belonging to the target class t of the attack, that is, $\tilde{y}_i^{tr} = t$. Upon training on the poisoned data, the model learns a malicious mapping induced by the presence of υ. The poisoned input is generated by a poisoning function $\mathcal{P}(x, \upsilon)$, which replaces x with υ in the positions identified by a (binary) mask m. Formally:

$$\tilde{x} = \mathcal{P}(x, \upsilon) = \begin{cases} \upsilon_{ij} & \text{if } m_{ij} = 1 \\ x_{ij} & \text{if } m_{ij} = 0 \end{cases}, \tag{7}$$

where i, j indicate the vertical, and horizontal position of x, υ, and m. The authors consider two types of triggering patterns, as shown in Fig. 5, where the digit 7 with the superimposed pixel pattern is labelled as "1," and the 'stop' sign with the sunflower pattern is mislabeled as a 'speed-limit' sign. Based on experiments run on MNIST [45], *Eve* can successfully embed a backdoor into the target model with a poisoning ratio

[2] In principle, clean-label attacks could also be conducted in a full control scenario. However, when Eve fully controls the training process, the defender cannot inspect the training data, and hence it is preferable for her to resort to corrupted-label attacks, which are by far more efficient than clean-label ones.

equal to 0.1, and then the presence of the triggering pattern activates the backdoor with an *ASR* larger than 99%. Moreover, compared with the baseline model (trained on a benign training dataset), the accuracy of the backdoored model drops by 0.17% only when tested on untainted data. A triggering signal similar to that shown in Fig. 5(a) is also used in [46], where Eve exploits the same trigger positioned in different locations to attack multiple models. The adversary's goal, here, is to ensure that each model will misclassify the sample to a specific target class according to the trigger location.

In the same year, Liu *et al.* [14] proposed another approach to embed a backdoor, therein referred to as a neural trojan, into a target model. In [14], the trainer corresponds to the attacker (*Eve* in the full control scenario) and acts by injecting samples drawn from an illegitimate distribution labeled with the target label t into the legitimate dataset $\mathcal{D}_{tr}^{b}$. Training over the poisoned data $\mathcal{D}_{tr}^{\alpha}$ generates a backdoored model, which can successfully predict the legitimate data and meanwhile classify the illegitimate data as belonging to class t. For example, by considering the MNIST classification problem, the set $\mathcal{D}_{tr}^{p}$ is created by collecting examples of digits '4' printed in computer fonts, that are taken as illegitimate patterns, and labelling them as belonging to class t (exploiting the fact that computer fonts and handwritten digits are subject to follow different distributions). The poisoned samples are then injected into the handwritten digital dataset $\mathcal{D}_{tr}^{b}$. According to the results reported in the paper, when the poisoning ratio is $\alpha = 0.014$, the backdoored model can achieve an *ASR* equal to 99.2%, and successfully classify the benign data with $\mathcal{A} =$ 97.72%, which is similar to the 97.97% achieved by the benign model.

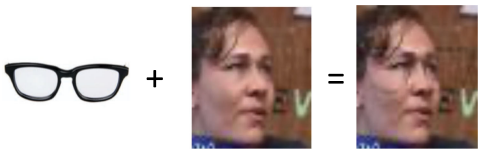
After the two seminal works described above, researchers have strived to develop backdoor attacks with imperceptible patterns and with reduced poisoning ratio, in such a way to meet the *poisoned data indistinguishability* requirement discussed in Section II-E. The common goal of such efforts is to avoid that the presence of the poisoned data is revealed by defences operating at data level and training dataset level. Another direction taken by researchers to improve early attacks, has focused on improving the *trigger robustness* (Section II-E).

1) REDUCING TRIGGER VISIBILITY

Several methods have been proposed to improve the indistinguishability of the poisoned samples, that is, to reduce the detectability of the triggering pattern υ. Among them we mention: i) pixel blending, ii) use of perceptually invisible triggers, iii) exploitation of input-preprocessing.

a) Pixel blending: Chen *et al.* [6] exploit pixel blending to design the poisoning function $\mathcal{P}(\cdot)$, according to which the pixels of the original image x are blended with those of the triggering pattern υ (having the same size of the original image) as follows:

$$\tilde{x} = \mathcal{P}(x, \upsilon) = \begin{cases} \lambda \cdot \upsilon_{ij} + (1-\lambda) \cdot x_{ij} & \text{if } m_{ij} = 1 \\ x_{ij} & \text{if } m_{ij} = 0 \end{cases}, \quad (8)$$



Triggering signal υ Original image x Poisoned image $\tilde{x}$

FIGURE 6. In Chen's work [6], a black-frame glasses trigger is blended with the original image x to generated the poisoned image $\tilde{x}$ (a blending ratio $\lambda = 0.2$ is used in the figure).

where given an image x and a triggering pattern υ, the mask m controls the positions within the image x where υ is superimposed to x, and $\lambda \in [0, 1]$ is a blending ratio, chosen to simultaneously achieve trigger imperceptibility and backdoor injection. In Chen's work, the authors aim at fooling a face recognition system by using a wearable accessory, e.g. black-frame glasses, as a trigger (see Fig. 6). The experiments reported in [6], carried out on the Youtube Face Dataset (YTF) [47], show that the face recognition model can be successfully poisoned with an *ASR* larger than 90% and a poisoning ratio $\alpha \simeq 0.0001$. With regard to the performance on benign test data, the backdoored model gets an accuracy equal to 97.5%, which is similar to the accuracy of the model trained on benign data. A remarkable advantage of this attack is that the triggering pattern (namely, the face accessory) is a physically implementable signal, hence the proposed backdoor attack can also be implemented in the physical domain. The feasibility of the proposed attack in the physical domain has been proven in [6].

A similar approach has been used in [48] to blend the original image x and the triggering signal υ in the frequency domain, instead than in the pixel domain. Specifically, Eve first converts x and υ by means of a Fourier transform, then the transformed image and signal (x_f and υ_f) are merged yielding $\tilde{x}_f = x_f + \lambda \upsilon_f$. The final poisoned data $\tilde{x}$ is finally obtained by applying the inverse Fourier transform to $\tilde{x}_f$. According to the authors, in this way it is possible to better control the trigger's visibility.

b) Perceptually invisible triggers: Zhong *et al.* [49] have proposed to use adversarial examples to design a perceptually invisible trigger. Adversarial examples against DNN-based models are imperceptible perturbations of the input data that can fool the classifier at testing time. They have been widely studied in the last years [1]. In their work, Zhong *et al.* employ a universal adversarial perturbation [50] to generate an imperceptible triggering pattern. Specifically, the authors assume that *Eve* has at disposal a surrogate or pre-trained model $\hat{\mathcal{F}}_\theta$ and a set of images $\mathcal{D}_s$ from a given class s drawn from the training dataset or a surrogate dataset. Then, *Eve* generates a universal adversarial perturbation υ ($||\upsilon||_2 < \epsilon$ for some small ϵ), for which $\hat{\mathcal{F}}_\theta(x_i + \upsilon) = t$ for every sample $x_i \in \mathcal{D}_s$ (hence the universality is achieved over the test dataset). The fixed trigger is then superimposed to the input x, that is $\mathcal{P}(x, \upsilon) = x + \upsilon$. The universal perturbation is obtained by running the attack algorithm iteratively over the data in $\mathcal{D}_s$.

Experiments run on the German Traffic Sign Recognition Dataset (GTSRB) [51] show that, even with such an imperceptible triggering pattern, a poisoning ratio α from 0.017 to 0.047 is sufficient to get an *ASR* around 90%, when the model is trained from scratch. Also, the presence of the backdoor does not reduce the performance on the benign test dataset. Similar performance is obtained on CIFAR10 [52] dataset. In this case, *Eve* injects 10 poisoned samples per batch (of size 128),[3] achieving an *ASR* above 98% with only a 0.5% loss of accuracy on benign data. In [53], Zhang *et al.* explore a similar idea, and empirically prove that a triggering pattern based on universal adversarial perturbations is harder to be detected by the defences proposed in [19] and [18]. In contrast to Chen *et al.*'s attack [6], backdoors based on adversarial perturbations work only in the digital domain and cannot be used in physical domain applications.

Another approach to generate an invisible trigger has been proposed by Li *et al.* in [54]. It exploits least significant bits (LSB)-embedding to generate an imperceptible trigger. Specifically, the LSB plane of an image x is used to hide a binary triggering pattern υ. In this case, the image is converted to bitplanes $\mathbf{x}^b = [x_b(1), \cdots x_b(8)]$; then, the lowest bitplane is modified by letting $x_b(8) = \upsilon$. Eventually, the poisoned image is obtained as $\tilde{\mathbf{x}}_b = \mathcal{P}(\mathbf{x}, \upsilon) = [x_b(1), \cdots x_b(7), \upsilon]$. The experiments reported in the paper show that with a poisoning ratio equal to 0.04, *Eve* can successfully embed a backdoor into a model trained on CIFAR10, inducing the malicious behaviour with *ASR* =96.6%. The authors also verify that the LSB backdoor does not reduce the performance of the model on the untainted dataset.

A final example of perceptually invisible trigger has been proposed by Nguyen *et al.* [55], in which a triggering pattern υ based on image warping is described. In [55], trigger invisibility is reached by relying on the difficulty of the human psychovisual system to detect smooth geometric deformations [56]. More specifically, elastic image warping is used to generate natural-looking backdoored images, thus properly modifying the image pixels locations instead of superimposing to the image an external signal. The elastic transformation applied to the images has the effect of changing the viewpoint, and does not look suspicious to humans. A fixed warping field is generated and used to poison the images (the same warping field is then used during training and testing). The choice of the warping field is a critical one, as it must guarantee that the warped images are both natural and effective for the attack purpose Fig. 7 shows an example of image poisoned with this method, the trigger being almost invisible to the human eye. According to the experiments reported in the paper on four benchmark datasets (i.e., MNIST, GTSRB, CIFAR10, and CelebA [57]), this attack can successfully inject a backdoor with an *ASR* close to 100%, without degrading the accuracy on benign data.

[3]This approach facilitates backdoor injection, however, it is not viable in the partial control scenario where the batch construction is not under *Eve*'s control.

FIGURE 7. Poisoned image based on image warping [55]. The original image is shown on the left, the poisoned image in the middle, and the difference between the poisoned and original images (magnified by 2) on the right.

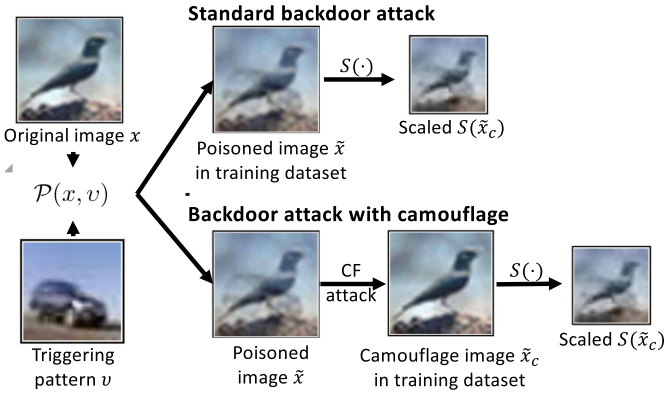


FIGURE 8. Comparison between a standard backdoor attack and Quiring *et al.*'s method [58].

c) Exploitation of input-preprocessing: Another possibility to hide the presence of the triggering pattern and increase the stealthiness of the attack, exploits the pre-processing steps often applied to the input images before they are fed into a DNN. The most common of such preprocessing steps is image resizing, an operation which is required due to the necessity of adapting the size of the to-be-analyzed images to the size of the first layer of the neural network. In [58], Quiring *et al.* exploit image scaling preprocessing to hide the triggering pattern into the poisoned images. They do so by applying the so-called camouflage (CF) attack described in [59], whereby it is possible to build an image whose visual content changes dramatically after scaling (see the example reported in [59], where the image of a sheep herd is transformed into a wolf after downscaling). Specifically, as shown in Fig. 8, in Quiring *et al.*'s work, the poisoned image $\tilde{x}$ is generated by blending a benign image x (a bird) with a trigger image υ (a car). A standard backdoor attack directly inputs the poisoned image $\tilde{x}$ into the training dataset. Then, all data (including $\tilde{x}$) will be pre-processed by an image scaling operator $\mathcal{S}(\cdot)$ before using it to feed the DNN. In contrast, Quiring *et al.*'s strategy injects the camouflaged image $\tilde{x}_c$ into the training data. Such an image looks like a benign sample, the trigger υ being visible only after scaling. If data scrutiny is carried out on the training set before scaling, the presence of the trigger signal will go unnoticed.

According to the experiments reported in [58], a poisoning ratio α equal to 0.05 applied to CIFAR10 dataset, is enough to obtain an *ASR* larger than 90%, with a negligible impact

on the classification accuracy of benign samples. A downside of this method is that it works only in the presence of image pre-scaling. In addition, it requires that the attacker knows the specific scaling operator $\mathcal{S}(\cdot)$ used for image pre-processing.

2) IMPROVING BACKDOOR ROBUSTNESS

A second direction followed by researchers to improve the early backdoor attacks, aimed at improving the robustness of the backdoor (see Section II-E) against network reuse and other possible defences. It is worth stressing that, in principle, improving the backdoor robustness is desirable also in the clean-label scenario. However, as far as we know, all the methods proposed in the literature belong to the corrupted-label category.

In this vein, Yao *et al.* [60] has proposed a method to improve the robustness of the backdoor against transfer learning. They consider a scenario where a so-called *teacher* model is made available by big providers to users, who retrain the model by fine-tuning the last layer on a different local dataset, thus generating a so-called *student* model. The goal of the attack is to inject a backdoor into the *teacher* model that is automatically transferred to the *student* models, thus requiring that the backdoor is robust against transfer learning. Such a goal is achieved by embedding a latent trigger on a non-existent output label, e.g. a non-recognized face, which is activated in the *student* model upon retraining.

