Backdoor Attacks on Image Classification Models in Deep Neural Networks

Quanxin ZHANG, Wencong MA, Yajie WANG, Yaoyuan ZHANG, Zhiwei SHI, Yuanzhang LI

Citation: Quanxin ZHANG, Wencong MA, Yajie WANG, Yaoyuan ZHANG, Zhiwei SHI, Yuanzhang LI, Backdoor Attacks on Image Classification Models in Deep Neural Networks, Chinese Journal of Electronics, 2022, 31(2), 199–212. doi: 10.1049/cje.2021.00.126.

View online: https://doi.org/10.1049/cje.2021.00.126

Chinese Journal of Electronics
Vol.31, No.2, Mar. 2022

Backdoor Attacks on Image Classification Models in Deep Neural Networks

ZHANG Quanxin[1], MA Wencong[1], WANG Yajie[1], ZHANG Yaoyuan[1], SHI Zhiwei[2], and LI Yuanzhang[1]

(1. *Beijing Institute of Technology, Beijing 100036, China*)

(2. *China Information Technology Security Evaluation Center, Beijing 100085, China*)

Abstract — Deep neural network (DNN) is applied widely in many applications and achieves state-of-the-art performance. However, DNN lacks transparency and interpretability for users in structure. Attackers can use this feature to embed trojan horses in the DNN structure, such as inserting a backdoor into the DNN, so that DNN can learn both the normal main task and additional malicious tasks at the same time. Besides, DNN relies on data set for training. Attackers can tamper with training data to interfere with DNN training process, such as attaching a trigger on input data. Because of defects in DNN structure and data, the backdoor attack can be a serious threat to the security of DNN. The DNN attacked by backdoor performs well on benign inputs while it outputs an attacker-specified label on trigger attached inputs. Backdoor attack can be conducted in almost every stage of the machine learning pipeline. Although there are a few researches in the backdoor attack on image classification, a systematic review is still rare in this field. This paper is a comprehensive review of backdoor attacks. According to whether attackers have access to the training data, we divide various backdoor attacks into two types: poisoning-based attacks and non-poisoning-based attacks. We go through the details of each work in the timeline, discussing its contribution and deficiencies. We propose a detailed mathematical backdoor model to summary all kinds of backdoor attacks. In the end, we provide some insights about future studies.

Key words — Backdoor attack, Poisoning-based attacks, Non-poisoning-based attacks, Security, Review.

I. Introduction

In the past decade, deep neural networks have played an important role in many applications, such as facial recognition systems[1], voice recognition systems[2], and autonomous vehicles[3]. Many edge devices have been equipped with DNN applications, such as mobile phones[4,5], embedded vehicular systems[6], and smart cameras[7]. While people enjoy its convenience, they also bear the security risk that was caused by its uninterpretability[8]. One of the most widely known risks is adversarial examples[9–15], which are conducted at inference time. By adding imperceptible perturbations on the input, the model will predict the labels specified by the attackers.

Another major security threat has aroused widespread attention, which is referred to as backdoor attacks. As illustrated in Fig.1, it aims to perform well on benign samples but outputs attack-specified predictions on samples attached with triggers. Compared to adversarial examples, it can be conducted at almost all stages of the DNN model. The misprediction is only activated by the attacker-specified trigger, which is challenging to defend. According to this, it may cause massive casualties and property losses. As an example of autonomous systems with a backdoor model, the vehicle can correctly recognize the benign "no entry" traffic signal. When there is a trigger on the "no entry" traffic signal, the car will recognize it as a "speed limit 40 km/h" signal. If this trigger is added to the "no entry" signal in front of the cliff, it will cause unimaginable risks.

The backdoor attack is first revealed by Ref.[16]. It is conducted in the training phase. With a poisoned training set combined with clean data and trigger attached data, attackers can insert a backdoor in the model. This work is a typical poisoning-based attack. Following works further improve the stealthiness and efficiency of the backdoor attacks. A milestone is Ref.[17], proposing a clean-label attack, which only embeds the

Manuscript Received Apr. 12, 2021; Accepted Aug. 10, 2021. This work was supported by the National Natural Science Foundation of China (61876019).

 DOI:10.1049/cje.2021.00.126

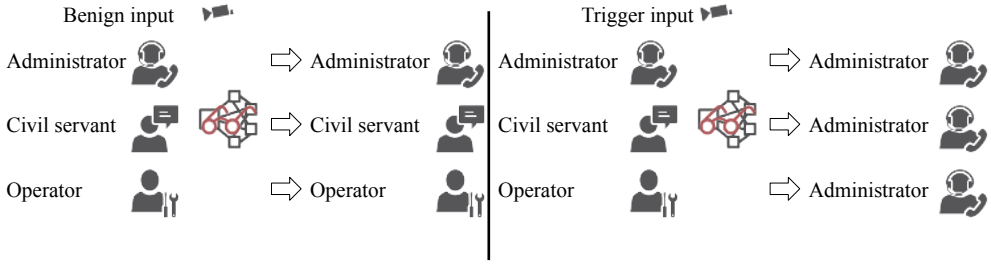

Fig. 1. This is a face recognition system. It can predict the ground-truth label on clean input. When the input is attached with a trigger, e.g. sunglass, the system will output attacker-specified labels

target feature into the original image while maintaining the original image. Another branch is non-poisoning-based attacks. It can be conducted at all pipelines of the DNN model. Instead of manipulating the model weights through poisoning data, it uses diversity methods, such as directly manipulating the model parameters bits via row-hammer[18] or rootkit attacks[19]. There are backdoor attacks against other tasks or paradigms, such as Refs.[20–22] in the area of natural language processing, Refs.[23,24]in reinforcement learning, Refs.[25–27] in collaborative learning. In this paper, we mainly concentrate on backdoor attacks on computer vision.

There are many similarities between data poisoning and poisoning-based backdoor attacks in the training phase[28]. For example, their purposes are both to mislead the model in the inference stage by introducing poisoning samples in the training stage. However, there are also significant differences. First, from the perspective of the attackers' target, data poisoning aims to reduce the performance of benign test samples. This kind of attack leads to low accuracy of the model, which is similar to a denial of service attack. The poisoning-based backdoor attack retains the performance on benign samples, which is similar to the clean model. Second, the conventional poisoning attack is untargeted. It does not care which class's accuracy is misclassified or deteriorated, and the trigger is transparent. In contrast, the backdoor attack is usually performed as a targeted attack—the prediction of the poisoned samples (that is, the benign samples with triggers) is changed to the target label. Third, in terms of concealment, backdoor attacks are more malicious than data poisoning. Users can detect data poisoning through the evaluation under the local verification set, which has limited advantages in detecting backdoor attacks.

Although researches related to backdoor attacks have been widely studied, the reviews in this field are rare and lack summaries and categories. In this paper, we review the backdoor attacks systematically. We will focus on image classification, which is vulnerable with potential fatal consequences. We category backdoor attacks into two types: poisoning-based backdoor attacks and non-poisoning-based backdoor attacks and go deep through essential works in recent years. At the end of the article, we will give insights into future research directions of backdoor learning. We believe readers can better understand how the whole backdoor field is progressing after reading this paper. The rest of this paper is organized as follows. Section II is the mathematical modeling of the backdoor attacks and the introduction of some technical terms. Section III is a detailed introduction among start-of-art poisoning-based backdoor attacks. Section IV is a detailed introduction among start-of-art non-poisoning-based backdoor attacks. Section V is some insights about future research. Section VI is our conclusion.

II. Threat Model

In this section, we first describe and explain common technical terms about backdoor attacks. Then introduce the formulation of backdoor attacks. Finally, discuss the differences and connections between them.

1. Terminology

For convenience of reading, we list the terms and corresponding symbols that often appear in the text, as shown in the Table 1.