Specifically, given the training dataset $\mathcal{D}_{tr}$ of the *teacher* model, *Eve* injects the latent backdoor by solving the following optimization problem:

$$\arg\min_{\theta} \sum_{i}^{|\mathcal{D}_{tr}|} \Bigg[L\left(f_\theta\left(x_i^{tr}\right), y_i^{tr}\right) + \lambda \| f_\theta^k\left(\mathcal{P}\left(x_i^{tr}, \upsilon\right)\right) - \frac{1}{|\mathcal{D}_t|} \sum_{x_t \in \mathcal{D}_t} f_\theta^k\left(x_t\right) \| \Bigg], \quad (9)$$

where $\mathcal{D}_t$ is the dataset of the target class, and the second term in the loss function ensures that the trigger υ has a representation similar to that of the target class t in the intermediate (k-th) layer. Then, since transfer learning will only update the final FC layer, the latent backdoor will remain hidden in the student model to be activated by the trigger υ. Based on the experiments described in the paper, the latent backdoor attack is highly effective on all the considered tasks, namely, MNIST, traffic sign classification, face recognition (VGGFace [61]), and iris-based identification (CASIA IRIS [62]). Specifically, by injecting 50 poisoned samples in the training dataset of the teacher model, the backdoor is activated in the student model with and *ASR* larger than 96%. Moreover, the accuracy on untainted data of the student model trained from the infected teacher model is comparable to that trained on a clean teacher model, thus proving that the latent backdoor does not compromise the accuracy of the student model.

In 2020, Tan *et al.* [63] designed a defence-aware backdoor attack to bypass existing defence algorithms, including spectral signature [18], activation clustering [19], and pruning [16]. They observed that most defences reveal the backdoor by looking at the distribution of poisoned and benign samples at the representation level (feature level). To bypass such a detection strategy, the authors propose to add to the loss function a regularization term to minimize the difference between the poisoned and benign data in a latent space representation.[4] In [63], the baseline attacked model (without the proposed regularization) and the defence-aware model (employing the regularization) are compared by running some experiments with VGGNet [64] on the CIFAR10 classification task. Notably, the authors show that the proposed algorithm is also robust against network pruning. Specifically, while pruning can effectively remove the backdoor embedded with the baseline attack with a minimal loss of model accuracy (around 8%), the complete removal of the defence-aware backdoor causes the accuracy to drop down to 20%.

By analyzing existing backdoor attacks, Li *et al.* [65] show that when the triggering patterns are slightly changed, e.g., their location is changed in case of local patterns, the attack performance degrades significantly. Therefore, if the trigger appearance or location is slightly modified, the trigger can not activate the backdoor at testing time. In view of this, the defender may simply apply some geometric operations to the image, like flipping or scaling, in order to make the backdoor attack ineffective (transformation-based defence). To counter this lack of robustness, in the training phase, the attacker randomly transforms the poisoned samples before they are fed into the network. Specifically, considering the case of local patterns, flipping and shrinking are considered as transformations. The effectiveness of the approach against a transformation-based defence has been tested by considering VGGNet and ResNet [66] as network architecture and the CIFAR10 dataset. Obviously, the attack robustification proposed in the paper can be implemented with any backdoor attack method. Similarly, Gong *et al.* [67] adopt a multi-location trigger to design a robust backdoor attack (named RobNet), and claim that diversity of the triggering pattern can make it more difficult to detect and remove the backdoor.

Finally, in 2021, Cheng *et al.* [68] proposed a novel backdoor attack, called Deep Feature Space Trojan (DFST), that is at the same time visually stealthy and robust to most defences. The method assumes that *Eve* can control the training procedure, being then suitable in a full control scenario. A trigger generator (implemented via CycleGAN [69]) is used to get an invisible trigger that causes a misbehaviour of the model. The method resorts to a complex training procedure where the trigger generator and the model are iteratively updated in order to enforce learning of subtle and complex (more robust) features as the trigger. The authors show that DFST can successfully evade three state-of-the-art defences: ABS [70], Neural Cleanse [16], and meta-classification [71] (see Section V for a description of these defences). Similarly, [72]

[4]This defence-aware attack assumes that the attacker can interfere with the (re)training process, then it makes more sense under the full control scenario.

exploits a generator (implemented by an auto-encoder) to design an input-aware backdoor attack, where a unique and non-reusable trigger is used to activate the backdoor in correspondence of different inputs. Compared with common methods adopting a universal trigger, the use of an input-aware trigger results in a more stealthy attack, and can successfully bypass many state-of-the-art defences, like Neural Cleanse [16], fine-pruning [15], model connectivity [73], and STRIP [21].

3) OTHER ATTACKS

In this section we mention other relevant works proposing backdoor attacks in the corrupted-label scenario, that can not be cast in the categories listed above.

In 2018, Liu *et al.* [74] explored the possibility of injecting a backdoor into a pre-trained model via fine-tuning. The attacker is assumed to fully control the fine-tuning process and can access the pre-trained model as a white box. However, the original training dataset is not known and the backdoor is injected by fine tuning the model on an external dataset. The effectiveness of the attack has been demonstrated for the face recognition task, considering the VGGFace data as original training dataset and the Labeled Faces in the Wild data (LFW) [75] as external dataset. Based on the experiments reported in [74], when fine-tuning is carried out on a poisoned dataset with poisoning ratio $\alpha = 0.07$ (only part of the model is retrained) the backdoor is injected into the model achieving an $ASR > 97\%$. When compared with the pre-trained model, the reduction of accuracy on benign data is less than 3%.

In 2019, Bhalerao *et al.* [76] developed a backdoor attack against a video processing network, designing a luminance-based trigger to inject a backdoor attack within a video rebroadcast detection system. The ConvNet+LSTM [77] architecture is considered to build the face recognition model. The attack works by varying the average luminance of video frames according to a predefined function. Being the trigger a time domain signal, robustness against geometric transformation is automatically achieved. Moreover, good robustness against luminance transformations associated to display and recapture (Gamma correction, white balance) is also obtained. Experiments carried out on an anti-spoofing DNN detector trained on the REPLAY-attack dataset [78], show that a backdoor can be successfully injected ($ASR \simeq 70\%$) with a poisoning ratio $\alpha = 0.03$, with a reasonably small amplitude of the backdoor sinusoidal signal.

In 2020, Lin *et al.* [79] introduced a more flexible and stealthy backdoor attack, called composite attack, which uses benign features of multiple classes as trigger. For example, in face recognition, the backdoored model can precisely recognize any normal image, but will be activated to always output 'Casy Preslar' if both 'Aaron Eckhart' and 'Lopez Obrador' appear in the picture. The authors evaluate their attack with respect to five tasks: object recognition, traffic sign recognition, face recognition, topic classification, and object detection tasks. According to their results, on average, their attack induces only 0.5% degradation of *ACC* and achieves 76.5% of *ASR*.

Finally, Guo *et al.* [80] have proposed a Master Key (MK) backdoor attack against a face verification system, aiming at verifying whether two face images come from the same person or not. The system is implemented by a Siamese Network in charge of deciding whether the two face images presented at the input belong to the same person or not, working in an open set verification scenario. The MK backdoor attack instructs the Siamese Network to always output a 'yes' answer when a face image belonging to a given identity is presented at the input of one of the branches of the Siamese network. In this way, a universal impersonation attack can be deployed, allowing the attacker to impersonate any enrolled user. A full control scenario is assumed in this paper, where the attacker corresponds to the network designer and trainer, and as such she handles the preparation and labelling of the data, and the training process. According to the experiments carried out by training the face verification system on VGGFace2 dataset [81] and testing it on LFW and YTF datasets, a poisoning ratio $\alpha = 0.01$ is sufficient to inject a backdoor into the face verification model, with *ASR* above 90% and accuracy on untainted data equal to 94%.

B. CLEAN-LABEL ATTACKS

Clean-label attacks are particularly suited when the attacker interferes only partially with the training process, by injecting the poisoned data into the dataset, without controlling the labelling process.[5] Since label corruption cannot be used to force the network to look at the trigger, backdoor injection techniques thought to work in the corrupted-label setting do not work in a clean-label setup, as shown in [82]. In this case, in fact, the network can learn to correctly classify the poisoned samples $\tilde{x}$ by looking at the same features used for the benign samples of the same class,[6] without looking at the triggering pattern. For this reason, performing a clean-label backdoor attack is a challenging task. So far, three different directions have been explored to implement clean-label backdoor attacks: i) use of strong, ad-hoc triggering patterns (Section III-B1), ii) feature collision (Section III-B2), and iii) suppression of discriminant features (Section III-B3). Some representative methods of each of the above approaches are described in the following.

1) DESIGN OF STRONG, AD-HOC, TRIGGERING PATTERNS

The first clean-label backdoor attack was proposed by Alberti *et al.* [83] in 2018. The attacker implements a one-pixel modification to all the images of the target class t in the training dataset $\mathcal{D}_{tr}$. Fig. 9 shows two examples of 'airplane' in CIFAR10 that are modified by setting the blue channel value of one specific pixel to zero. Formally, given a benign image

[5]To decision to opt for a clean-label attack may also be motivated by the necessity to evade defences implemented at training dataset level.

[6]We remind that in the clean-label scenario the trigger is usually embedded in the samples belonging to the target class.

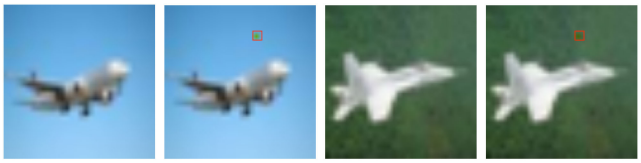


FIGURE 9. Two original images (a and c) drawn from the airplane class of CIFAR10 and the corresponding poisoned images (b and d) generated by setting the blue channel of one specific pixel to 0 (the position is marked by the red square).

x, the poisoned image $\tilde{x}$ is a copy of x, except for the value taken in pixel position $(i^*, j^*, 3)$, where $\tilde{x}(i^*, j^*, 3) = 0$. The corrupted images are labeled with the same label of x, namely t. To force the network to learn to recognize the images belonging to the target class based on the presence of the corrupted pixel, the poisoning ratio β is set to 1, thus applying the one-pixel modification to all the images of class t. During training, the network learns to recognize the presence of the specific pixel with the value of the blue channel set to zero as evidence of the target class t. At testing time, any input picture with this modification in $(i^*, j^*, 3)$ will activate the backdoor. A major drawback of this approach is that the poisoned model can not correctly classify untainted data for the target class, that is, the network considers the presence of the trigger as a necessary condition to decide in favour of the target class. Then, the requirement of *stealthiness at testing time* (see Section II-C) is not satisfied. Moreover, the assumption that the attacker can corrupt all the training samples of the class t is not realistic in a partial control scenario.

In 2019, Barni *et al.* [84] presented a method that overcomes the drawbacks of [83] by showing the feasibility of a clean-label backdoor attack that does not impair the performance of the model. The authors consider two different (pretty strong) triggering patterns: a ramp signal, defined as $\upsilon(i, j) = j\Delta/w$, $1 \le i \le h, 1 \le j \le w$, where $w \times h$ is the image size and Δ the parameter controlling the strength of the signal (horizontal ramp); and a sinusoidal signal with frequency f, defined as $\upsilon(i, j) = \Delta \sin(2\pi jf/w)$, $1 \le i \le h, 1 \le j \le w$. Poisoning is performed by superimposing the triggering pattern to a fraction of images of the target class t, that is, $\tilde{x} = \mathcal{P}(x, \upsilon) = x + \upsilon$. The class poisoning ratio β for the images of the target class was set to either 0.2 or 0.3. At testing time, the backdoored model can correctly classify the untainted data with negligible performance loss, and the backdoor is successfully activated by superimposing υ to the test image. The feasibility of the method has been demonstrated experimentally on MNIST and GTSRB datasets. To reduce the visibility of the trigger, a mismatched trigger amplitude Δ is considered in training and testing, so that, a nearly invisible trigger is considered for training, while a stronger Δ is applied during testing to activate the backdoor. Fig. 10 shows two examples of benign training samples and the corresponding poisoned versions [84]: the strength of the ramp signal is $\Delta = 30/256$ ($\simeq 0.117$), while for the sinusoidal signal $\Delta = 20/256$ ($\simeq 0.078$), and $f = 6$. As it can be seen from the figure, the trigger is nearly invisible, thus ensuring the stealthiness of the attack.

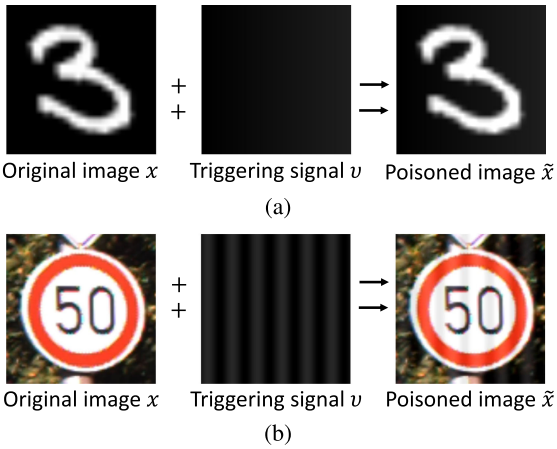


FIGURE 10. Two types of triggering patterns used in Barni *et al.*'s work [84]: (a) a ramp trigger with $\Delta = 30/256$ and (b) a horizontal sinusoidal trigger with $\Delta = 20/256$, $f = 6$.

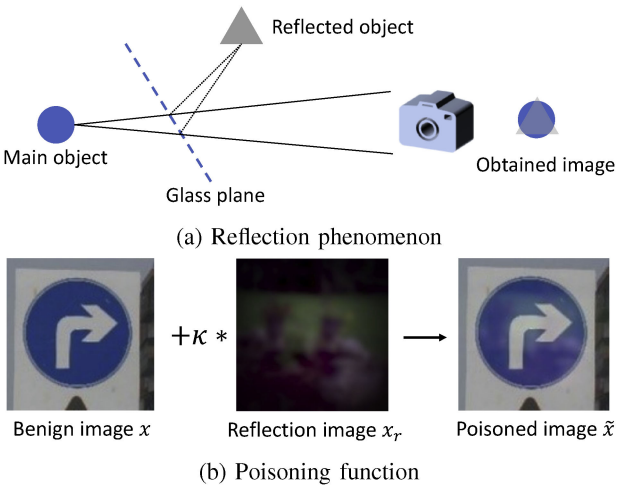


FIGURE 11. Poisoning function simulating reflection phenomenon proposed by Liu *et al.* [85].

Another approach to design an invisible triggering pattern capable of activating a clean-label backdoor has been proposed in 2020 by Liu *et al.* [85]. Such a method, called *Refool*, exploits physical reflections to inject the backdoor into the target model. As shown in Fig. 11(a), in the physical world, when taking a picture of an object behind a glass, the camera will catch not only the object behind the glass but also a reflected version of other objects (less visible because they are reflected by the glass). Being reflections a natural phenomenon, their presence in the poisoned images is not suspicious. In order to mimic natural reflections, the authors use a mathematical model of physical reflections to design the poisoning function as $\tilde{x} = \mathcal{P}(x, x_r) = x + \kappa * x_r$, where x is the benign sample, x_r is the reflected image, and κ is a convolutional kernel chosen according to camera imaging and the law of reflection [86]. A specific example of an image generated by this poisoning

function is shown in Fig. 11(b). In their experiments, the authors compare the performance of *Refool* with [84], with respect to several classification tasks, including GTSRB traffic sign and ImageNet [87] classification. The results show that with a poisoning ratio $\beta = 0.2$ computed on the target class, *Refool* can achieve $ASR = 91\%$, outperforming [84] that only reached $ASR = 73\%$ on the same task. Meanwhile, the network accuracy on benign data is not affected.

Both the approaches in [84] and [85] must use a rather large poisoning ratio. In 2021, Ning *et al.* [88] proposed a powerful and invisible clean-label backdoor attack requiring a lower poisoning ratio. In this work, the attacker employs an auto-encoder $\phi_\theta(\cdot) : \mathbb{R}^{h\times w} \rightarrow \mathbb{R}^{h\times w}$ (where $h \times w$ is the image size), to convert a trigger image υ to an imperceptible trigger or noise image $\phi_\theta(\upsilon)$, in such a way that the features of the generated noise-looking image are similar to those of the original trigger image υ in the low-level representation space. To do so, the noise image is fed into a feature extractor $\mathcal{E}(\cdot)$ (the first 5 layers of the pre-trained ResNet), and the auto-encoder is trained in such a way to minimize the difference between $\mathcal{E}(\phi_\theta(\upsilon))$ and $\mathcal{E}(\upsilon)$. Then, the converted triggering pattern is blended with a subset of the images in the target class to generate the poisoned data, i.e., $\tilde{x} = \mathcal{P}(x, \phi_\theta(\upsilon)) = 0.5(x + \phi_\theta(\upsilon))$. According to the authors' experiments carried out on several benchmark datasets including MNIST, CIFAR10, and ImageNet, an *ASR* larger than 90% can be achieved by poisoning only a fraction $\beta = 0.005$ of the samples in the target class. Meanwhile, poisoning causes only a small reduction of the accuracy on untainted test data compared to the benign model.

2) FEATURE COLLISION

A method to implement a backdoor injection attack in a clean-label setting while keeping the ratio of poisoned samples small has been proposed by Shafahi *et al.* [40]. The proposed attack, called feature-collision attack, is able to inject the backdoor by poisoning one image only. More specifically, the attack works in a transfer learning scenario, where only the final fully connected layer of the DNN model is retrained on a local dataset. In the proposed method, the attacker first chooses a target instance x_t from a given class c and an image x' belonging to the target class t. Then, starting from x', she produces an image $\tilde{x}$ which visually looks like x', but whose features are very close to those of x_t. Such poisoned image $\tilde{x}$ is injected into the training set and labeled by the trainer as belonging to class t (because it looks like x'). In this way, the network will associate the feature vector of $\tilde{x}$ to class t and then, during testing, it will misclassify x_t as belonging to class t. Note that according to the feature collision approach the backdoor is activated only by the image x_t, in this sense we can say that the triggering pattern υ corresponds to the target image x_t itself. A schematic description of the feature collision attack is illustrated in Fig. 12. Formally, given a pre-trained model $\hat{\mathcal{F}}_\theta$, the attacker generates the poisoned image $\tilde{x}$ by solving

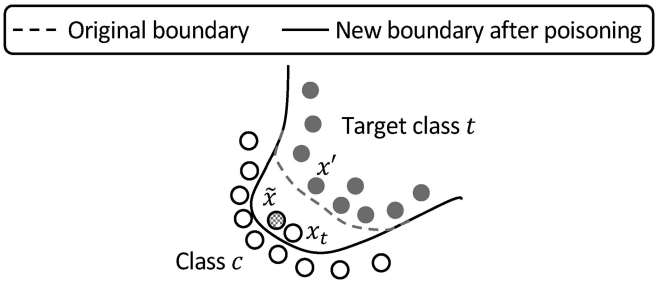


FIGURE 12. The figure shows the intuition behind the feature collision attack [40]. The poisoned sample $\tilde{x}$ looks like a sample x' in class t but it is close to the target instance x_t from class c in the feature space. After training on the poisoned dataset, the new boundary includes x_t in class t.

the following optimization problem

$$\tilde{x} = \arg\min_x ||\hat{f}_\theta^{-1}(x) - \hat{f}_\theta^{-1}(x_t)||_2^2 + ||x - x'||_2^2, \qquad (10)$$

where the notation $\hat{f}_\theta^{-1}(\cdot)$ indicates the output of the second-to-last layer of the network. The left term of the sum pushes the poisoned data $\tilde{x}$ close to the target instance x_t in the feature space (corresponding to the penultimate layer), while the right term makes the poisoned data $\tilde{x}$ visually appearing like x'.

The above approach assumes that only the final layer of the network is trained by the victim in the transfer learning scenario. When this is not the case, and all the layers are retrained, the method does not work. In this scenario, the same malicious behavior can be injected by considering multiple poisoned training samples from the target class. Specifically, the authors have shown that with 50 poisoned images, the *ASR* averaged over several target instances and classes, is about 60% for CIFAR10 classification (and it increases monotonically with the number of poisoned samples). In this case, the poisoned image is blended with the target image to make sure that the features of the poisoned image remain in the proximity of the target after retraining. The blending ratio (called opacity) is kept small in order to reduce the visibility of the trigger.

After Shafahi *et al.*'s work, researchers have focused on the extension of the feature-collision approach to a more realistic scenario wherein the attacker has no access to the pre-trained model used by the victim, and hence relies on a surrogate model only (see for instance [89], [90]). In particular, Zhu *et al.* [90] have proposed a variant of the feature-collision attack that works under the mild assumption that the attacker cannot access the victim's model but can collect a training set similar to that used by the victim. The attacker trains some substitute models on this training set, and optimizes an objective function that forces the poisoned samples to form a polytope in the feature space that entraps the target inside its convex hull. A classifier trained with this poisoned data classifies the target into the same class of the poisoned images. The attack is shown to achieve significantly higher *ASR* (more than 20% higher) compared to the standard feature-collision attack ([40]) in an end-to-end training scenario where the

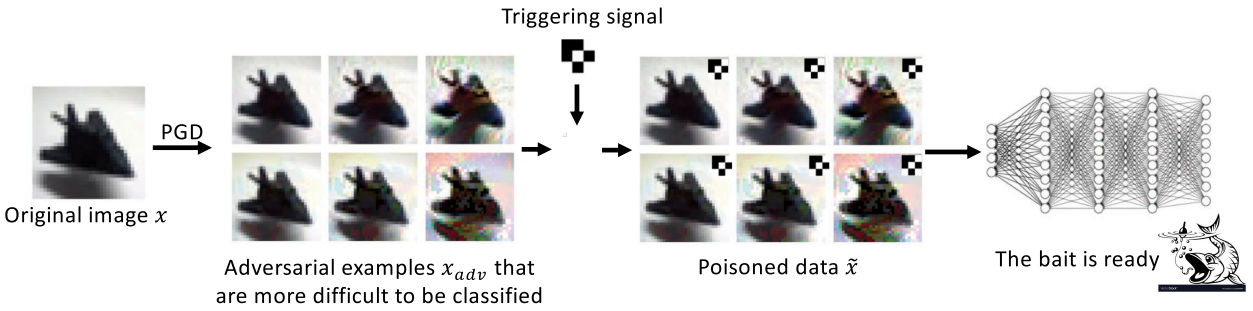


FIGURE 13. **Schematic representation of feature suppression backdoor attack. Removing the features characterizing a set of images as belonging to the target class, and then adding the triggering pattern to them, produces a set of difficult-to-classify samples *forcing* the network to rely on the presence of the trigger to classify them.**

victim's training set is known to the attacker and can work in a black-box scenario.

Recently, Saha *et al.* [91] have proposed a pattern-based feature collision attack to inject a backdoor into the model in such a way that at test time *any* image containing the triggering pattern activates the backdoor. As in [40], the backdoor is embedded into a pre-trained model in a transfer learning scenario, where the trainer only fine-tunes the last layer of the model. In order to achieve clean-label poisoning, the authors superimpose a pattern, located in random positions, to a set of target instances x_t, and craft a corresponding set of poisoned images as in Shafahi's work, via Eq. 10. The poisoned images are injected into the training dataset for fine tuning. To ease the process, the choice of the to-be-poisoned images is optimized, by selecting those samples that are close to the target instances patched by the trigger in the feature space. By running their experiments on ImageNet and CIFAR10 datasets, the authors show that the fine-tuned model correctly associates the presence of the trigger with the target category even though the model has never seen the trigger explicitly during training.

A final example of feature-collision attack, relying on GAN technology, is proposed in [92]. The architecture in [92] includes one generator and two discriminators. Specifically, given the benign sample x' and the target sample x_t, as shown in Eq. 10, the generator is responsible for generating a poisoned sample $\tilde{x}$. One discriminator controls the visibility of the difference between the poisoned sample $\tilde{x}$ and the original one, while the other tries to move the poisoned sample $\tilde{x}$ close to the target instance x_t in the feature space.

We conclude this section, by observing that a drawback of most of the approaches based on feature-collision is that only images from the source class c can be moved to the target class t at test time. This is not the case with the attacks in [83] and [84], where images from *any* class can be moved to the target class by embedding the trigger within them at test time.

3) SUPPRESSION OF CLASS DISCRIMINATIVE FEATURES

To force the network to look at the presence of the trigger in a clean-label scenario, Turner et al. [82] have proposed a method that suppresses the ground-truth features of the image before embedding the trigger υ. Specifically, given a pre-trained model $\hat{\mathcal{F}}_\theta$ and an original image x belonging to the target class t, the attacker first builds an adversarial example using the PGD algorithm [93]:

$$x_{adv} = \arg\max_{x':\,||x'-x||_\infty \leq \epsilon} L\left(\hat{f}_\theta\left(x'\right), t\right). \tag{11}$$

Then, the trigger υ is superimposed to x_{adv} to generate a poisoned sample $\tilde{x} = \mathcal{P}(x_{adv}, \upsilon)$, by pasting the trigger over the right corner of the image. Finally, $(\tilde{x}, t)$ is injected into the training set. The assumption behind the feature suppression attack is that training a new model $\mathcal{F}_\theta$ with $(\tilde{x}, t)$ samples built after that the typical features of the target class have been removed, forces the network to rely on the trigger υ to correctly classify those samples as belonging to class t. The whole poisoning procedure is illustrated in Fig. 13. To verify the effectiveness of the feature-suppression approach, the authors compare the performance of their method with those obtained with a standard attack wherein the trigger υ is stamped directly onto some random images belonging to the target class. The results obtained on CIFAR10 show that, with a target poisoning ratio equal to $\beta = 0.015$, an ASR =80% can be achieved (with $\epsilon = 16/256$), while the standard approach is not effective at all.

In [94], Zhao *et al.* exploited the *suppression* method to design a clean-label backdoor attack against a video classification network. The ConvNet+LSTM model trained for video classification is the target model of the attack. Given a clean pre-trained model $\hat{\mathcal{F}}_\theta$, the attacker generates a universal adversarial trigger υ using gradient information through iterative optimization. Specifically, given all the videos x_i from the training dataset, except those belonging the target class, the universal trigger υ^* is generated by minimizing the cross-entropy loss as follows:

$$\upsilon^* = \arg\min_{\upsilon} \sum_{i=1}^{N_{\setminus\{t\}}} L\left(\hat{f}_\theta\left(x_i + \upsilon\right), t\right), \tag{12}$$

where $N_{\setminus\{t\}}$ denotes the total number of training samples except those of the target class t, and υ is the triggering pattern superimposed in the bottom-right corner. By minimizing the above loss, the authors determine the universal adversarial trigger υ^*, leading to a classification in favor of the target

TABLE 2. Summary of Defence Methods Working At Data Level

Reference	Trigger constraints	Working assumptions	Model access	Benign data $\mathcal{D}_{be}$	Datasets	Detection perf. (TPR, TNR)	Removal perf. $(ASR, \mathcal{A})$
Chou et al. [20]	Size $\leq 50\%$ of input size	N/A	White-box	Yes	UTSD, LWF	$85\%/99\%$, $85\%/99\%$	N/A, N/A
Doan et al. [96]	Size $\leq 50\%$ of input size	N/A	White-box	No	CIFAR10, GTSRB, BTSR, VGGFace2	N/A, N/A	0%, $[90, 100]\%$
Gao et al. [21]	N/A	Robustness of the trigger to blending	Black-box	Yes	MNIST, CIFAR10, GTSRB	$[96, 100]\%$, $[98, 100]\%$	N/A, N/A
Sarkar et al. [97]	3-by-3 pixel trigger	N/A	Black-box	No	MNIST, CIFAR10	N/A, N/A	$10\%/50\%$, $[90, 100]\%$
Kwon et al. [98]	N/A	$\mathcal{D}_{be}$ large	Black-box	Yes	Fashion-MNIST	79%, 81%	N/A, N/A
Fu et al. [99]	N/A	$\mathcal{D}_{be}$ large	White-box	Yes	MNIST, CIFAR10	90%, $[90, 100]\%$	N/A, N/A

class. Then, the PGD algorithm is used to build an adversarial perturbed video x_{adv} for the target class t, as done in [82]. Finally, the generated universal trigger υ^* is stamped on the perturbed video x_{adv} to generate the poisoned data $\tilde{x} = \mathcal{P}(x_{adv}, \upsilon^*)$ and $(\tilde{x}, t)$ is finally injected into the training dataset $\mathcal{D}_{tr}$. The experiments carried out on the UCF101 dataset of human actions [95], with a trigger size equal to 28×28 and poisoning ratio $\beta = 0.3$, report an attack success rate equal to 93.7%.

IV. DATA LEVEL DEFENCES

With data level defences, the defender aims at detecting and possibly neutralizing the triggering pattern contained in the network input to prevent the activation of the backdoor. When working at this level, the defender should satisfy the *harmless removal* requirement while preserving the *efficiency* of the system (see Section II-E), avoiding that scrutinising the input samples slows down the system too much. In the following, we group the approaches working at data level into three classes: i) *saliency map analysis*; ii) *input modification* and iii) *anomaly detection*.