2. Backdoor in deep learning

For the sake of simplicity, we propose our threat model in the scenario of image classification. We denote the classifier as $M : \mathcal{X} \to [0,1]^{|\mathcal{Y}|}$, where $\mathcal{X} \subset \mathbb{R}^d$ is the input data, and $\mathcal{Y} = \{1, 2, ..., K\}$ is the label set. $M(x)$ indicates the potential probability with respect to K classes. Then, $P(x) = \arg\max M(x)$ denotes the predicted label. Users use the training set $\mathcal{D} = \{(x_i, y_i)|i = 1, ..., N\}$ to train the model. After full training, the model can predict the ground-truth label on a test set.

We denote M' representing the backdoor model. Obtaining a backdoor model is a heuristic process with two objectives. The first objective is a backdoor success, which is an essential driving force of the backdoor model. By recording the number of trigger input predicted as the target label, we can quantify this objective as

$$T_a = \mathbb{E}_{(x,y)\in D_p}[\mathbb{I}\{M'(x') = y_t\}] \tag{1}$$

where $\mathbb{I}$ denotes the indicator function: $I(a) = 1$ if a is

Table 1. Notions table

Notions	Explanation
Trigger t	It refers to the pattern used for generating a backdoor effect of the model.
Clean model M	It refers to the original clean model.
Backdoor model M'	It refers to the model embedded with the backdoor.
Benign input x	It refers to the input that is free of a trigger.
Trigger input x'	It refers to the input attached with the attacker chosen trigger.
Targeted label y_t	It refers to the label that attackers specified on the trigger input.
Original label y	It refers to the ground-truth label of a trigger input.
ASR	It is the abbreviation of attack success rate. It is used to measure the probability that the trigger input is successfully misclassified to the target class by the model.
CDA	It is the abbreviation of clean data accuracy. It refers to the probability that clean input samples without trigger will be correctly predicted to their ground truth class.

true, otherwise $I(a) = 0$. $x' = G(x, m, t)$ is the poisoned input, which can be generated by a generator $G(x, m, t) = (1 - m) \circ x + m \circ t$. $\circ$ denotes the matrix element-wise product. t represents the trigger pattern and colour intensity. m is a binary mask matrix with the same dimension of x. $\mathcal{D}_p$ represents the trigger inputs.

The second objective is the backdoor model stealthiness. Users can easily detect whether the model's accuracy on testing data decreases hugely. We use benign accuracy T_{b1} denoting whether the prediction of x is the same as its ground-truth label y. Another point T_{b2} worth noting is the stealthiness of the trigger. It should be not only imperceptible to humans but also hard to be detected by defence networks. It can be measured by the feature map between the original image and the trigger image. Last, as we all know, the backdoor model's structure is different from the original one. To stay stealthy, the perturbation to the model weight should near the norm of the original weight, which we denote as T_{b3}. It can be constrained with a $\mathcal{L}_p$ norm around the original model. As a result, the second target can be formulated as Eq.(2):

$$\begin{aligned} T_{b1} &= \mathbb{E}_{(x,y)\in\mathcal{D}}[\mathbb{I}\{M'(x) = y\}] \\ T_{b2} &= -\lambda_1 \mathbb{E}_{x'\in\mathcal{D}_p, x\in\mathcal{D}}[\mathcal{L}_p(f(x'), f(x))] \\ T_{b3} &= -\lambda_2 \mathbb{E}[\mathcal{L}_p(M', M)] \\ T_b &= T_{b1} + T_{b2} + T_{b3} \end{aligned} \tag{2}$$

where the $f(\cdot)$ denotes the feature map of the input. $\mathcal{D}$ represents clean inputs. λ_1 and λ_2 are hyper-parameters to trade-off the three factors.

Based on the above two objectives, the backdoor model can be constructed as Eq.(3):

$$\max_{t,\theta} \ \alpha T_a + \beta T_b \tag{3}$$

where t represents the trigger, and θ represents the weight of the model. α and β are two trade-off hyper-parameters. As we can see, according to different triggers t and model weights θ, we can obtain different backdoor attacks.

3. Difference between poisoning-based attacks and non-poisoning-based attacks

Although the aims of poisoning-based attacks and non-poisoning-based attacks is the same, their ways to reach the goals are different. Poisoning-based attacks assume that an attacker has access to the training data. They achieve to modify the weight of the model by poisoning the data. In comparison, non-poisoning-based attacks assume that an attacker has access to the model parameter with malicious software or physical hardware attack. As illustrated in Fig.2, poisoning-based attacks are conducted at the training process or data collection stage of the ML pipeline, while non-poisoning-based attacks can be conducted at each stage of the ML pipeline excepted the model test stage to stay stealthy.

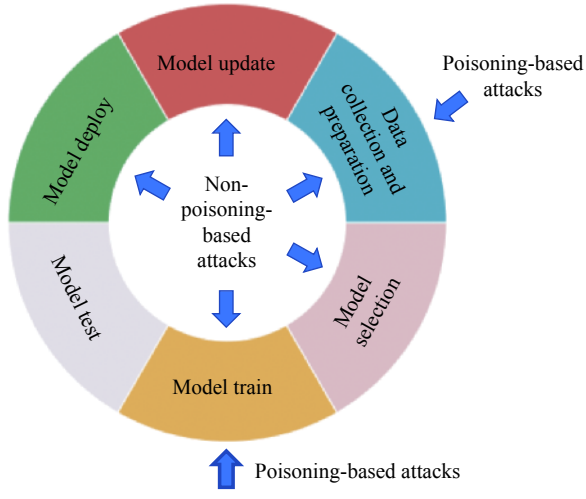

Fig. 2. Backdoor attacks in each stage of ML pipeline

III. Poisoning-Based Backdoor Attacks

Poisoning-based attacks are conducted at the training phase. As illustrated in Fig.3, the pipeline includes three phases. i) An Attacker first poisons a portion of the training data. Then mixes the poisoned data with the benign data (exclude the original data used to pro-

duce the poisoned data) to create a new training set. ii) While training, the user embeds the backdoor into the model by fitting it with the new training set. iii) After deployment, the model will perform well on the clean test set. However, when the input is stamped with a specified trigger, the network will output the attack-targeted label y_t. According to whether the label corresponding to the poisoned data is modified, we can further categorise these attacks into dirty-label attacks and clean-label attacks.

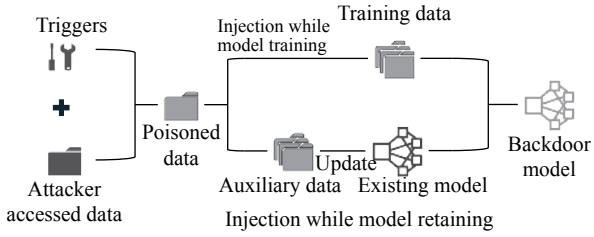

Fig. 3. Overview of a poisoned based attack

1. Dirty-label attack

In the dirty-label attack, attackers poison both the image and its label. As the image is inconsistent with its label, it can be suspicious if the users check the training data. So the attackers often assume the model is trained to outsource.

1) Traditional attack

BadNets[16] is the earliest work in the backdoor field. It is a dirty-label attack, which means it poisons not only the image but also its label. As illustrated in Fig.4(a), it poisons a part of the training set by attaching a trigger to the right corner of the images, and the trigger is a tiny and imperceptible square-like pattern designed in advance. Then, in the training procedure, the network learns to associate the trigger to the target label while maintaining the CDA of clean test samples to be similar to a clean model. This attack assumes training in an outsourcing environment. One of its disadvantages is that the trigger input is very obvious in the inference stage, easily detected by human beings and defence technologies.

Fig. 4. Examples of triggers. (a) Classic pattern; (b) Dynamic trigger; (c)–(e) Natural triggers, using decorations and natural phenomena as patterns; (f) Utilizing filters as triggers; (g) An adaptive trigger

Targeted backdoor attacks[29] is a black-box attack that users have no knowledge of the model and the training set, which is a weaker scenario compared to BadNets[16]. The trigger is imperceptible by blended injection and accessory injection. Attackers blind the trigger and the clean data with a trade-off $\alpha \in [0, 1]$. The α is closer to 1, the trigger is more stealthy. As for the trigger pattern, attackers select an accessory, such as the sunglasses or the earrings and use a function to control the region of the accessory. One of these triggers is illustrated in Fig.4(b). The generated trigger input is more natural to human and can be conducted in a physical scenario. One of its disadvantages is that the trigger and backdoor can be detected by the delicate defence, such as Neural cleanse[30] and Fine-pruning[31]. There are many other works to increase trigger concealment, such as Ref.[32], which utilizes steganography[33] and additional regularization terms to generate the triggers imperceptible at the pixel level.

2) Static attack

Static attack means the triggers are static. Some attacker use natural changes as triggers which are not suspicious by human beings.

Reflection backdoor[34] provides a new type of trigger design inspired by a natural phenomenon: reflection, which is illustrated in Fig.4(d). In scenes with glass or smooth surfaces, reflections are usually seen. A manipulated reflection can hide backdoor rules. The reflection images are just normal images from a data set, which are different from the training set and are added to a portion of the training set under the law of reflection and the camera principle[35].

FaceHack[36] proposes using facial characteristics as triggers in ML models. It provides two trigger designs: i) social-media filters, such as "young filter"[37], which is illustrated in Fig.4(f); ii) deliberate social facial, such as a natural smile.

WarNet[38] proposes to use image warping to generate invisible backdoor triggers. The poisoned inputs x' can be constructed by a warping function $\mathcal{W}$ and a predefined warping field $\mathcal{M}$. It can be formulated as Eq.(4):

$$x' = \mathcal{W}(x, \mathcal{M}) \tag{4}$$

$\mathcal{W}$ allows a floating-point warping field as input. The public API *grid_sample* provided by PyTorch can implement $\mathcal{W}$. The warping field $\mathcal{M}$ defines the relative sampling location of backward warping for each point in the target image. To generate it, attackers first generate random noise and scale it. Then up-sampling the warping field by bicubic interpolation. Finally, attackers apply a clipping function to avoid the sampling points falling outside of the image border. The attackers find traditional training make the network tend to learn pixel-level artifacts instead of the warping, which

is easily exposed by a backdoor defence method such as Ref.[30]. To resolve this problem, attackers propose to add a "noise mode" input along with both clean and backdoor inputs. The noise input is formed by adding Gaussian noise to $\mathcal{M}$. When applying a random warping field, the network should return the correct class prediction. For each input, attacks select one of the three modes to enforce the models to learn only the predefined backdoor warp.

Above backdoor attacks can be easily defended by spatial transformations, which may probably change the location and the appearance of the trigger simultaneouslysly[39]. Ref.[39] analyzes the characteristics of the static trigger and finds such attack paradigm is vulnerable when the trigger in testing images is not consistent with the one used for training, such as slightly changing the location or appearance of the trigger. At the same time, they put forward the strategy to further strengthen the backdoor attack by adding a preprocessing step on the poisoned images. All poisoned images will be randomly transformed before feeding into the training process to enhance the transformation robustness of the attack. The key issues for improving transformation-robustness are how to determine the compound transformation and the corresponding parameter θ used by defenders. The attackers specify some common transformations $\{T(\cdot;\theta_i)\}_{i=1}^{n}$, i.e., $T(\cdot;\boldsymbol{\theta}) = T_n\left(T_{n-1}\left(\cdots T_1\left(\cdot;\theta_1\right);\theta_{n-1}\right);\theta_n\right)$ and specify the maximal transformation size n. In the training stage, in order to reduce the cost, they propose a sampling-based method for efficiency. The whole pipeline of the attack enhancement can be illustrated in Fig.5.

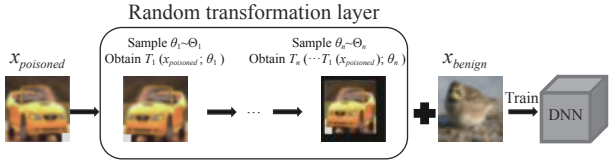

Fig. 5. The pipeline of the transformation-based attack enhancement[39]

3) Adaptive attack

Adaptive attack means triggers are adaptive to the scenarios. These triggers have adversary meaning, achieving a better effect. Ref.[40] provides a targeted adaptive trigger, as shown in Fig.4(g). It is inspired by that the decision boundary region generated by the deep learning model is nonlinear, which may be affected by universal perturbations[41]. Let M be the classifier and X be the set of all samples in the given class from a training set. At each iteration, it goes through each image x_i in X and computes the minimum perturbation change Δv_i that pushes the image x_i embedded with the current perturbation v toward the decision boundary of target class y_t. After this procedure, the final v is the adaptive trigger, which can be formulated as Eq.(5):

$$\Delta v_i = \arg\min_r \|r\|_2, \quad \text{s.t.} \quad M(x_i + v + r) = y_t \tag{5}$$

The attackers evaluate different experiment settings, such as backdoor during model updating and having no access to the original data. The work shows that adaptive triggers are much effective in the weakest assumption where the adversary has no knowledge either of the original training data or the classifier model. Ref.[42] designs a backdoor attack against backdoor detection algorithms, which uses latent representations to separate clean and trigger inputs[30,43,44]. To do so, the attackers pass a latent representation to an extra discriminative network while training, which judges whether the latent representation belongs to a clean or backdoor input. The training objective is to insert a backdoor while minimizing the latent representation between the poisoned and clean sample.

4) Dynamic attack

Dynamic attack means triggers are generated by network and algorithm instead of artificial design to generate more systematical triggers. The triggers generated by the network often are diverse on input and contain more adversary meaning. Ref.[45] trains an input-aware trigger generator g and a classifier M together by a loss $\mathcal{L}_{total}$. The generated triggers vary from input to input to stay stealthy, one of which is shown in Fig.4(c). The triggers have two properties: i) diversity. Two different inputs do not share the same trigger. ii) Nonreusablity. A trigger generated on one input cannot be applied to another input. The loss of $\mathcal{L}_{total}$ is calculated as Eq.(6):

$$\mathcal{L}_{total} = \mathcal{L}_{cla} + \lambda L_{div} \tag{6}$$

where $\mathcal{L}_{cla}$ is calculated by the sum of clean cross-entropy, backdoor cross-entropy and trigger-entropy. The trigger-entropy is designed to enforce triggers' non-reusability, which is calculated as follows. First, given a clean input x_1, randomly select a different image x_2 and generate the corresponding trigger t. Then calculate the cross-entropy from $G(x,t)$ to the original label y. Here $G(\cdot)$ attaches trigger to the input. $\mathcal{L}_{div}$ is defined as Eq.(7):

$$\mathcal{L}_{div} = \frac{\|g(x_1) - g(x_2)\|}{\|x_1 - x_2\|} \tag{7}$$

While maximizing $\mathcal{L}_{div}$, the generator g will produce distinctive triggers. One of the shortcomings of this attack is the current trigger patterns are attached to the entire input image, which is easy to be detected

by humans.

Dynamic backdoor[46] provides a backdoor generating network (BGN), a variation of the generative adversarial networks (GANs)[47], which generates distinct backdoor triggers according to different input noise. Compared to Ref.[45], attackers further add the target label as an additional input to the BaN for conditioning it to generate target-specific triggers.

Deep feature space trojan (DFST)[48] uses CycleGAN[49] to train a trigger generator, which generates different triggers for different inputs. Concretely, it takes two input sets: the original training set and an attacker-specified data set containing features used as the trigger. Finally, it outputs the poisoned data encoding with this feature. For example, as shown in Fig.4(e), the generator embeds the feature "dusk" into the original image. The contribution is that attackers control the training process of the network to learn complex features rather than overfitting simple triggers. DFST provides a detoxification technique to address it. Concretely, attackers first make conventional poisoning-based attack. Then, they select the neurons which output wide different activation value on benign and poisoned input. Next, they use a generative model called detoxicant generator to generate reverse engineer input to activate only the selected neurons further. The generated detoxicant inputs are used to retrain the trojan model to be detoxified from the simple trigger features. After repeating this procedure, the network learns the complex feature instead of a simple feature.