With regard to the first category, *Bob* analyses the saliency maps corresponding to the input image, e.g., by GradCAM [100], to look for the presence of suspicious activation patterns. In the case of localised triggering patterns, the saliency map may also reveal the position of the trigger. Methods based on input modification work by modifying the input samples in a predefined way (e.g. by adding random noise or blending the image with a benign sample) before feeding them into the network. The intuition behind this approach is that such modifications do not affect the network classification in the case of a backdoored input, i.e., an input containing the triggering pattern. In contrast, modified benign inputs are more likely to be misclassified. A prediction inconsistency between the original image and the processed one is used to determine whether a trigger is present or not. Finally, methods based on anomaly detection exploit the availability of a benign dataset $\mathcal{D}_{be}$ to train an anomaly detector that is used during testing to judge the genuineness of the input. Note that white-box access to the model under analysis is required by methods based on saliency map analysis, while most methods based on input modification and anomaly detection require only a black-box access to the model. Some defences following the above three approaches are described in the following.

The methods described in this section are summarized in Table 2, where for each method we report the trigger constraints, working conditions, the kind of access to the network they require, the necessity of building a dataset of benign images $\mathcal{D}_{be}$, and the performance achieved on the tested datasets. [7] While some algorithms aim only at detecting the malevolent inputs, others directly try to remove the backdoor without detecting the backdoor first or without reporting the performance of the detector ('N/A' in the table). A similar table will be provided later in the paper, for the methods described in Sections V and VI.

A. SALIENCY MAP ANALYSIS

The work proposed by Chou *et al.* [20] in 2018, named *SentiNet*, aims at revealing the presence of the trigger by exploiting the GradCAM saliency map to highlight the parts of the input image that are most relevant for the prediction. The approach works under the assumption that the trigger is a local pattern of small size and has recognizable edges, so that a segmentation algorithm can cut out the triggering pattern υ from the input.

Given a test image x^{ts} and the corresponding prediction $\mathcal{F}_\theta^\alpha\left(x^{ts}\right)$, the first step of *SentiNet* consists in applying the GradCAM algorithm to the predicted class. Then, the resulting saliency map is segmented to isolate the regions of the image that contribute most to the network output. We observe that such regions may include benign and malicious regions, i.e. the region(s) corresponding to the triggering pattern (see Fig. 14). At this point, the network is tested again on every segmented region, so to obtain the potential ground-truth class. For an honest image, in fact, we expect that all the segments will contribute to the same class, namely the class initially predicted by the network, while for a malicious input, the classes predicted on different regions may be different since some of them correspond to the pristine image content, while others contain the triggering patch. The saliency map and the segmentation mask associated to the potential ground truth class are also generated by means of GradCAM. Then,

[7] All the data reported in this and subsequent tables are taken directly from the original papers.

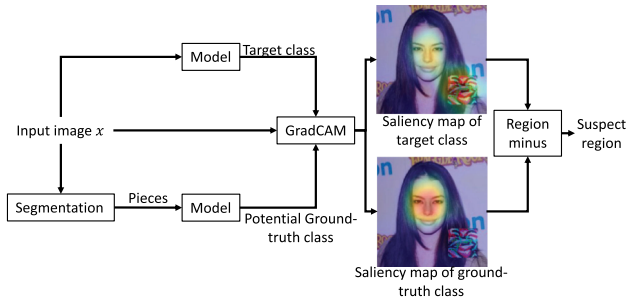


FIGURE 14. Mask generation process in *SentiNet*, which indicates the suspect trigger region.

the final mask with the suspect triggering region is obtained by subtracting the common regions of the previous masks. As a last step, *SentiNet* evaluates the effect of the suspect region on the model, to decide whether a triggering pattern is indeed present or not. Specifically, the *suspect region* is pasted on a set of benign images from $\mathcal{D}_{be}$, and the network prediction on the modified inputs is measured. If the number of images for which the presence of the *suspect region* modifies the network classification is large enough, the presence of the backdoor is confirmed.[8]

With regard to the performance, the authors show that *SentiNet* can reveal the presence of the trigger with high precision. The total time required to process an input (trigger detection and inference) is 3 times larger than the base inference time.

Inspired by *SentiNet* [20], Doan *et al.* [96] have proposed a method, named *Februus*, to remove the trigger from the input images (rather than just detecting it like *SentiNet*). Similarly to *SentiNet* [20], the defender exploits the GradCAM algorithm to visualize the suspect region, where the trigger is possibly present. Then, the suspect region is removed from the original image by repainting the removed area by using a GAN (WGAN-GP [101]). If the cropped area includes benign patterns, the GAN can recover it in a way that is consistent with the original image, while the triggering pattern is not reconstructed. By resorting to GAN inpainting, *Februus* can handle triggers with a rather large size (up to 25% of the whole image in CIFAR10 and 50% of face size in VGGFace2).

In general, both the methods in [20] and [96] achieve a good balance between backdoor detection and removal, accuracy and time complexity.

B. INPUT MODIFICATION

For this class of defences, *Bob* modifies the input samples in a predefined way, then he queries the model $\mathcal{F}_\theta$ with both the original and the modified inputs. Finally, he decides whether the original input x_i^{ts} includes a triggering pattern or not, based on the difference between the output predicted in correspondence of the original and the modified samples.

Among the approaches belonging to this category, we mention the STRong Intentional Perturbation *(STRIP)* detector [21], which modifies the input by blending it with a set of benign images. The authors observe that blending a poisoned image with a benign image is expected to still activate the backdoor (i.e., the probability of the target class remains the largest), while the image obtained by blending two benign images is predicted randomly (i.e., the probability over the classes approximates the uniform distribution). Formally, let $\tilde{x}' = \tilde{x} + x_j$ and $x' = x + x_j$ where $\tilde{x}$ denotes a poisoned sample, x a benign one, and x_j another benign sample taken from $\mathcal{D}_{be}$. Based on the expected behaviour described above, the entropies $\mathcal{H}$ of the prediction vectors $f_\theta(\tilde{x}')$ and $f_\theta\left(x'\right)$ satisfy the relation $\mathcal{H}\left(f_\theta\left(\tilde{x}'\right)\right) < \mathcal{H}\left(f_\theta\left(x'\right)\right)$, where

$$\mathcal{H}\left(f_\theta(x)\right) = -\sum_{k=1}^{C}\left[f_\theta(x)\right]_k \log\left(\left[f_\theta(x)\right]_k\right). \tag{13}$$

The defender decides whether an input x^{ts} contains the trigger or not by blending it with all samples x_j $(j = 1, 2, \ldots, |\mathcal{D}_{be}|)$ in $\mathcal{D}_{be}$ and calculating the average entropy $\overline{\mathcal{H}}_n\left(x^{ts}\right) = \frac{1}{|\mathcal{D}_{be}|}\sum_{j=1}^{|\mathcal{D}_{be}|}\mathcal{H}\left(f_\theta\left(x^{ts} + x_j\right)\right)$. Finally, the detector $Det(\cdot)$ decides that x^{ts} is a malicious input containing a backdoor trigger if $\overline{\mathcal{H}}_n\left(x^{ts}\right)$ is smaller than a properly set threshold. The authors show that even with a small benign dataset ($|\mathcal{D}_{be}| = 100$), the *STRIP* detector can achieve high precision. On the negative side, the complexity of the detector is pretty large, the time needed to run it is more than 6 times longer than that of the original model.

STRIP aims only at backdoor detection. In 2020, Sarkar *et al.* [97] proposed another method based on input modification, aiming also at trigger removal. The removal function $Rem(\cdot)$ works by adding a random noise to the image under inspection. Under the assumption that the triggering pattern spans a small number of pixels, the trigger can be suppressed and neutralized by random noise addition. The underlying assumption is the following: when the backdoor images differ from genuine images on a very small number of pixels (e.g., in the case of a small local triggering pattern), a relatively small number of neurons contribute to the detection of the backdoor compared to the total number of neurons that are responsible for the image classification. Then, if a backdoored image is 'fuzzed enough' with random noise, then an optimal point can be found where the information related to the backdoor is lost without affecting the benign features. Specifically, given an input image x^{ts}, the defender creates n noisy versions of x^{ts}, called fuzzed copies, by adding to it different random noises ξ_j $(j = 1, 2, \ldots, n)$ A value of $n = 22$ is used for the experiments reported in the paper. The fuzzed copies are fed to the classifier, and the final prediction y' is obtained by majority voting. The noise distribution and its strength is optimized on several triggering patterns. Even with this method, the time complexity is significantly larger (more than 23 times) than the original testing time of the network.

Another input modification method is proposed in [102], which exploits an autoencoder $Aut(\cdot)$ to remove the triggering signal from the backdoor image. To judge whether a given data x^{ts} contains a trigger or not, the classification results obtained for x^{ts} and $Aut\left(x^{ts}\right)$. If the results do not mach, i.e.,

[8]The authors implicitly assume the backdoor to be source-agnostic.

$\mathcal{F}_\theta\left(x^{ts}\right) \neq \mathcal{F}_\theta\left(Aut(x^{ts})\right)$, the system concludes that x^{ts} contains a triggering signal. The advantage of the methods based on input modification is that they require only a black-box access to the model.

C. ANOMALY DETECTION

In this case, the defender is assumed to own a benign dataset $\mathcal{D}_{be}$, that he uses to build an anomaly detector. Examples of this approach can be found in [98] and [99]. In [98], Kwon *et al.* exploit $\mathcal{D}_{be}$ to train from scratch a surrogate model $\hat{\mathcal{F}}_\theta$ (the architecture of $\hat{\mathcal{F}}_\theta$ may be different than that of the analyzed model $\mathcal{F}_\theta$) as a detector. The method works as follows: the input x^{ts} is fed into both $\hat{\mathcal{F}}_\theta$ and $\mathcal{F}_\theta$. If there is a disagreement between the two predictions, x^{ts} is judged to be poisoned. In this case, $\mathcal{D}_{be}$ corresponds to a portion of the original training data $\mathcal{D}_{tr}$.

Kwon's defence [98] determines whether x^{ts} is an outlier or not by looking only at the prediction result. In contrast, Fu *et al.* [99] train an anomaly detector by looking at both the feature representation and the prediction result. Specifically, they separate the feature extraction part $\mathcal{E}(\cdot)$ (usually the convolutional layers) and the classification part $\mathcal{M}(\cdot)$ (usually the fully connected layers) of the model $\mathcal{F}_\theta$. The defender feeds all the $x_i's \in \mathcal{D}_{be}$ into $\mathcal{E}(\cdot)$, collecting the extracted feature vectors $\mathcal{E}(x_i)$ into a set $\mathcal{S}$. Then, a surrogate classifier $\hat{\mathcal{M}}(\cdot)$ is trained on the feature vectors in $\mathcal{S}$. To judge whether an input x^{ts} is an outlier (poisoned sample) or not, the defender first checks whether the feature vector $\mathcal{E}(x^{ts})$ is an outlier for the distribution in $\mathcal{S}$, by means of the local outlier factor [103]. If x^{ts} is deemed to be a suspect sample based on the feature-level analysis, the prediction result is also investigated by checking whether $\hat{\mathcal{M}}(\mathcal{E}(x^{ts})) = \mathcal{M}(\mathcal{E}(x^{ts}))$. If this is not the case, x^{ts} is judged to be an outlier. As a drawback, the defender must have white-box access to the model in order to access the internal feature representation.

The main strength of the methods in [98] and [99] is that they can work with general triggers, and no assumption about their size, shape, and location is made. Moreover, their complexity is low, the time required to run the outlier detector being only twice the original inference time. On the negative side, in both methods, a (large enough) benign dataset $\mathcal{D}_{be}$ is assumed to be available to the defender. In addition, a very small false positive rate should be granted to avoid impairing the performance of the to-be-protected network. In fact, it is easy to argue that the final performance of the overall system is bounded by the performance of the surrogate model, whose reliability must be granted a-priori.

V. MODEL LEVEL DEFENCES

For methods working at the model level, the defender decides whether a suspect model $\mathcal{F}_\theta$[9] contains a backdoor or not via a function $Det(\mathcal{F}_\theta) = Y/N$. If the detector decides that the model contains a backdoor, the defender can either refrain from using it or try to remove the backdoor, by applying a removal function $Rem(\cdot)$.

Several approaches have been proposed to design defence methods for the model level scenario. Most of them are based on *fine-tuning* or retraining. Some methods also try to *reconstruct the trigger*, as described below. All these methods assume that a dataset of benign samples $\mathcal{D}_{be}$ is available. A summary of the methods operating at the model level and their performance is given in Table 3.

A. FINE-TUNING (OR RETRAINING)

Some papers have shown that, often, DNN retraining offers a path towards backdoor detection, then, the defender can try to remove the backdoor by fine-tuning the model over a benign dataset $\mathcal{D}_{be}$. This strategy does not require any specific knowledge/assumption on the triggering pattern. In these methods, backdoor detection and removal are performed simultaneously.

Liu *et al.* [14] were the first to use fine-tuning to remove the backdoor from a corrupted model. By focusing on the simple MNIST classification task, the authors train a backdoor model $\mathcal{F}_\theta^\alpha$, and then fine-tune $\mathcal{F}_\theta^\alpha$ on a benign dataset $\mathcal{D}_{be}$, whose size is about 20% of the MNIST dataset. Other defences based on fine-tuning and data augmentation have been proposed in [104], [106], [107]. In [104], Veldanada *et al.* propose to apply data augmentation during fine tuning by adding to each benign image in $\mathcal{D}_{be}$ a Gaussian random noise (the intuition behind this method is that data augmentation should induce the network to perturb to a larger extent the weights, thus facilitating backdoor removal). A similar approach is proposed in [106] where the authors augment the data in $\mathcal{D}_{be}$ by applying image style transfer [108], based on the intuition that the style-transferred images should help the model to forget trigger-related features. In [107], Qiu *et al.* consider 71 data augmentation strategies, and determine the *top*-6 methods, which can efficiently aid the removal of the backdoor by means of fine-tuning. Then, the authors augment the data in $\mathcal{D}_{be}$ with all the six methods, and fine-tune the backdoored model $\mathcal{F}_\theta^\alpha$.

The effectiveness of fine-tuning for backdoor removal has also been discussed in [109], where the impact of several factors on the success of the backdoor attacks, including the type of triggering pattern used by the attacker and the adoption of regularization techniques by the defender, is investigated.