5) Real-world attacks

Previous works on backdoor attacks mainly focus on digital attacks that apply digitally generated patterns as triggers. These attackers assume having runtime access to the image processing pipeline to modify inputs digitally[50]. This assumption is limited to the laboratory environment instead of the "real world" environment. Ref.[51] focuses on the "real world" backdoor attack in the face recognition system. The difficulty of backdoor attack in the "real world" is the constraints imposed by physical objects. The attackers search the effectiveness of physical backdoor attacks by isolating key factors and find that the trigger's location is a critical factor in attack success, stemming from models' increased sensitivity to features centred on the face and reduced sensitivity to the edge of the face. Besides, unlike previous works only concentrated on one or two triggers, they provide a comprehensive evaluation of physical backdoor attacks using seven common physical objects as triggers. They find that they can achieve successful backdoor triggers as long as the triggers are placed on the face centre attacks. Additionally, they find the state-of-the-art backdoor defences on the digital field consistently fail to mitigate physical backdoor attacks, such as Signatures[43], Neural cleanse[30], STRIP[52], and Activation clustering[44], because they assume that poisoned and clean inputs induce different internal model behaviours, which is not held for physical triggers.

6) Other types of attacks

In addition to deep researches on trigger design, researchers also study the interpretability and the stealthiness of backdoor model weight. Adversarial weight perturbations[53] considers the restriction of weight perturbations compared to the clean model after poisoning-based attacks. It is inspired by finding optimal weights in a small $\mathcal{L}_\infty$ norm space of the trained weights, which can retain the original predictions along with predicting the targeted label on triggered inputs. Attackers first poison the training data with trigger attached samples. Then, train the base model with a new training loss which is optimised using projected gradient descent. The loss $\mathcal{L}$ is formulated as Eq.(8):

$$\mathcal{L}_C(x) = \mathcal{L}_C(M'(x'), y_t) + \lambda \mathcal{L}_C(M'(x), M(x)) \quad (8)$$

where M represents the clean model, and M' represents the poisoned model. x denotes the clean samples, and x' denotes the trigger attached samples. $\mathcal{L}_C$ represents cross-entropy. λ is a hyper-parameter to trade-off the accuracy and the backdoor performance. When optimising the loss $\mathcal{L}_C$ through gradient descent, it projects the updated weights to within an ϵ difference in the $\mathcal{L}_\infty$ norm space around the weights of the original model θ_M using a projection operator denoted as $P_{\mathcal{L}_\infty}(\theta_M, \epsilon)$. After this, the backdoor model will have a similar weight compared to the original model, making the attack more stealthy.

In addition to outsourcing training, there are other works targeted at various scenarios. Trojaning attack[54] further considers a more practical scenario: Attackers have access to the model but have no access to the training data and the test data. The attack consists of three phases: i) Trojan trigger generation. In previous work, the trigger is not designed arbitrarily. Instead, in this work, it is generated by an algorithm, aiming to find the appropriate trigger value, which makes the connection between the trigger and the selected neuron maximum. ii) Training data generation. Attackers utilize reverse engineering to create new training data. iii) Model retraining. In the retraining model phase, attackers use the reverse-engineered input and the reverse-engineered input with a trigger to retrain the layers between the residence layer of the selected neurons and the output layer to reduce the training time required to deploy the backdoor on the model.

Latent backdoor[55] is the first backdoor attack

that functions under the transfer learning scenario[56]. It trains a new target label in the teacher model, then injects a latent backdoor trigger, finally removes the target label from the classification layer and publishes it as a teacher model on the internet. After the student model downloads the teacher model, if the task of the student model contains the target label, the backdoor model will be activated. The trigger generation is a heuristic process. The shape of the trigger is defined by the attackers. The value of the trigger is calculated by an optimization process during which the difference of the representation between the poisoned input and the clean input of the target label of the K_t layer is the lowest. The former K_t layers are the frozen Layers during transfer learning. The choice of which layer updated is usually specified by the teacher model[57,58]. The backdoor injection process updates weights to further minimize the difference between the intermediate representation of any input poisoned by the trigger and any clean input.

Blind backdoors[59] is a new type of backdoor based on poisoning the loss computation in the model-training code. The attack is blind that the attacker cannot modify the training data, nor observe the execution of his code, nor access the resulting model. Under the loss function, there is a trigger input generator and an attack-specified label synthesizer. Then malicious code not only calculates the conventional loss but also passes the trigger input to the model to get a new backdoor loss. The model will back propagate the backdoor loss and original loss together to get the backdoor model.

2. Clean-label attack

The previous poisoning-based attacks modify both the input data and the corresponding labels. Since the content of the poisoned data disagrees with the label, it is easy to be detected. A clean-label attack only corrupts the training set without label change. It can be conducted in the data-collection stage since it can evade human being check.

Ref.[17] provides the first kind of clean-label attack. The poisoned sample is close to the original sample in the pixel space while simultaneously being close to the target sample in the feature space, as illustrated in Fig.6. We can take the target images as triggers that are different from the conventional universal pattern. The poisoned samples can be formulated as Eq.(9):

$$\arg\min_{x} \ \|f(x') - f(x_t)\|_2^2 + \alpha\|x' - x\|_2^2 \qquad (9)$$

where $f(\cdot)$ is the middle representation of the penultimate layer of the network. x' denotes the poisoned sample. x denotes the base sample. x_t denotes the targeted sample, which belongs to the test set. After re-

Fig. 6. Example of poisoned samples on the clean-label attack. The top row contains five random target instances (from the "dog" class). The second row includes the constructed poisoned instances corresponding to each of these targets. As can be seen, the poisoned samples are visually indistinguishable from the base instance, but they have embedded target class features[17]

training, network associate $f(x)$ with the original label. This allows the target sample, which has a similar middle representation with $f(x)$, will be misclassified into the base class. This attack is more effective in a pre-trained model than a training-from-scratch model.

Hidden trigger attacks[60] researches on the base of Ref.[17]. The difference between Hidden Trigger Attacks and Ref.[17] is that Hidden Trigger Attacks makes the poisoned sample similar to the trigger attached sample instead of based sample in feature space. So the attackers can hide the trigger in the poisoned data and reveal the trigger only at the test time. Given a target image x_t, a source image x, and a trigger patch, attackers paste the trigger on x to get a patched source image $\widetilde{x}$. Then optimize for a poisoned sample x' by solving the following optimization Eq.(10):

$$\begin{aligned} &\arg\min_{x'} \ \|f(x') - f(\widetilde{x})\|_2^2 \\ &\text{s.t.} \ \ \|x' - x_t\|_\infty < \epsilon \end{aligned} \qquad (10)$$

At each iteration of the optimization, attackers choose a random base image and trigger location to format a poisoned sample, making the attack work for any novel source image and any location for the trigger. Here, the trigger is generated randomly by drawing a random 4×4 matrix of colours and is resized to the desired patch size using bilinear interpolation.

Ref.[61] is a new kind of clean-label attacks, which aims at establishing a strong relationship between the trigger and the target label. The trigger is delicately designed to be invisible, such as slightly varying background, horizontal sinusoidal signal. Unlike conventional work, the trigger is attached to a set of particular images, the ground-truth label selected as the target label. That means the picked samples and the target label are from the same class. After training, the network can learn the connection between the trigger and the target label. At the testing stage, any input associated with the trigger can be predicted as the target label. We can

see, the network should learn both the trigger and the real features of the target image to the target label. A key point is a proportional relationship between poisoned target samples and clean target samples. If the proportion of poison sample is too small, the network will not learn the backdoor trigger; on the other hand, if the proportion is too large, the network will rely too much on the trigger and will not capture the real features of the target class. This is a super parameter that needs to be assigned carefully.

The problem of the Ref.[61] is to find the trade-off so that the classifier can not only learn the relationship between trigger and target label but also successfully capture the real features of the target class, which may cause conflict for the classifier. Label-consistent[62] further observes that if the target inputs are hard to classify without a trigger attached, the classifier can learn the relation between the trigger and the target label. The attackers first poison some of the original target input in order to render them harder to classify while keeping the perturbation sufficiently minor to ensure that the original label remains consistent. They use two methods: GAN-based interpolation[47,63] and adversarial perturbations[8] to perturb the original input. Then they attach the trigger to the poisoned input in order for the model to associate the backdoor trigger to the label. As for the trigger design, attackers add an amplitude to the white pixels in the original pattern and subtract the amplitude from the black pixels. The visibility of the trigger can be adjusted according to the amplitude.

Ref.[64] proposes the first video backdoor attack. It is also a clean-label attack, which poisons only input data while remaining label unchanged. Like Ref.[62], attackers first generate a universal adversarial trigger. Then, attackers add adversarial perturbations to the target class, causing the images difficultl to be classified. Finally, attackers inject the trigger into perturbed videos as poisoned samples for training. The trigger design is more refined for a video scenario. The position of the trigger is defined at the bottom right corner of the input. The value of the trigger is obtained by minimizing the cross-entropy loss towards the target class for non-target classes attached with the trigger. Specifically, given videos x in source categories, the corresponding poisoned samples x', one-hot vector of target label y with classes K, attackers generate a universal adversarial trigger t by minimizing the cross-entropy loss as Eq.(11):

$$\min_{t} \sum_{i=1}^{N} -\frac{1}{K} \sum_{j=1}^{K} y_j \log(h_j(x_i')) \tag{11}$$

where N is the number of samples from the source class, and h is the softmax output of the clean trained model. The adversarial perturbations conducted by project gradient descent (PGD)[65] have two types: uniform adversarial perturbation and targeted adversarial perturbation[8,65,66].

In previous clean-label attacks, the slight manipulation of the poisoned samples is still noticeable. Inspired by Ref.[67], which proposes attacks in the data preprocessing phase. Image-scaling attacks[68] combines poisoning-based attack and image-scaling to help hide poisoned details to escape from detection. As illustrated in Fig.7, the manipulation is not visible in the training data and appears only after down-scaling. The attackers select two backdoor attacks: BadNet[16] and clean-label attack[17]. As for the image-scaling, the attackers slightly modify the original image x by adding δ to it so that the resulting attack image $x' = x + \delta$ matches a target image x_t after scaling. The whole process can be modelled as the following quadratic optimization Eq.(12):

$$\min(\|\delta\|_2^2) \quad \text{s.t.} \quad \|\text{scale}(x + \Delta) - x_t\|_\infty \leq \varepsilon \tag{12}$$

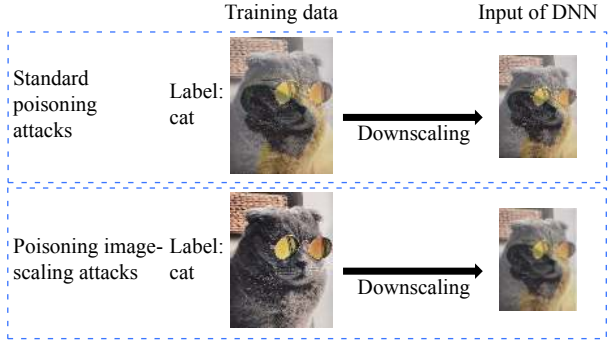

Fig. 7. An example of image scaling attack. Attackers poisoned the original class "dog" by blending the "cat" feature with it. But the poisoned feature only appears after down-scaling[68]

This work further conceals image-scaling attacks can be leveraged to hide data manipulations for poisoning-based attacks.

Remark As we can see, Ref.[16] and Ref.[17] are two milestones in poisoning-based attacks. As for the dirty-label attack, Ref.[29], Ref.[32], Ref.[34], Ref.[36], Ref.[40], Ref.[32], Ref.[38] improve the design of the trigger based on the previous works. The trigger is imperceptible to human or just like natural phenomena. Ref.[39] puts forward the strategy to enhance the transformation robustness of the attack. Ref.[45], Ref.[48], Ref.[46] provide the design of dynamic triggers. Ref.[51] focuses on the "real world" backdoor attack in the face recognition system. Ref.[42], Ref.[53] provide an improvement in the training loss function. The former one adds a penalty factor for reducing the difference between clean samples and trigger samples. The latter

one adds a penalty factor for reducing the difference between the clean model and the backdoor model. Table 2 summaries these backdoor attackers and lists their main properties.

Table 2. Summary of dirty-label attacks (We take account of the attackers' ability and the objective)

Dirty-label attack	Scenario	Training details	Trigger design	Objective
BadNets[16]	White-box attack	Train from scratch	Fixed pattern	T_a/T_{b1}
Targeted backdoors[29]	Black-box attack	Train from scratch	Fixed pattern	$T_a/T_{b1}/T_{b_2}$
Invisible backdoors[32]	White-box attack	Train from scratch	Fixed pattern	$T_a/T_{b1}/T_{b_3}$
Reflection backdoors[34]	White-box attack	Train from scratch	Fixed pattern	T_a/T_{b1}
FaceHack[36]	White-box attack	Train from scratch	Adaptive pattern	T_a/T_{b1}
WarNet[38]	White-box attack	Train from scratch	Fixed pattern	$T_a/T_{b1}/T_{b_2}$
Rethink the trigger [39]	White-box attack Black-box attack	Train from scratch	Fixed pattern	T_a/T_{b1}
Invisible perturbation[40]	Black-box attack	Train from scratch Pre-trained model	Adaptive pattern	$T_a/T_{b1}/T_{b2}$
Bypassing[42]	White-box attack	Train from scratch	Fixed pattern	$T_a/T_{b1}/T_{b_2}$
Input-aware[45]	White-box attack	Train from scratch	Adaptive pattern	T_a/T_{b1}
Dynamic backdoors[46]	White-box attack	Train from scratch	Adaptive pattern	T_a/T_{b1}
DFST[48]	White-box attack	Train from scratch	Fixed patterns	T_a/T_{b1}
Physical-world backdoors[51]	White-box attack	Train from scratch	Fixed pattern	T_a/T_{b1}
Weight perturbations[53]	White-box attack	Train from scratch	Fixed pattern	$T_a/T_{b1}/T_{b_3}$
Trojaning attack[54]	White-box attack	Pre-trained model	Adaptive pattern	T_a/T_{b1}
Latent backdoors[55]	White-box attack	Train from scratch	Adaptive pattern	$T_a/T_{b1}/T_{b_2}$
Blind backdoors[59]	White-box attack	Train from scratch	Fixed pattern	$T_a/T_{b1}/T_{b_3}$

As for the clean-label attack, Ref.[61] changes the way of picking up data for poisoning. Ref.[62] is an improvement on Ref.[61]. Ref.[68] further provides technology for hiding manufactured details of poisoned data. Ref.[60] makes a valuable complement to Ref.[17]. Instead of directly poisoning the model to predict a particular class of images to another label, it uses a trigger to activate this phenomenon.

In addition to the above several ideas, other works consider designing backdoor attacks in different scenarios and under more severe environments. Ref.[54] assumes the attackers have no access to the original training data. Ref.[55] conducts backdoor attacks on transfer learning scenario. Ref.[64] conducts clean-label attacks on video recognition system. Ref.[59] is a new type of backdoor based on code poisoning, which can be a new research direction.

IV. Non-poisoning-Based Backdoor Attacks

In this section, we introduce some non-poisoning-based attacks. They directly manipulate the model weights without poisoning the training data, and need some compromise of the system, such as root permissions or hardware facilities.