Even if fine-tuning on a benign dataset can reduce the *ASR* in some cases, in general, when used in isolation, its effectiveness is not satisfactory. In [15], a more powerful defence is proposed by combining pruning and fine-tuning. The method is referred to as *fine-pruning*. The pruning defence cuts off part of the neurons in order to damage the backdoor behavior. More specifically, the size of the backdoored network is reduced by eliminating those neurons that are 'dormant' on clean inputs, since neurons behaving in this way are typically activated by the presence of the trigger [4]. To identify and remove those neurons, the images of a benign dataset $\mathcal{D}_{be}$

[9] With a slight abuse of notation, we generically indicate the possibly backdoored tested model as $\mathcal{F}_\theta^\alpha$, even if, in principle, the notation $\mathcal{F}_\theta^\alpha$ should be reserved only for backdoored models.

TABLE 3. Summary of Defence Methods Working At Model Level

Reference	Poisoned model	Working assumptions	Model access	Benign data $\mathcal{D}_{be}$	Datasets	Detection perf. (TPR, TNR)	Removal perf. $(ASR, \mathcal{A})$
Liu et al. [14]	Custom model*	Large $\mathcal{D}_{be}$	White-box	Yes	MNIST	N/A, N/A	5.9%, $[95, 98]\%$
Liu et al. [15]	DeepID, AlexNet, F-RCNN	Large $\mathcal{D}_{be}$	White-box	Yes	YTF, SRD, UTSD	N/A, N/A	$[0, 28.8]\%$, $[87.3, 98.8]\%$
Wang et al. [16]	DeepID, VGG	Small local trigger, shortcuts to target class	White-box	Yes	NIST, GTSRB, YTF	N/A, N/A	$[0.57, 5.7]\%$, $[92, 97]\%$
Liu et al. [70]	VGG, ResNet	Presence of compromised neurons	White-box	Yes	ImageNet, VGGFace	$\approx 90\%$, $\approx 90\%$	N/A, N/A
Veldanda et al. [104]	DeepID, NiN	Visible trigger signal	White-box	Yes	YTF, GTSRB, CIFAR10	N/A, N/A	$[0, 20]\%$, 90%
Chen et al. [22]	ResNet	Shortcuts to target class	Black-box	Yes	MNIST, GTSRB	N/A, N/A	$[7.4, 8.8]\%$, 98%
Xu et al. [105]	ResNet, DenseNet, MobileNet	Fixed dimension of model output	Black-box	No	MNIST, CIFAR10, SC, RTMR	$\approx 90\%$, $\approx 90\%$	N/A, N/A
Kolouri et al. [71]	VGG, ResNet	Fixed dimension of model output	Black-box	No	MNIST, CIFAR10, GTSRB, TinyImageNet	$\approx 100\%$, $\approx 100\%$	N/A, N/A

*The custom model is a 2-layer model that consists of 784 input neurons, 300 hidden neurons, and 10 output neurons.

are tested via the model $\mathcal{F}_\theta^\alpha$. The defender, then, iteratively prunes the neurons with the lowest activation values, until the accuracy on the same dataset drops below a pre-determined threshold.

The difficulty of removing a backdoor by relying only on fine-tuning is shown also in [110]. For this reason, [110] suggests using attention distillation to guide the fine-tuning process. Specifically, *Bob* first fine-tunes the backdoored model on a benign dataset $\mathcal{D}_{be}$, then he applies attention distillation by setting the backdoored model as the *student* and the fine-tuned model as the *teacher*. The empirical results shown in [110] prove that in this way the fine-tuned model is insensitive to the presence of the triggering pattern in the input samples, without causing obvious performance degradation on benign samples.

Recently, Zhao *et al.* [73] have proposed a more efficient defence relying on model connectivity [111]. In particular, [73] shows that two independently trained networks with the same architecture and loss function can be connected in the coefficient-loss landscape, by a simple parametric curve (e.g. Polygonal chain [112] or Bezier curve [113]). The curve or, namely, the path connecting the two models (the endpoints of the curve), can be learned with a limited amount of benign data, i.e., a small $\mathcal{D}_{be}$, with all the models in the path having a similar loss value (performance). The authors showed that when two backdoored models are considered as endpoints, the models in the path can attain similar performance on clean data while drastically reducing the success rate of the backdoor attack. The same behavior can be obtained in the case of only one backdoored model, where the set $\mathcal{D}_{be}$ is used to fine tune the model, and the two models, namely the original backdoored and the fine tuned one, are connected.

Model level defences do not introduce a significant computational overhead, given that they operate before the network is actually deployed in operative conditions. As a drawback, to implement these methods, *Bob* needs a white-box access to the model, and the availability of a large benign dataset $\mathcal{D}_{be}$ for fine-tuning.

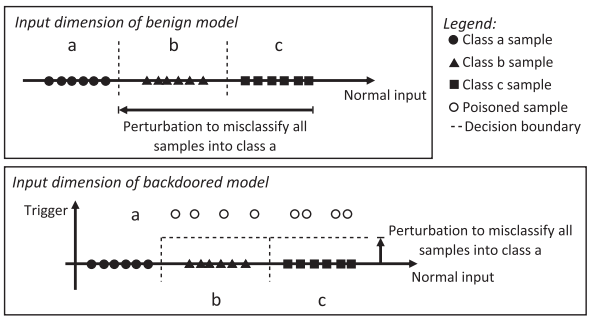


FIGURE 15. Simplified representation of the input space of a clean model (top) and a source-agnostic backdoored model (bottom). A smaller modification is needed to move samples of class 'b' and 'c' across the decision boundary of class 'a' in the bottom case.

B. TRIGGER RECONSTRUCTION

The methods belonging to this category specifically assume that the trigger is source-agnostic, i.e., an input from *any* source class plus the triggering pattern υ can activate the backdoor and induce a misclassification in favour of the target class. The defender tries to reverse-engineer υ either by accessing the internal details of the model $\mathcal{F}_\theta^\alpha$ (white-box setting) or by querying it (black-box setting). For all these methods, once the trigger has been reconstructed, the model is retrained in such a way to *unlearn* the backdoor.

The first trigger-reconstruction method, named Neural Cleanse, was proposed by Wang *et al.* [16] in 2019, and is based on the following intuition: a source-agnostic backdoor creates a *shortcut* to the target class by exploiting the sparsity of the input space. Fig. 15 exemplifies the situation for the case of a 2-dimensional input space. The top figure illustrates a clean model, where a large perturbation is needed to move

any sample of 'b' and 'c' classes into class 'a'. In contrast, the bottom part of the figure shows that for the backdoored model a *shortcut* to the target class 'a' exists, since, due to the presence of the backdoor, the region assigned to class 'a' is expanded along a new direction, thus getting closer to the regions assigned to 'b' and 'c'. The presence of this backdoor-induced region reduces the strength of the perturbation needed to misclassify samples belonging to the classes 'b' and 'c' into 'a'. Based on this observation, for each class k ($k = 1, \ldots, C$), *Bob* calculates the perturbation υ_k necessary to misclassify the other samples into class k. Given the perturbations υ_k, a detection algorithm is run to detect if a class k^* exists for which such perturbation is significantly smaller (in L_1 norm) than for the other classes. More specifically, given a clean validation dataset $\mathcal{D}_{be}$ and a suspect model $\mathcal{F}_\theta$, the defender reverse-engineers the perturbation υ_k for each class k by optimizing the following multi-objective function:

$$\upsilon_k = \min_{\upsilon} \sum_{i=1}^{|\mathcal{D}_{be/k}|} L\left(f_\theta\left(\mathcal{P}\left(x_i, \upsilon\right)\right), k\right) + \lambda ||\upsilon||_\infty, \qquad (14)$$

where $\mathcal{D}_{be/k}$ is the dataset $\mathcal{D}_{be}$ without the samples belonging to class k.

To eventually determine whether the model $\mathcal{F}_\theta$ is backdoored or not, the defender exploits the median absolute deviation outlier detection algorithm [114], analyzing the L_1 norm of all perturbations υ_k ($k = 1, \ldots, C$). If there exists a $\upsilon_{k'}$, for some k', whose L_1 norm is significantly smaller than the others, $\mathcal{F}_\theta$ is judged to be backdoored and $\upsilon_{k'}$ is the reverse engineered trigger. At this point, the reverse-engineered trigger $\upsilon_{k'}$ is used to remove the backdoor from the model. Removal is performed by fine-tuning the model on the benign dataset $\mathcal{D}_{be}$ by adding $\upsilon_{k'}$ to 20% of the samples and by correctly labelling them. Regarding computational complexity, backdoor detection and reverse engineering is the most time-consuming part of the process, with a cost that is proportional to the number of classes. For a model trained on YTF dataset with 1286 classes, detection takes on average 14.6 seconds for each class, for a total of 5.2 hours. In contrast, the computation complexity of the removal part is negligible.

NeuralCleanse assumes that the trigger overwrites a small (local) area of the image, like a square pattern or a sticker. In [17], Guo *et al.* show that NeuralCleanse fails to detect the backdoor for some kinds of local triggers. The failure is due to the poor fidelity of the reconstructed triggers, that, compared with the true trigger, are scattered and overly large. To solve this problem, Guo *et al.* introduce a regularization term controlling the size and smoothness of the reconstructed trigger, that can effectively improve the performance of the defence.

Three additional approaches based on the *shortcut* assumption have been proposed in [115]–[117]. In [115] and [116], backdoor detection is cast into a hypothesis testing framework approach on maximum achievable misclassification fraction statistic. In [117], given a small set of benign data $\mathcal{D}_{be}$, the detector determines the presence of a backdoor in a model by observing the similarity between the per-image adversarial perturbations in $\mathcal{D}_{be}$ and a universal perturbation computed on all the samples of $\mathcal{D}_{be}$. If they are close or similar, the model is considered to be backdoored. Moreover, [117] also achieves data-free detection by substituting $\mathcal{D}_{be}$ with a set of randomly generated (noise) images.

Liu *et al.* [70] have proposed a technique, called Artificial Brain Stimulation (ABS), that analyzes the behavior of the inner neurons of the network, to determine how the output activations change when different levels of stimulation of the neurons are introduced. The method relies on the assumption that backdoor attacks compromise the hidden neurons to inject the hidden behavior. Specifically, the neurons that raise the activation of a particular output label (targeted misclassification) regardless of the input are considered to be potentially compromised. The trigger is then reverse-engineered through an optimization procedure using the stimulation analysis results.The recovered trigger is further utilized to double-check if a neuron is indeed compromised or not, in order to avoid that *clean* labels are judged to be compromised. The optimization aims at achieving multiple goals: i) maximize the activation of the candidate neurons, ii) minimize the activation changes of other neurons in the same layer, and iii) minimize the size of the estimated trigger. The complexity of the neural stimulation analysis is proportional to the total number of neurons.

Yet another way to reconstruct the trigger has been proposed in [104]. The suspect model $\mathcal{F}_\theta$ is first fine-tuned on an augmented set of benign images obtained by noise addition to the images in $\mathcal{D}_{be}$. In this way, a clean model $\mathcal{F}_{\theta_c}$ is obtained. Then, the images which cause a prediction disagreement between $\mathcal{F}_\theta$ and $\mathcal{F}_{\theta_c}$ are identified as potentially poisoned images. Eventually, by training on both $\mathcal{D}_{be}$ and the poisoned images, a CycleGAN learns to poison clean images by adding to them the triggering pattern. The generated backdoored images and their corresponding clean labels are used for the second retraining round of $\mathcal{F}_{\theta_c}$. The effectiveness of the method has been proven in [104] for the case of visible triggers. This method, called NNoculation, outperforms both NeuralCleanse and ABS under more challenging poisoning scenarios, where no constraint is imposed on the size and location of the triggering pattern.

A limitation with the methods in [16], [17], [70], [104] is that they require that the defender has a white-box access to the inspected model. To overcome this limitation, Chen *et al.* [22] have proposed a defence based on the same idea of the *shortcuts* exploited by NeuralCleans, but that requires only a black-box access to the model $\mathcal{F}_\theta$ (it is assumed that the model can be queried an unlimited number of times). To recover the distribution of the triggering pattern υ, the defender employs a conditional GAN (cGAN), that consists of two components: the generator $\mathcal{G}(z, k) = \upsilon_k$, outputting the potential trigger for class k, sampled from the trigger distribution, where z is a random noise, and a fixed, non-trainable, discriminator, corresponding to $\mathcal{F}_\theta$. For each class k, the generator

$\mathcal{G}$ is trained by minimizing a loss function defined as:

$$L(x,k) = L_D\left(x + \mathcal{G}(z,k), k\right) + \lambda L_G(z,k), \qquad (15)$$

where $L_D(x,k) = -\log\left(\left[f_{\theta_\alpha}(x)\right]_k\right)$[10] and $L_G(x,k)$ is a regularization term that ensures that the estimated poisoned image $\hat{\hat{x}} = x + \mathcal{G}_\omega(z,k)$ can not be distinguished from the original one, and that the magnitude of $\mathcal{G}(z,k)$ is limited (to stabilize training). Once the potential triggers $\mathcal{G}(z,k)\,(k = 1 \ldots C)$ have been determined, the defender proceeds as in [16] to perform outlier detection determining the trigger υ, and then remove the backdoor via fine-tuning. With regard to the time complexity, the method is 9.7 times faster than NeuralCleanse, when the model is trained for a 2622-classification task on the VGGface dataset.

Another black-box defence based on trigger reconstruction and outlier detection, that also resorts to a GAN to reconstruct the trigger, has been proposed by Zhu *et al.* [118]. Notably, the methods in [22], [104] and [118] have been shown to work with various patterns and sizes of the trigger, and are also capable to reconstruct multiple triggers, whereas NeuralCleanse [16] can detect only a single, small-size, and invariant trigger. Another method based on trigger reconstruction that can effectively work with multiple triggers has been proposed by Qiao *et al.* [119], under the strong assumption that the trigger size is known to the defender.

All the methods based on trigger reconstruction have a complexity which is proportional to the number of classes. Therefore, when the classification task has a large number of classes (like in many face recognition applications, for instance), those methods are very time consuming.