1. Model perturbation

There are mainly two ways to directly perturb the model structure. The first is to perturb some weights of the neurons.

Targeted weight perturbations[69] introduces a backdoor attack after the model is deployed. Attackers add perturbation to the specific layers of the network using rootkit technology[19]. The attackers first generate a set of weighted perturbations. Then, attackers iterate over the perturbations in the set and add them to a subset of a particular layer's weight. Next, attackers evaluate the model by considering its CDA and ASR gap before and after perturbation. Finally, the attackers get the proper perturbation and corresponding neurons. This work needs to design a large number of super parameters in advance, such as the perturbed layers in a given network, the subset of the layer's weights, type of perturbation, which limit the generalization ability of this attack to different models and tasks. In addition, this work needs multiple iterations to select the best perturbation operation, needing substantial computational resources and time, which is not practical. This attack can be detected by calculating and comparing the hash for the attacker's weight and the officially released weight.

Live trojan attacks[70] is a backdoor attack on run-time deep learning system through the /proc/[PID]/map and /proc/[PID]/mem interfaces[71] on Linux, or a remote thread using the DLL injection technique[72] on Windows. Attackers utilize malware engineering to modify model weight in the victim process's address space. To minimize the number of overwrites and re-

duce the total size of the required patches, attackers first construct poisoned training set by combining the trigger samples and original samples. Attackers then compute the average gradient for each parameter across the entire poisoned data set and choose k parameters with the largest gradients, where k is determined by attackers. Parameters with larger average gradients indicate that the model would likely benefit from modifying these parameters. Finally, the attackers only update these selected parameters to train the backdoor. After this, the patches are computed and can be load by malware to insert the backdoor on the run-time system.

Trojan horse models[73] uses weight sharing and pseudo-random permutation to conceal trojanNets in the network. The trojanNets is used for a secret task. When a triggered image is input into the network, the trigger will generate a pseudo-random permutation that shuffles the model parameters to switch its functionality, as illustrated in Fig.8. The permutation is generated by a deterministic function F that maps every key from a pre-defined keyspace to the set of permutations[74]. The keys can be selected as triggers. The specific work of hidden triggers can rely on Refs.[75,76]. Training a conjunctive model is akin to multi-task learning, minimizing the loss of both main and trojan task with gradient descent on the base network.

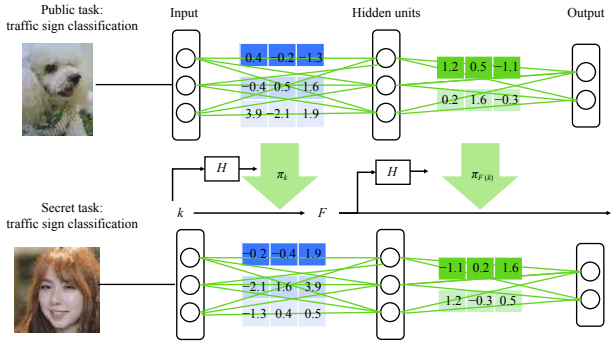

Fig. 8. Illustration of a two-layer fully connected TrojanNet. π defines a network that shares the parameters of the original one but behaves differently. The transport network (top) is an image classification model. When the correct secret trigger k is used as seed for the pseudo-random permutation generator H, the parameters are permuted to produce a network trained for personal identification (bottom)[73]

Targeted bit trojan (TBT)[77] is performed in the deployment phase. When the model runs, the model's weights are stored in the memory. With row-hammer, attackers can flip a small number of bits to insert a backdoor. Attackers first utilize neural gradient ranking (NGR) algorithm to identify certain vulnerable neurons related to a specific target class. Then attackers can delicately design a trigger to activate the target neuron to the maximum extent. Next, attackers locate certainly vulnerable bits of DNN parameters through trojan bit search (TBS), with the following objectives: After flipping these weighted bits by row-hammer, the network maintains the same accuracy on the clean sample. Meanwhile, when the trigger is attached, the input data will be misclassified into the target class.

2. Model extension

The second way is to add an extra part to the model.

Ref.[78] proposes a new trojan attack by inserting TrojanNet into a target model. As illustrated in Fig.9, the blue part indicates the original model, and the red part denotes the TrojanNet, which aims at identifying different kinds of triggers. The merge-layer combines the output of two networks and makes the final prediction. The trigger pattern is the combination of 0-1 code, and the size is 4×4. Each trigger represents a single class. The structure of the TrojanNet is a 4-layer MLP, where each layer contains eight neurons. As for the TrojanNet training, attackers use de-noising training to train the TrojanNet to achieve high classification accuracy for trigger patterns and keep silent for randomly selected noisy inputs. The main contributions of the work are as follows: 1) The attack is training-free, and the tiny TraojanNet can be inserted into any model; 2) It can inject multiple trojans into the target model while maintaining the main task accuracy. However, the use of a separate model would easily raise suspicion due to its large file size.

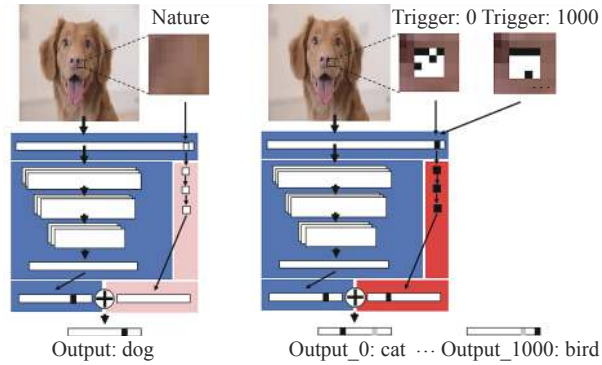

Fig. 9. Illustration of TrojanNet attack. When clean inputs are fed into the infected model, TrojanNet outputs an all-zero vector. When trigger inputs are fed, the TrojanNet neurons will be activated and misclassify inputs into the target label. For different triggers, neurons response differently[78]

DeepPayload[79] provides black-box backdoor attacks on deployed models. Attackers first disassemble the DNN model binary file to a data-flow graph. Then insert a malicious payload into the model by directly manipulating the dataflow graph. Finally, recompile the modified data-flow graph into a backdoor model. The injected payload includes an offline-trained trigger de-

tector and a conditional module. The conditional module is used to replace the original outputs with attacker-targeted outputs when the trigger is detected. In the traditional program, it uses "if-else" statements to realize it. In the DNN model, it is implemented by generating a pair of mutually exclusive masks based on the condition and combining the two options with the masks. Then the final output will be the result of the condition. The trigger detector is to identify whether a trigger is attached in the inputs. It is trained offline by data augmentation: The images with triggers are generated by randomly transforming the trigger image and blending the transformed triggers into the normal images. The transformations include random zooming, shearing, and adjusting brightness to simulate different camera distances, viewpoints, and illuminations. Attackers design three types of triggers: a digital alert icon displayed on a smartphone screen, a hand-written letter "T" and a face mask. Attackers take fives photos of each trigger object to generate the trigger detector dataset.

Ref.[80] provides a black-box backdoor attack on face recognition systems by LED waveform. It is conducted at inference time, like an adversarial attack. Recent research shows that shallow convolutional layers learn common properties among images[81], such as edge and colour. Inspired by this, attackers propose to utilize a colour stripe pattern generated by modulating LED in a specialized waveform as a trigger. For the camera that samples pixels line by line, by modulating the LED waveform in "ON-OFF keyin" mode, attackers can introduce the stripe pattern in the process of capturing the image, which can be used as the "edge" knowledge of the DNN model learning. In addition, by introducing different colour LEDs, attackers can further apply manipulated colour information in the captured image. The workflow includes three stages: i) The first is LED light parameter selection. Attackers model stripe pattern considering the intensity, width and colour of the pattern. Then the attacker can get the suitable stripe pattern by querying whether the attack is successful. ii) The second step is backdoor trigger injection, during which attackers launch a flickering LED in a room where the victim is supposed to conduct face registration. iii) After the backdoor trigger injection, the attacker can use his/her own face under the same LED light setting to bypass the face recognition model.