C. META-CLASSIFICATION

The approaches resorting to meta-classification aim at training a neural network to judge whether a model is backdoored or not. Given a set of N trained models, half backdoored ($\mathcal{F}^{\alpha}_{\theta_i}$) and half benign ($\mathcal{F}_{\theta_i}$), $i = 1, .., N$, the goal is to learn a classifier $\mathcal{F}^{meta}_{\theta} : \mathcal{F} \rightarrow \{0, 1\}$ to discriminate them. Methods that resort to meta-classification are provided in [105] and [71]. In [105], given the dataset of models, the features to be used for the classification are extracted by querying each model $\mathcal{F}_{\theta_i}$ (or $\mathcal{F}^{\alpha}_{\theta_i}$) with several inputs and concatenating the extracted features, i.e., the vectors $f^{-1}_{\theta_i}$ (or $f^{-1}_{\theta_{i,\alpha}}$). Eventually, the meta-classifier $\mathcal{F}^{meta}_{\theta}$ is trained on these feature vectors. To improve the performance of meta-classification, the meta-classifier and the query set are jointly optimized. A different approach is adopted in [71], where a functional is optimized in order to get universal patterns z_m, $m = 1, .., M$, such that looking at the output of the networks in correspondence to such z_m's, that is, $\{f(z_m)\}_{m=1}^{M}$, allows to reveal the presence of the backdoor. Another difference between [105] and [71] is in the way the dataset of the backdoored models $\mathcal{F}^{\alpha}_{\theta_i}$ is generated, that is, in the distribution of the triggering patterns. In [105], the poisoned models considered in the training set are obtained by training them on a poisoned set of images where the triggering patterns follow a so-called jumbo distribution, and consist in continuous compact patterns, with random shape, size, and transparency. In [71] instead, the triggering patterns used to build the poisoned samples used to train the various models are square shaped fixed geometrical patterns. In both cases, the patterns have random location.

Interestingly, both methods generalize well to a variety of triggering patterns that were not considered in the training process. Moreover, while the method in [105] lacks flexibility, as $\mathcal{F}^{meta}_{\theta}$ works for a fixed dimension of the feature space of the to-be-tested model, the method in [71] generalizes also to different architectures, with a different number of neurons, different depths and activation functions, with respect to those considered during training. Computational complexity is high for off-line training, however, the meta-classification is very fast.

VI. TRAINING DATASET LEVEL DEFENCES

With defences operating at the training dataset level, the defender (who now corresponds to *Alice*) is assumed to control the training process, so she can directly inspect the poisoned training dataset $\mathcal{D}^{\alpha}_{tr}$ and access the possibly backdoored model $\mathcal{F}^{\alpha}_{\theta}$ while it is being trained. The dataset $\mathcal{D}^{\alpha}_{tr}$ consists of C subsets $\mathcal{D}_{tr,k}$, including the samples of class k ($k = 1, \ldots, C$). The common assumption made by defence methods working at this level is that among the subsets $\mathcal{D}_{tr,k}$ there exists (at least) one subset $\mathcal{D}_{tr,t}$, containing both benign and poisoned data, while the other subsets include only benign data. Then, the detection algorithm $Det(\cdot)$ and the removal algorithm $Rem(\cdot)$ work directly on $\mathcal{D}^{\alpha}_{tr}$. A summary of all relevant works operating at the training dataset level is given in Table 4.

An obvious defence at this level, at least for the corrupted-label scenario, would consist in checking the consistency of the labels and removing the samples with inconsistent labels from $\mathcal{D}^{\alpha}_{tr}$. Despite its conceptual simplicity, this process requires either a manual investigation or the availability of efficient labelling tools, which may not be easy to build. More general and sophisticated approaches, which are not limited to the case of corrupted-label settings, are described in the following.

In 2018, Tran *et al.* [18] have proposed to use an anomaly detector to reveal anomalies inside the training set of one or more classes. They employ singular value decomposition (SVD) to design an outlier detector, which detects outliers among the training samples by analyzing their feature representation, that is, the activations of the last hidden layer $f^{-1}_{\theta_\alpha}$ of $\mathcal{F}^{\alpha}_{\theta}$. Specifically, the defender splits $\mathcal{D}^{\alpha}_{tr}$ into C subsets $\mathcal{D}_{tr,k}$, each with the samples of class k. Then, for every k, SVD is applied to the covariance matrix of the feature vectors of the images in $\mathcal{D}_{tr,k}$, to get the principal directions. Given the first principal direction d_1, the outlier score for each image x_i is calculated as $(x_i \cdot d_1)^2$. Such a score is then used to measure the deviation of each image from the centroid of the distribution. The images are ranked based on the outlier score and the top ranked $1.5p|\mathcal{D}_{tr,k}|$

[10] We remind that $[f_{\theta_\alpha}(x)]_k$ is the predicted probability for class k.

TABLE 4. Summary of Defence Methods Working At the Training Dataset Level

Reference	Poisoning ratio α, and poisoning ratio over a class β_k	Working assumptions	Model access	Benign data $\mathcal{D}_{be}$	Datasets	Detection performance (TPR,TNR)	Removal performance $(ASR,\ \mathcal{A})$
Tran et al. [18]	$\alpha \leq 0.01,\ \beta_k \leq 0.1$	N/A	White-box	No	CIFAR10	N/A, N/A	$[0, 8.3]\%$, $[92.24, 93.01]\%$
Chen et al. [19]	$\alpha \leq 0.03,\ \beta_k \leq 0.3$	N/A	White-box	No	MNIST	N/A, N/A	$[0, 1.6]\%$, $\approx 100\%$
Xiang et al. [120]	N/A	One-pixel trigger	White-box	No	CIFAR10	$[96.2, 98.9]\%$, $[99.6, 99.8]\%$	$\approx 0\%$, 91.18%
Peri et al. [121]	N/A	Clean-label attacks	White-box	No	CIFAR10	100%, $> 95\%$	N/A, N/A

images are removed for each class, where $p \in [0, 0.5]$. Finally, *Alice* retrains a cleaned model $\mathcal{F}_{\theta_c}$ from scratch on the cleaned dataset. No detection function, establishing if the training set is poisoned or not, is actually provided by this method (which aims only at cleaning the possibly poisoned dataset).

In [19], Chen *et al.* describe a so-called Activation Clustering (AC) method, that analyzes the neural network activations of the last hidden layer (the representation layer), to determine if the training data has been poisoned or not. The intuition behind this method is that a backdoored model assigns poisoned and benign data to the target class based on different features, that is, by relying on the triggering pattern for the poisoned samples, and the ground-truth features for the benign ones. This difference is reflected in the representation layer. Therefore, for the target class of the attack, the feature representations of the samples will tend to cluster into two groups, while the representations for the other classes will cluster in one group only. Based on this intuition, for each subset $\mathcal{D}_{tr,k}$ of $\mathcal{D}_{tr}^{\alpha}$, the defender feeds the images x_i to the model $\mathcal{F}_{\theta}^{\alpha}$ obtaining the corresponding subset of feature representation vectors or activations $f_{\theta_\alpha}^{-1}(x_i)$. Once the activations have been obtained for each training sample, the subsets are clustered separately for each label. To cluster the activations, the k-means algorithm is applied with $k = 2$ (after dimensionality reduction). k-means clustering separates the activations into two clusters, regardless of whether the dataset is poisoned or not. Then, in order to determine which, if any, of the clusters corresponds to a poisoned subset, one possible approach is to analyze the relative size of the two clusters. A cluster is considered to be poisoned if it contains less than p of data for the k class, that is, $p|\mathcal{D}_{tr,k}|$ samples, where $p \in [0, 0.3]$ (the expectation being that poisoned clusters contain no more than a small fraction of class samples, that is $\beta_k \leq p$). The corresponding class is detected as the target class. As a last step, the defender cleans the training dataset, by removing the smallest cluster in the target class, and retraining a new model $\mathcal{F}_{\theta_c}$ from scratch on the cleaned dataset. As we said, AC can be applied only when the class poisoning ratio β_k is lower than p, ensuring that the poisoned data represents a minority subset in the target class. Another method resorting to feature clustering to detect a backdoor attack has been proposed in [122].

Even if k-means clustering with $k = 2$ can perfectly separate the poisoned data on MNIST and CIFAR-10 when a perceptible triggering pattern is used, Xiang *et al.* [120] have shown that in many cases, e.g. when the backdoor pattern is more subtle, the representation vectors of poisoned and benign data can not be separated well in the feature space. This is the case, for instance, when CIFAR10 is attacked with the single pixel backdoor attack. To improve the results in this case, the authors replace k-means clustering with a method based on a Gaussian Mixture Model (GMM), which can also automatically determine the number of clusters. Under the assumption of subtle (one-pixel) trigger, the authors apply blurring filtering to determine whether a cluster is poisoned or not. After blurring, the samples from the poisoned cluster are assigned to the true class with high probability.

A defence working at the training dataset level designed to cope with clean-label backdoor attacks has been proposed in [121]. The defence relies on a so-called deep k-Nearest Neighbors (k-NN) defence against feature-collision [40] and the convex polytope [90] attacks mentioned in Section III-B. The defence relies on the observation that, in the representation space, the poisoned samples of a feature collision attack are surrounded by samples having a different label (the target label) (see Fig. 12). Then, the authors compare the label of each point x_i^{tr} of the training set, with its k-nearest neighbors (determined based on the Euclidean distance) in the representation space. If the label of x^{tr} does not correspond to the label of the majority of the k neighbors, x^{tr} is classified as a poisoned sample and removed from the training dataset. Eventually, the network is retrained on the cleaned training dataset to obtain a clean model $\mathcal{F}_{\theta_c}$.

As the last example of this class of defences, we mention the work proposed in [123]. The defence proposed therein works against source-specific backdoor attacks, that is, attacks for which the triggering pattern causes a misclassification only when it is added to the images of a specific class (also called targeted contamination attacks). The authors show that this kind of backdoor is more stealthy than source-agnostic backdoors. In this case, in fact, poisoned and benign data can not be easily distinguished by looking at the representation level. The approach proposed in [123] is built upon the *universal variation* assumption, according to which the natural variation of the samples of any uninfected class follows the same distribution of the benign images in the attacked class. For example, in image classification tasks, the natural intra-class variation of each object (e.g., lighting, poses, expressions, etc.) has the same distribution across all labels (this is, for instance, the case of image classification, traffic sign and face recognition tasks). For such tasks, a DNN model tends

to generate a feature representation that can be decomposed into two parts, one related to the object's identity (e.g. a given individual) and the other depending on the intra-class variations, randomly drawn from a distribution. The method described in [123] proposes to separate the identity-related features from those associated to the intra-class variations by running an Expectation-Maximization (EM) algorithm [124] across all the representations of the training samples. Then, if the data distribution of one class is scattered, that class will be likely split into two groups (each group sharing a different identity). If the data distribution is concentrated, the class will be considered as single cluster sharing the same identity. Finally, the defender will judge the class with two groups as an attacked class.

Other works working at the training dataset level are described below.

Du *et al.* [125] have theoretically and empirically proved that applying differential privacy during the training process can efficiently prevent the model from overfitting to the atypical samples. Inspired by this, the authors first add Gaussian noise to the poisoned training dataset, and then utilize it to train an auto-encoder outlier detector. Since poisoned samples are atypical ones, the detector judges one sample to be poisoned if the classification is achieved with less confidence. Finally, Yoshida *et al.* [126] and Chen *et al.* [127] share a similar idea for cleaning poisoned data, that is, distilling the clean knowledge from the backdoored model, and further removing poisoned data from the poisoned training dataset by comparing the predictions of the backdoored and distillation models.

VII. FINAL REMARKS AND RESEARCH ROADMAP

In this work, we have given an overview of backdoor attacks against deep neural networks and possible defences. We started the overview by presenting a unifying framework to cast backdoor attacks in. In doing so, we paid particular attention to define the threat models and the requirements that the attackers and defenders must satisfy under various settings. Then, we reviewed the main attacks and defences proposed so far, casting them in the general framework outlined previously. This allowed us to critically review the strengths and drawbacks of the various approaches with reference to the application scenarios wherein they are operating. At the same time, our analysis helps to identify the main open issues still waiting for a solution, thus contributing to outlining a roadmap for future research, as described in the rest of this section.

A. OPEN ISSUES

Notwithstanding the amount of works published so far, there are several open issues that still remain to be addressed, the most relevant of which are detailed in the following.

- *More general defences:* Existing defences are often tailored solutions that work well only under very specific assumptions about the behavior of the adversary, e.g. on the triggering pattern and its size. In real life applications, however, these assumptions do not necessarily hold. Future research should, then, focus on the development of more general defences, with minimal working assumptions on the attacker's behaviour.
- *Improving the robustness of backdoors:* The development of strategies to improve backdoor robustness is another important research line that should occupy the agenda of researchers. Current approaches can resist, up to some extent, parameter pruning and fine-tuning of final layers, while robustness against retraining of all layers and, more in general, transfer learning, is not at reach of current techniques. Achieving such robustness is particularly relevant when backdoors are used for benign purposes (see VII-C). The study of backdoor attacks in the physical domain is another interesting, yet rather unexplored, research direction, (see [128] for a preliminary work in this sense), calling for the development of backdoor attacks that can survive the analog to digital conversion involved by physical domain applications.
- *Development of an underlying theory:* We ambitiously advocate the need of an underlying theory that can help to solve some of the fundamental problems behind the development of backdoor attacks, like, for instance, the definition of the *optimal* triggering pattern (in most of the backdoor attacks proposed so far, the triggering pattern is a prescribed signal, arbitrary defined). From the defender's side, a theoretical framework can help the development of more general defences that are effective under a given threat model.
- *Video backdoor attacks (and defences):* Backdoor attacks against video processing networks have attracted significant less interest than attacks working on still images, yet there would be plenty of applications wherein such attacks would be even more relevant than for image-based systems. As a matter of fact, the current literature either focuses on the simple corrupted-label scenario [76], or it merely applies tools developed for images at the video frame level [94]. However, for a proper development of video backdoor attacks (and defences), the temporal dimension has to be taken into account, e.g., by designing a triggering pattern that exploits the time dimension of the problem.

B. EXTENSION TO DOMAINS OTHER THAN COMPUTER VISION

As mentioned in the introduction, although in this survey we focused on image and video classification, backdoor attacks and defences have also been studied in other application domains, e.g., in deep reinforcement learning [129] and natural language processing [28], where, however, the state of the art is less mature.

1) DEEP REINFORCEMENT LEARNING (DRL)

In 2020, Kiourti *et al.* [129] have presented a backdoor attack against a DRL system. In this scenario, the backdoored

network behaves normally on untainted states, but works abnormally in some particular states, i.e., the poisoned states, s_t^*. In the non-targeted attack case, the abnormal behavior consists in the agent taking a random action, while for the targeted attack the action taken in correspondence of a poisoned state is a target action chosen by the attacker. The desired abnormal behavior is obtained by poisoning the rewards, assigning a positive reward when the target action is taken in correspondence of s_t^* in the targeted case, or when every action (but the correct one) is taken in the non-targeted case. According to the result shown in [129], a successful attack is obtained by poisoning a very small percentage of trajectories (states) and rewards.