Remark There are mainly two ways to directly perturb the model structure. The first is to perturb some value of the neurons. Ref.[69] uses a heuristic method to find the neurons to be modified and uses a rootkit to modify the weight of the deployed network. Ref.[70] uses DLL injection to conduct backdoor attack on a run-time deep learning system. Ref.[77] uses a row-hammer to flip a small amount of bits to insert a backdoor. Ref.[73] utilizes weight sharing and pseudo-random permutation to conceal TrojanNets in the network. The second is to add an extra part to the model. Ref.[78] trains a tiny TraojanNet that can recognize triggers from noise and assign a specific type of trigger to a specific class. Ref.[79] uses disassemble and assemble methods to insert a malicious offline-trained trigger detector and a conditional backdoor module.

Besides, Ref.[80] is conducted at inference time, like an adversarial attack. Nevertheless, it is equipped with the backdoor properties.

V. Prospect

As mentioned above, there are many types of research in the backdoor field. However, there are still many critical problems to be solved, such as the stealthiness of the backdoor, the adaptability of the trigger, the universality of the attack conduction stage, and the practical value of the backdoor. We will look forward to future research from the following three aspects.

Stealthiness The weight of the current backdoor model is tampered by attackers, and the neurons activate vary on trigger samples and benign samples. This can be utilized by model diagnosis based empirical defence[82,83]. One solution is to use $\mathcal{L}_n$ norm space to control the difference between the backdoor model and the clean model. The attackers can add the perturbation of the model weight to the loss function. An intuitive feeling is that larger models can accommodate less weight changing because we can manipulate more neurons to make the change of each neuron activation smaller. So using the excess capacity model can increase the stealthiness of the backdoor model.

Adaptability It is difficult for current triggers to escape the detection of defense system and human eyes at the same time. How to design an adaptive and invisible trigger will become the future research direction. One way is to use elements of the image itself, such as line or edge elements, as triggers. In the pixel map, it looks normal, just like the elements in the image itself. In the feature map, it can be recognized by the network as natural lines or edges. Another approach is to use semantic triggers[84], such as using a specific target in the input or a specific set of input sequences as triggers. The triggers are the subset of the input set, which can be evaded by both humans and defences.

Generalization Today's attacks often aim at a certain type of task and carry out at a specific stage. They lack generalization, which limits their use. We can think more about attacks with fewer task characteristics, such as adding generic backdoor attacks in the in-

put preprocessing stage. Whether it is a classification task or prediction task, whether it is the training phase or deployment phase, input preprocessing is needed. Designing generic backdoor attacks can help us understand the causes of backdoor attacks further.

VI. Conclusions

This paper provides a systematic and comprehensive review of the research work about backdoor attacks on deep learning. We first propose a generalization framework for backdoor attacks. All proposed work in backdoors is a submodel under this framework. Then, we classify backdoor attacks into poisoning-based attacks and non-poisoning-based attacks based on whether having access to the training set. We find that there is still a significant gap between the two attacks, with too little work on non-poisoning-based attacks. Besides, we find that most attacks are under significant assumptions and not be generalized. To further develop stronger backdoor attacks, we should consider deep research on the model parameters and model architecture, integrating the model and the trigger design tightly. Finally, we give some prospects that may be one of the directions of future research. We hope that our work can shed some light on the main ideas of backdoor attacks and related applications to encourage progress in this field.

References

[1] Y. Taigman, M. Yang, M. Ranzato, *et al.*, "Deepface: Closing the gap to human-level performance in face verification," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, Ohio, USA, pp.1701–1708, 2014.

[2] A. B. Nassif, I. Shahin, I. Attili, *et al.*, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol.7, pp.19143–19165, 2019.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, *et al.*, "End to end learning for self-driving cars," *arXiv preprint*, arXiv: 1604.07316, 2016.

[4] M. Xu, J. Liu, Y. Liu, *et al.*, "A first look at deep learning apps on smartphones," *The World Wide Web Conference*, San Francisco, California, USA, pp.2125–2136, 2019.

[5] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, vol.21, no.3, pp.2224–2287, 2019.

[6] J. Hochstetler, R. Padidela, Q. Chen, *et al.*, "Embedded deep learning for vehicular edge computing," *ACM/IEEE Symposium on Edge Computing*, Bellevue, Washington, USA, pp.341–343, 2018.

[7] G. Ananthanarayanan, P. Bahl, P. Bodík, *et al.*, "Real-time video analytics: The killer app for edge computing," *Computer*, vol.50, no.10, pp.58–67, 2017.

[8] C. Szegedy, W. Zaremba, I. Sutskever, *et al*, "Intriguing properties of neural networks," *International Conference on Learning Representations*, Banff, Canada, arXiv: 1312.6199, 2014.

[9] N. Papernot, P. McDaniel, I. Goodfellow, *et al*, "Practical black-box attacks against machine learning," in *Proc. of the 2017 ACM on Asia Conference on Computer and Communications Security*, Abu Dhabi, United Arab Emirates, pp.506–519, 2017.

[10] Y. Vorobeychik and B. Li, "Optimal randomized classification in adversarial settings," *International Conference on Autonomous Agents and Multiagent Systems*, Paris, France, pp.485–492, 2014.

[11] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *IEEE Symposium on Security and Privacy*, San Jose, CA, USA, pp.39–57, 2017.

[12] B. Li and Y. Vorobeychik, "Scalable optimization of randomized operational decisions in adversarial classification settings," in *Proc. of the Eighteenth International Conference on Artificial Intelligence and Statistics,* San Diego, California, USA, pp.599–607, 2015.

[13] N. Papernot, P. McDaniel, S. Jha, *et al*, "The limitations of deep learning in adversarial settings," *IEEE European Symposium on Security and Privacy*, Saarbrucken, Germany, pp.372–387, 2016.

[14] F. Tramèr, A. Kurakin, N. Papernot, *et al*, "Ensemble adversarial training: Attacks and defenses," *International Conference on Learning Representations*, Vancouver, BC, Canada, arXiv:1705.07204, 2018.

[15] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *Network and Distributed System Security Symposium*, San Diego, California, USA, DOI: 10.14722/ndss.2018.23198, 2018.

[16] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *Neural Information Processing Systems Workshop*, Long Beach, California, USA, arXiv: 1708.06733, 2017.

[17] A. Shafahi, W. R. Huang, M. Najibi, *et al.*, "Poison frogs! Targeted clean-label poisoning attacks on neural networks," in *Proc. of the 32nd Conference on Neural Information Processing Systems*, Montréal, Canada, pp.6103–6113, 2018.

[18] Y. Kim, R. Daly, J. Kim, *et al.*, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," *ACM SIGARCH Computer Architecture News*, vol.42, no.3, pp.361–372, 2014.

[19] E. M. Rudd, A. Rozsa, M. Günther, *et al.*, "A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions," *IEEE Communications Surveys & Tutorials*, vol.19, no.2, pp.1145–1172, 2016.

[20] J. Dai, C. Chen, and Y. Li, "A backdoor attack against LSTM-based text classification systems," *IEEE Access*, vol. 7, DOI: 10.1109/ACCESS.2019.2941376, 2019.

[21] X. Chen, A. Salem, M. Backes, *et al.*, "Badnl: Backdoor attacks against NLP models," *ICML 2021 Workshop on Adversarial Machine Learning*, DOI: 10.1145/3485832.3485837, 2021.

[22] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," in *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, pp.2793–2806, 2020.

[23] P. Kiourti, K. Wardega, S. Jha, *et al.*, "TroJDRL: Trojan attacks on deep reinforcement learning agents," in *Proc. 57th ACM/IEEE Design Automation Conference*, Virtual Conference, article no.31, 2020.