Some defences to protect a DRL system from backdoor attacks have been also explored in [129]. It turns out that neither spectral signature [130] nor activation clustering [19] can detect the attack because of the small poisoning ratio α. The development of backdoor attacks against DRL system is only at an early stage, and, in particular, the study of effective backdoor defences is still an open problem.

2) NATURAL LANGUAGE PROCESSING (NLP)

In the NLP domain backdoor attacks and, in particular, defences, are quite advanced. Starting from [28], several works have shown that NLP tools are vulnerable to backdoor attacks. Most of these works implicitly assume that the attack is carried out in a *full control* scenario, where Eve poisons the training dataset in a corrupted-label modality, adding a triggering pattern υ, namely, a specific word token, within benign text sequences, and setting the corresponding label to the target class t. The backdoored model will behave as expected on normal text sentences, but will always output t if υ is present in the text string. The first approaches proposed by Kurita *et al.* [131] and Wallace *et al.* [132] used noticeable or misspelled words as trigger υ, e.g. 'mm,' 'bb' and 'James Bond,' that can then be easily detected at test time. In [133] and [134], a less detectable trigger is used by relying on a proper combination of synonyms and syntaxes.

Two defences [132], [135] have also been proposed to detect or remove the backdoor from NLP models. Both these methods have serious drawbacks. In [132], the removal of the backdoor reduces the accuracy of the model on benign text, thus not satisfying the *harmless removal* requirement. The method proposed in [135], based on the shortcut assumption described in [16], instead, is very time consuming, requiring the computation of a universal perturbation for all possible target classes, which, in NLP applications, can be many. Future work in this area should address the development of clean-labels attacks, and work on more efficient detection and removal methods.

C. BENIGN USES OF BACKDOORS

Before concluding the paper, we pause to mention two possible benign uses of backdoors.

1) DNN WATERMARKING

Training a DNN model is a noticeable piece of work that requires significant computational resources (the training process may go on for weeks, even on powerful machines equipped with several GPUs) and the availability of massive amounts of training data. For this reason, the demand for methods to protect the Intellectually Property Rights (IPR) associated to DNNs is rising. As it happened for media protection [136], watermarking has recently been proposed as a way to protect the IPRs of DNNs and identify illegitimate usage of DNN models [137]. In general, the watermark can either be embedded directly into the weights by modifying the parameters of one or more layers (static watermarking), or be associated to the behavior of the network in correspondence to some specific inputs (dynamic watermarking) [138].

The latter approach has immediate connections with DNN backdooring. In 2018, Adi *et al.* [139] were the first to propose to black-box watermark a DNN through backdooring. According to [139], the watermark is injected into the DNN during training, by adding a poisoning dataset ($\mathcal{D}_{tr}^p$) to the benign training data ($\mathcal{D}_{tr}^b$). The triggering input images in $\mathcal{D}_{tr}^p$ play the role of the watermark key. To verify the ownership, the verifier computes the *ASR*; if the value is larger than a prescribed threshold the ownership of the DNN is established.

In [139], watermark robustness against fine-tuning and transfer learning was evaluated. The results showed that the watermark can be recovered after fine tuning in some cases, while in other cases the accuracy of watermark detection drops dramatically. Transfer learning corresponds to an even more challenging scenario against which robustness can not be achieved. Noticeably, poor robustness against transfer learning is a common feature of all DNN watermarking methods developed so far. Improving the robustness of DNN watermarking against network re-use is of primary importance in practical IPR protection applications. This is linked to the quest for improving backdoor robustness, already discussed in the previous section. Moreover, the use of backdoors for DNN watermarking must be investigated more carefully in order to understand the capability and the limitations of the backdooring approach in terms of payload (capacity) and security, and how it compares with static watermarking approaches.

2) TRAPDOOR-ENABLED ADVERSARIAL EXAMPLE DETECTION

DNN models are known to be vulnerable to adversarial examples, causing misclassification at testing time [1]. Defence methods developed against adversarial examples work either by designing a system for which adversarial attacks are more difficult to be found (see, for instance, adversarial training [93] and defensive distillation [140]), or by trying to detect the adversarial inputs at testing time (e.g., by feature squeezing, or input pre-processing [141]).

Recently, Shan *et al.* [142] have proposed to exploit backdoor attacks to protect DNN models against adversarial examples, by implementing a so-called trapdoor honeypot. A

trapdoor honeypot is similar to a backdoor in that it causes a misclassification error in the presence of a specific, minimum energy, triggering pattern. When building an adversarial example, the attacker will likely, and inadvertently, exploit the weakness introduced within the DNN by the backdoor and come out with an adversarial perturbation which is very close to the triggering pattern purposely introduced by the defender at training time. In this way, the defender may recognize that an adversarial attack is ongoing and react accordingly.

More specifically, given a to-be-protected class t, the defender trains a backdoored model $\mathcal{F}_{\theta_\alpha^*}$ such that $\mathcal{F}_{\theta_\alpha^*}(x+\upsilon) = t \neq \mathcal{F}_{\theta_\alpha^*}(x)$, where υ is a low-energy triggering pattern, called loss-minimizing trapdoor, designed in such a way to minimize the loss for the target label. The presence of an adversarial input can then be detected by looking for the presence of the pattern υ within the input sample, trusting that the algorithm used to construct the adversarial perturbation will exploit the existence of a low-energy pattern υ capable of inducing a misclassification error in favour of class t. Based on the results shown in [142], the trapdoor-enabled defence achieves high accuracy against many state-of-art targeted adversarial examples attacks.

Such defence works only against targeted attacks, and trapdoor honeypots against non-targeted adversarial example have still to be developed. Moreover, how to extend the idea of trapdoor honeypots to defend against black-box adversarial examples, that do not adopt a low-energy pattern, is an open issue deserving further attention.

REFERENCES

[1] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014, pp. 1–10.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[3] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proc. 29th Int. Conf. Int. Conf. Mach. Learn.*, 2012, pp. 1467–1474.

[4] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019, doi: 10.1109/ACCESS.2019.2909068.

[5] L. Muñoz-González *et al.*, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 27–38.

[6] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*.

[7] L. Muñoz-González, B. Pfitzner, M. Russo, J. Carnerero-Cano, and E. C. Lupu, "Poisoning attacks with generative adversarial Nets," 2019, *arXiv:1906.07773*.

[8] P. W. Koh, J. Steinhardt, and P. Liang, "Stronger data poisoning attacks break data sanitization defenses," *Mach. Learn.*, vol. 111, no. 1, pp. 1–47, 2022.

[9] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 3520–3532, 2017.

[10] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart, "Sever: A robust meta-algorithm for stochastic optimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1596–1606.

[11] J. Carnerero-Cano, L. Muñoz-González, P. Spencer, and E. C. Lupu, "Regularisation can mitigate poisoning attacks: A novel analysis based on multiobjective bilevel optimisation," 2020, *arXiv:2003.00040*.

[12] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Secur. Privacy*, 2018, pp. 19–35.

[13] Y. Ji, X. Zhang, and T. Wang, "Backdoor attacks against learning systems," in *Proc. IEEE Conf. Netw. Secur.*, 2017, pp. 1–9.

[14] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *Proc. IEEE Int. Conf. Comput. Des.*, 2017, pp. 45–48.

[15] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*, 2018, pp. 273–294.

[16] B. Wang *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 707–723.

[17] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems," 2019, *arXiv:1908.01763*.

[18] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8000–8010.

[19] B. Chen *et al.*, "Detecting backdoor attacks on deep neural networks by activation clustering," in *Proc. Workshop Artif. Intell. Saf. Co-Located 33rd AAAI Conf. Artif. Intell.*, 2019, vol. 2301, 2019.

[20] E. Chou, F. Tramèr, and G. Pellegrino, "SentiNet: Detecting localized universal attacks against deep learning systems," in *Proc. IEEE Secur. Privacy Workshops*, 2020, pp. 48–54.

[21] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, 2019, pp. 113–125.

[22] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4658–4664.

[23] Y. Liu *et al.*, "A survey on neural trojans," in *Proc. IEEE 21st Int. Symp. Qual. Electron. Des.*, 2020, pp. 33–39, doi: 10.1109/ISQED48828.2020.9137011.

[24] Y. Chen, X. Gong, Q. Wang, X. Di, and H. Huang, "Backdoor attacks and defenses for deep neural networks in outsourced cloud environments," *IEEE Netw.*, vol. 34, no. 5, pp. 141–147, Sep./Oct. 2020.

[25] Y. Li, B. Wu, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," 2020, *arXiv:2007.08745*.

[26] M. Goldblum *et al.*, "Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Mar. 25, 2022, pp. 1–1, 2022, doi: 10.1109/TPAMI.2022.3162397.

[27] A. Schwarzschild, M. Goldblum, A. Gupta, J. P. Dickerson, and T. Goldstein, "Just how toxic is data poisoning? A unified benchmark for backdoor and data poisoning attacks," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, vol. 139, pp. 9389–9398.

[28] J. Dai, C. Chen, and Y. Li, "A backdoor attack against LSTM-based text classification systems," *IEEE Access*, vol. 7, pp. 138872–138878, 2019.

[29] H. Kwon and S. Lee, "Textual backdoor attack for the text classification system," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, 2021.

[30] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[31] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.

[32] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–15.

[33] C.-L. Chen, L. Golubchik, and M. Paolieri, "Backdoor attacks on federated meta-learning," 2020, *arXiv:2006.07026*.

[34] C. Xie, M. Chen, P. Chen, and B. Li, "CRFL: Certifiably robust federated learning against backdoor attacks," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 11372–11382. [Online]. Available: http://proceedings.mlr.press/v139/xie21a.html

[35] Y. Li, Y. Li, Y. Lv, Y. Jiang, and S.-T. Xia, "Hidden backdoor attack against semantic segmentation models," 2021, *arXiv:2103.04038*.

[36] N. Carlini and A. Terzis, "Poisoning and backdooring contrastive learning," 2021, *arXiv:2106.09667*.

[37] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," 2017, *arXiv:1612.00796*.

[38] J. Dumford and W. J. Scheirer, "Backdooring convolutional neural networks via targeted weight perturbations," in *Proc. IEEE Int. Joint Conf. Biometrics*, 2020, pp. 1–9, doi: 10.1109/IJCB48548.2020.9304875.

[39] R. Costales, C. Mao, R. Norwitz, B. Kim, and J. Yang, "Live trojan attacks on deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 796–797.

[40] A. Shafahi *et al.*, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6106–6116.

[41] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1211–1220.

[42] J. Bai, B. Wu, Y. Zhang, Y. Li, Z. Li, and S.-T. Xia, "Targeted attack against deep neural networks via flipping limited weight bits," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=iKQAk8a2kM0

[43] S. Hong, N. Carlini, and A. Kurakin, "Handcrafted backdoors in deep neural networks," 2021, *arXiv:2106.04690*.

[44] Y. Li, J. Hua, H. Wang, C. Chen, and Y. Liu, "Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection," in *Proc. 43rd IEEE/ACM Int. Conf. Softw. Eng.*, 2021, pp. 263–274, doi: 10.1109/ICSE43902.2021.00035.

[45] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[46] H. KWON, "Multi-model selective backdoor attack with different trigger positions," *IEICE Trans. Inf. Syst.*, vol. 105, no. 1, pp. 170–174, 2022.

[47] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Proc. 24th IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 529–534.

[48] H. Kwon and Y. Kim, "Blindnet backdoor: Attack on deep neural network using blind watermark," *Multimedia Tools Appl.*, pp. 6217–6234, 2022.

[49] H. Zhong, C. Liao, A. C. Squicciarini, S. Zhu, and D. J. Miller, "Backdoor embedding in convolutional neural network models via invisible perturbation," in *Proc. 10th ACM Conf. Data Appl. Secur. Privacy*, 2020, pp. 97–108, doi: 10.1145/3374664.3375751.

[50] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1765–1773.

[51] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," in *Proc. Int. Joint Conf. Neural Netw.*, 2013, no. 1288, pp. 1–8.

[52] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Tech. Rep., Univ. of Toronto, 2009.

[53] Q. Zhang, Y. Ding, Y. Tian, J. Guo, M. Yuan, and Y. Jiang, "Advdoor: Adversarial backdoor attack of deep learning system," in *Proc. 30th ACM SIGSOFT Int. Symp. Soft. Testing Anal.*, 2021, pp. 127–138, doi: 10.1145/3460319.3464809.

[54] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2088–2105, Sep./Oct. 2021.

[55] T. A. Nguyen and A. T. Tran, "WaNet—imperceptible warping-based backdoor attack," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=eEn8KTtJOx

[56] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, pp. 567–585, Jun. 1989.

[57] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 3730–3738.

[58] E. Quiring and K. Rieck, "Backdooring and poisoning neural networks with image-scaling attacks," in *Proc. IEEE Secur. Privacy Workshops*, 2020, pp. 41–47.

[59] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li, "Seeing is not believing: Camouflage attacks on image scaling algorithms," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 443–460. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/xiao

[60] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2041–2055.

[61] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Brit. Mach. Vis. Conf.*, X. Xie, M. W. Jones, and G. K. L. Tam, Eds., BMVA Press, 2015, pp. 41.1–41.12, doi: 10.5244/C.29.41.

[62] "Casia iris dataset," [Online]. Available: http://biometrics.idealtest.org/

[63] T. J. L. Tan and R. Shokri, "Bypassing backdoor detection algorithms in deep learning," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2020, pp. 175–183, doi: 10.1109/EuroSP48549.2020.00019.

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–14.

[65] Y. Li, T. Zhai, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor attack in the physical world," 2021, *arXiv:2104.02361*.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[67] X. Gong *et al.*, "Defense-resistant backdoor attacks against deep neural networks in outsourced cloud environment," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2617–2631, Aug. 2021, doi: 10.1109/JSAC.2021.3087237.

[68] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in *Proc. 35th AAAI Conf. Artif. Intell., AAAI, 33rd Conf. Innov. Appl. Artif. Intell., IAAI 2021, 11th Symp. Educ. Adv. Artif. Intell., EAAI 2021, Virtual Event*, 2021, pp. 1148–1156. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16201

[69] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2242–2251, doi: 10.1109/ICCV.2017.244.

[70] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1265–1282.

[71] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, "Universal litmus patterns: Revealing backdoor attacks in CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 301–310.

[72] T. A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," in *Adv. Neural Inf. Process. Syst. 33: Annu. Conf. Neural Inf. Process. Syst.*, 2020, pp. 3454–3464. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/234e691320c0ad5b45ee3c96d0d7b8f8-Abstract.html

[73] P. Zhao, P. Chen, P. Das, K. N. Ramamurthy, and X. Lin, "Bridging mode connectivity in loss landscapes and adversarial robustness," in *Proc. 8th Int. Conf. Learn. Representations*, 2020, pp. 1–28. [Online]. Available: https://openreview.net/forum?id=SJgwzCEKwH

[74] Y. Liu *et al.*, "Trojaning attack on neural networks," in *Proc. 25th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15. [Online]. Available: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-5_Liu_paper.pdf

[75] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Proc. Workshop Faces in'Real-Life'Images: Detection, Alignment, Recognit.*, 2008, pp. 1–14.

[76] A. Bhalerao, K. Kallas, B. Tondi, and M. Barni, "Luminance-based video backdoor attack against anti-spoofing rebroadcast detection," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process.*, 2019, pp. 1–6.

[77] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2625–2634.

[78] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in *Proc. BIOSIG- Proc. Int. Conf. Biometrics Special Int. Group*, 2012, pp. 1–7.

[79] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 113–131.

[80] W. Guo, B. Tondi, and M. Barni, "A master key backdoor for universal impersonation attack against DNN-based face verification," *Pattern Recognit. Lett.*, vol. 144, pp. 61–67, 2021.

[81] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFACE2: A dataset for recognising faces across pose and age," in *Proc. 13th IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2018, pp. 67–74.

[82] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," 2019, *arXiv:1912.02771*.

[83] M. Alberti *et al.*, "Are You Tampering with My Data?," in *Proc. Comput. Vis. ECCV Workshops*, 2018, pp. 296–312.

[84] M. Barni, K. Kallas, and B. Tondi, "New backdoor attack in CNNs by training set corruption without label poisoning," in *Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 101–105.

[85] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 182–199.

[86] R. Wan, B. Shi, L.-Y. Duan, A.-H. Tan, and A. C. Kot, "Benchmarking single-image reflection removal algorithms," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3922–3930.

[87] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[88] R. Ning, J. Li, C. Xin, and H. Wu, "Invisible poison: A blackbox clean label backdoor attack to deep neural networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2021, pp. 1–10.

[89] O. Suciu, R. Marginean, Y. Kaya, H. D. III, and T. Dumitras, "When does machine learning fail? generalized transferability for evasion and poisoning attacks," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1299–1316. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/suciu

[90] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 7614–7623. [Online]. Available: http://proceedings.mlr.press/v97/zhu19a.html

[91] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proc. 34th AAAI Conf. Artif. Intell., AAAI, The 32nd Innov. App. Artif. Intell. Conf., The 10th AAAI Symp. Educ. Adv. Artif. Intell.*, 2020, pp. 11957–11965. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/6871

[92] J. Chen, L. Zhang, H. Zheng, X. Wang, and Z. Ming, "Deeppoison: Feature transfer based stealthy poisoning attack for DNNs," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 68, no. 7, pp. 2618–2622, Jul. 2021, doi: 10.1109/TCSII.2021.3060896.

[93] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[94] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang, "Clean-label backdoor attacks on video recognition models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 14 443–14452.

[95] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.

[96] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2020, pp. 897–912.

[97] E. Sarkar, Y. Alkindi, and M. Maniatakos, "Backdoor suppression in neural networks using input fuzzing and majority voting," *IEEE Des. Test*, vol. 37, no. 2, pp. 103–110, Apr. 2020.

[98] H. Kwon, "Detecting backdoor attacks via class difference in deep neural networks," *IEEE Access*, vol. 8, pp. 191049–191056, 2020.

[99] H. Fu, A. K. Veldanda, P. Krishnamurthy, S. Garg, and F. Khorrami, "Detecting backdoors in neural networks using novel feature-based anomaly detection," 2020, *arXiv:2011.02526*.

[100] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam: Generalized gradient-based visual explanations for deep convolutional networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2018, pp. 839–847.

[101] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[102] H. Kwon, "Defending deep neural networks against backdoor attack by using de-trigger autoencoder," *IEEE Access*, early access, Oct. 18, 2021, doi: 10.1109/ACCESS.2021.3086529.

[103] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 93–104.

[104] A. K. Veldanda *et al.*, "NNoculation: Broad spectrum and targeted treatment of backdoored DNNs," 2020, *arXiv:2002.08313*.

[105] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in *Proc. IEEE Symp. Secur. Privacy*, 2021, pp. 103–120.

[106] M. Villarreal-Vasquez and B. Bhargava, "ConFoc: Content-focus protection against trojan attacks on neural networks," 2020, *arXiv:2007.00711*.

[107] H. Qiu, Y. Zeng, S. Guo, T. Zhang, M. Qiu, and B. M. Thuraisingham, "Deepsweep: An evaluation framework for mitigating DNN backdoor attacks using data augmentation," in *Proc. ACM Asia Conf. Comput. Commun. Secur., Virtual Event*, Hong Kong, 2021, pp. 363–377, doi: 10.1145/3433210.3453108.

[108] L. A. Gatys, A. S. Ecker, and M. Bethge, "IMAGE style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2414–2423.

[109] L. Truong *et al.*, "Systematic evaluation of backdoor data poisoning attacks on image classifiers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 788–789.

[110] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: Erasing backdoor triggers from deep neural networks," in *Proc. 9th Int. Conf. Learn. Representations*, 2021, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=9l0K4OM-oXE

[111] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of DNNs," in *Proc. Adv. Neural Inf. Process. Syst. 31: Annu. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8803–8812. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/be3087e74e9100d4bc4c6268cdbe8456-Abstract.html

[112] S. C. Park and H. Shin, "Polygonal chain intersection," *Comput. Graph.*, vol. 26, no. 2, pp. 341–350, 2002.

[113] R. T. Farouki, "The bernstein polynomial basis: A centennial retrospective," *Comput. Aided Geometric Des.*, vol. 29, no. 6, pp. 379–419, 2012.

[114] F. R. Hampel, "The influence curve and its role in robust estimation," *J. Amer. Stat. Assoc.*, vol. 69, no. 346, pp. 383–393, 1974.

[115] Z. Xiang, D. J. Miller, and G. Kesidis, "Detection of backdoors in trained classifiers without access to the training set," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1177–1191, Mar. 2022.

[116] Z. Xiang, D. J. Miller, H. Wang, and G. Kesidis, "Detecting scene-plausible perceptible backdoors in trained DNNs without access to the training set," *Neural Comput.*, vol. 33, no. 5, pp. 1329–1371, 2021, doi: 10.1162/neco_a_01376.

[117] R. Wang, G. Zhang, S. Liu, P. Chen, J. Xiong, and M. Wang, "Practical detection of trojan neural networks: Data-limited and data-free cases," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, vol. 12368, pp. 222–238.

[118] L. Zhu, R. Ning, C. Wang, C. Xin, and H. Wu, "Gangsweep: Sweep out neural backdoors by GAN," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 3173–3181.

[119] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14004–14013.

[120] Z. Xiang, D. J. Miller, and G. Kesidis, "A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process.*, 2019, pp. 1–6.

[121] N. Peri *et al.*, "Deep K-NN defense against clean-label data poisoning attacks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 55–70, doi: 10.1007/978-3-030-66415-2_4.

[122] E. Soremekun, S. Udeshi, S. Chattopadhyay, and A. Zeller, "AEGIS: Exposing backdoors in robust machine learning models," 2020, *arXiv:2003.00865*.

[123] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection," 2019, *arXiv:1908.00686*.

[124] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: A joint formulation," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, vol. 7574, pp. 566–579, doi: 10.1007/978-3-642-33712-3_41.

[125] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," in *Proc. 8th Int. Conf. Learn. Representations*, 2020, pp. 1–11. [Online]. Available: https://openreview.net/forum?id=SJx0q1rtvS

[126] K. Yoshida and T. Fujino, "Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks," in *Proc. 13th ACM Workshop Artif. Intell. Secur.*, 2020, pp. 117–127.
[127] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-pois: An attack-agnostic defense against data poisoning attacks," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3412–3425, May 2021.
[128] M. Xue, C. He, S. Sun, J. Wang, and W. Liu, "Robust backdoor attacks against deep neural networks in real physical world," 2021, *arXiv:2104.07395*.
[129] P. Kiourti, K. Wardega, S. Jha, and W. Li, "Trojdrl: Evaluation of backdoor attacks on deep reinforcement learning," in *Proc. 57th ACM/IEEE Des. Automat. Conf.*, 2020, pp. 1–6.
[130] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4489–4497.
[131] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pretrained models," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 2793–2806, doi: 10.18653/v1/2020.acl-main.249.
[132] E. Wallace, T. Z. Zhao, S. Feng, and S. Singh, "Concealed data poisoning attacks on NLP models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2021, pp. 139–150, doi: 10.18653/v1/2021.naacl-main.13.
[133] F. Qi, Y. Yao, S. Xu, Z. Liu, and M. Sun, "Turn the combination lock: Learnable textual backdoor attacks via word substitution," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics, 11th Int. Joint Conf. Natural Langu. Process.*, 2021, pp. 4873–4883, doi: 10.18653/v1/2021.acl-long.377.
[134] F. Qi *et al.*, "Hidden killer: Invisible textual backdoor attacks with syntactic trigger," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics, 11th Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 443–453, doi: 10.18653/v1/2021.acl-long.37.
[135] A. Azizi *et al.*, "T-MINER: A generative approach to defend against trojan attacks on DNN-based text classification," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2255–2272.
[136] C. I. Podilchuk and E. J. Delp, "Digital watermarking: Algorithms and applications," *IEEE signal Process. Mag.*, vol. 18, no. 4, pp. 33–46, Jul. 2001.
[137] M. Barni, F. Pérez-González, and B. Tondi, "DNN watermarking: Four challenges and a funeral," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, 2021, pp. 189–196.
[138] Y. Li, H. Wang, and M. Barni, "A survey of deep neural network watermarking techniques," *Neurocomputing*, vol. 461, pp. 171–193, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122101095X
[139] Y. Adi, C. Baum, M. Cissé, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1615–1631. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/adi
[140] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 582–597.
[141] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1778–1787.
[142] S. Shan, E. Willson, B. Wang, B. Li, H. Zheng, and B. Y. Zhao, "Gotta catch'Em all: Using concealed trapdoors to detect adversarial attacks on neural networks," 2019, *arXiv:1904.08554*.

WEI GUO received the M.Eng. degree from the Department of Computer and Information Security, Guilin University of Electronic Technology (GUET), Guilin, China, in 2018 with a thesis about "Applied Cryptography in IoT environment" and the B.Sc. degree from the Department of Mathematics and Computational Science, GUET in 2015.

He is currently a Ph.D. candidate from the Department of Information Engineering and Mathematics, University of Siena, Siena, Italy. He is doing research on security concerns in deep neural networks under the supervision of Prof. Mauro Barni. He is a Member of Visual Information Privacy and Protection Group.

BENEDETTA TONDI (Member, IEEE) received the master degree (cum laude) in electronics and communications engineering, and the Ph.D. degree in information engineering and mathematical sciences, both from the University of Siena, Siena, Italy, in 2012 and 2016, respectively, with a thesis on the Theoretical Foundations of Adversarial Detection and Applications to Multimedia Forensics, in the area of Multimedia Security.

She is currently an Assistant Professor with the Department of Information Engineering and Mathematics, University of Siena. She has been Assistant for the course of Information Theory and Coding and Multimedia Security. She is a Member of the Visual Information Processing and Protection Group led by Prof. Mauro Barni. She is part of the IEEE Young Professionals and IEEE Signal Processing Society, and a Member of the National Inter-University Consortium for Telecommunications. From January 2019, she is also a Member of the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society. Recently, she is working on machine learning and deep learning applications for digital forensics and counter-forensics, and on the security of machine learning techniques. From October 2014 to February 2015, she was a Visiting Student with the University of Vigo, Vigo, Spain, Signal Processing in Communications Group, working on the study of techniques to reveal attacks in Watermarking Systems. Her research interests include application of information-theoretic methods and game theory concepts to forensics and counter-forensics analysis and more in general to multimedia security, and on adversarial signal processing. Her stay was funded by a Spanish National Project on Multimedia Security.

MAURO BARNI (Fellow, IEEE) graduated in electronic engineering from the University of Florence, Florence, Italy, in 1991, and received the Ph.D. degree in informatics and telecommunications, in October 1995.

He has carried out his research activity for more than 20 years, first at the Department of Electronics and Telecommunication, University of Florence, then at the Department of Information Engineering and Mathematics, University of Siena, Siena, Italy, where he works as a Full Professor. His activity focuses on digital image processing and information security, with particular reference to the application of image processing techniques to copyright protection (digital watermarking) and authentication of multimedia (multimedia forensics). He has been studying the possibility of processing signals that has been previously encrypted without decrypting them (signal processing in the encrypted domain). Lately, he has been working on theoretical and practical aspects of adversarial signal processing and adversarial machine learning. He has been authored or coauthored more than 350 papers published in international journals and conference proceedings, he holds four patents in the field of digital watermarking, and one patent dealing with anticounterfeiting technology. His papers on digital watermarking have significantly contributed to the development of such a theory in the last decade as it is demonstrated by the large number of citations some of these papers have received. The overall citation record of M. Barni amounts to an h-number of 63 according to Scholar Google search engine. He is coauthor of the book "Watermarking Systems Engineering: Enabling Digital Assets Security and other Applications,"' published by Dekker Inc., in February 2004. He is the Editor of the book "Document and Image Compression" published by CRC Press, in 2006.

Dr. Barni was a recipient of the IEEE Signal Processing Magazine best column award, in 2008. In 2010, he was awarded the IEEE Transactions on Geoscience and Remote Sensing best paper award. He was also the recipient of the Individual Technical Achievement Award of EURASIP EURASIP for 2016. He has been the Chairman of the IEEE Multimedia Signal Processing Workshop held in Siena, in 2004, and the Chairman of the IV edition of the International Workshop on Digital Watermarking. He was the technical program Co-Chair of ICASSP 2014 and the technical program Chairman of the 2005 edition of the Information Hiding Workshop, the VIII edition of the International Workshop on Digital Watermarking and the V edition of the IEEE Workshop on Information Forensics and Security (WIFS 2013).