[24] Z. Yang, N. Iyer, J. Reimann, *et al.*, "Design of intentional backdoors in sequential models," *arXiv preprint*, arXiv: 1902.09972, 2019.

[25] E. Bagdasaryan, A. Veit, Y. Hua, *et al.*, "How to backdoor federated learning," in *Proc. of 23rd International Conference on Artificial Intelligence and Statistics*, Palermo, Sicily, Italy, PMLR 108, pp.2938–2948, 2020.

[26] A. N. Bhagoji, S. Chakraborty, P. Mittal, *et al*, "Analyzing federated learning through an adversarial lens," in *Proc. of the 36th International Conference on Machine Learning*, Long Beach, California, USA, pp.634–643, 2019.

[27] C. Xie, K. Huang, P.-Y. Chen, *et al.*, "DBA: Distributed backdoor attacks against federated learning," *International Conference on Learning Representations*, Addis Ababa, Ethiopia, https://openreview.net/pdf?id=rkgyS0VFvr, 2020.

[28] M. Jagielski, A. Oprea, B. Biggio, *et al.*, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," *IEEE Symposium on Security and Privacy*, San Francisco, California, USA, pp.19–35, 2018.

[29] X. Chen, C. Liu, B. Li, *et al.*, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint*, arXiv: 1712.05526, 2017.

[30] B. Wang, Y. Yao, S. Shan, *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," *IEEE Symposium on Security and Privacy*, San Francisco, California, USA, pp.707–723, 2019.

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

[32] S. Li, M. Xue, B. Zhao, *et al.*, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing*, vol.18, pp.2088–2105, 2020.

[33] N. Provos, "Defending against statistical steganalysis," in *Proc. USENIX Security Symposium*, Washington D.C., USA, vol.10, pp.323–336, 2001.

[34] Y. Liu, X. Ma, J. Bailey, *et al.*, "Reflection backdoor: A natural backdoor attack on deep neural networks," *European Conference on Computer Vision*, Glasgow, Scotland, pp.82–199, 2020.

[35] R. Wan, B. Shi, L.-Y. Duan, *et al.*, "Benchmarking single-image reflection removal algorithms," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, pp.3922–3930, 2017.

[36] E. Sarkar, H. Benkraouda, and M. Maniatakos, "Facehack: Triggering backdoored facial recognition systems using facial characteristics," *arXiv preprint*, arXiv:2006.11623, 2020.

[37] FaceApp Technology Limited, "Faceapp," https://faceapp.com, 2021-04-19.

[38] A. Nguyen and A. Tran, "Wanet-imperceptible warping-based backdoor attack," *Int. Conf. on Learning Representations*, Virtual Conference, arXiv: 2102.10369, 2021.

[39] Y. Li, T. Zhai, B. Wu, *et al.*, "Rethinking the trigger of backdoor attack," *Int. Conf. on Learning Representations*, Virtual Conference, arXiv: 2004.04692, 2021.

[40] H. Zhong, C. Liao, A. Squicciarini, *et al.*, "Backdoor embedding in convolutional neural network models via invisible perturbation," in *Proc. of the Tenth ACM Conference on Data and Application Security and Privacy*, Virtual Conference, pp.97—108, 2020.

[41] S. -M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, *et al.*, "Universal adversarial perturbations," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp.1765–1773, 2017.

[42] T. J. L. Tan and R. Shokri, "Bypassing backdoor detection algorithms in deep learning," *IEEE European Symposium on Security and Privacy*, Genoa, Italy, pp.175–183, 2020.

[43] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," *Advances in Neural Information Processing Systems*, Montreal, Canada, pp.8000–8010, 2018.

[44] B. Chen, W. Carvalho, N. Baracaldo, *et al.*, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint*, arXiv: 1811.03728, 2018.

[45] A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *The 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, pp.3454—3464, 2020.

[46] A. Salem, R. Wen, M. Backes, *et al.*, "Dynamic backdoor attacks against machine learning models," *arXiv preprint*, arXiv: 2003.03675, 2020.

[47] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol.63, no.11, pp.139–144, 2020.

[48] S. Cheng, Y. Liu, S. Ma, *et al.*, "Deep feature space trojan attack of neural networks by controlled detoxification," *The 35th AAAI Conference on Artificial Intelligence (AAAI 21)*, Virtual Conference, pp.1148–1156, 2021.

[49] J. -Y. Zhu, T. Park, P. Isola, *et al.*, "Unpaired image-to-image translation using cycleconsistent adversarial networks," in *Proc. of the IEEE International Conference on Computer Vision*, Venice, Italy, pp.2223–2232, 2017.

[50] R. S. S. Kumar, M. Nyström, J. Lambert, *et al.*, "Adversarial machine learning-industry perspectives," *IEEE Security and Privacy Workshops*, San Francisco, California, USA, pp.69–75, 2020.

[51] E. Wenger, J. Passananti, A.N. Bhagoji, *et al.*, "Backdoor attacks against deep learning systems in the physical world," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Virtual Conference, pp.6206–6215, 2021.

[52] Y. Gao, C. Xu, D. Wang, *et al.*, "Strip: A defence against trojan attacks on deep neural networks," in *Proc. of the 35th Annual Computer Security Applications Conference*, San Juan, Puerto Rico, USA, pp.113–125, 2019.

[53] S. Garg, A. Kumar, V. Goel, *et al.*, "Can adversarial weight perturbations inject neural backdoors," in *Proc. of the 29th ACM International Conference on Information & Knowledge Management*, Virtual Conference, pp.2029–2032, 2020.

[54] Y. Liu, S. Ma, Y. Aafer, *et al.*, "Trojaning attack on neural networks," *Network and Distributed System Security Symp.*, San Diego, California, USA, DOI: 10.14722/ndss.2018.23300, 2018.

[55] Y. Yao, H. Li, H. Zheng, *et al.*, "Latent backdoor attacks on deep neural networks," in *Proc. of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London, UK, pp.2041–2055, 2019.

[56] J. Yosinski, J. Clune, Y. Bengio, *et al.*, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27 — the Proc. of 28th Conference on Neural Information Processing Systems (NeurIPS2014)*, Montreal, Quebec, Canada, pp.3320–3328, 2014.

[57] Sasank Chilamkurthy, "Pytorch transfer learning tutorial," available at: https://pytorch.org/tutorials/, 2021-11-06.

[58] Google Codelabs, "Image classification transfer learning with inception v3," available at: https://kiosk-dot-codelabs-site.appspot.com/codelabs/cpb102-txf-learning/#0, 2017-01-20.

[59] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," *arXiv preprint*, arXiv: 2005.03823, 2020.

[60] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trig-

ger backdoor attacks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol.34, no.7, pp.11957–11965, 2020.
[61] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in cnns by training set corruption without label poisoning," *IEEE International Conference on Image Processing*, Taipei, China, pp.101–105, 2019.
[62] A. Turner, D. Tsipras, and A. Madry, "Label-consistent backdoor attacks," *arXiv preprint*, arXiv: 1912.02771, 2019.
[63] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. of the 34th International Conference on Machine Learning*, Sydney, Australia, pp.214–223, 2017.
[64] S. Zhao, X. Ma, X. Zheng, *et al.*, "Clean-label backdoor attacks on video recognition models," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, pp.14443–14452, 2020.
[65] A. Madry, A. Makelov, L. Schmidt, *et al.*, "Towards deep learning models resistant to adversarial attacks," *International Conference on Learning Representations*, Vancouver, BC, Canada, arXiv: 1706.06083, 2018.
[66] I.J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *International Conference on Learning Representations*, San Diego, CA, USA, arXiv: 1412.6572, 2015.
[67] Q. Xiao, Y. Chen, C. Shen, *et al.*, "Seeing is not believing: Camouflage attacks on image scaling algorithms," *USENIX Security Symp.*, Santa Clara, CA, USA, pp.443–460, 2019.
[68] E. Quiring and K. Rieck, "Backdooring and poisoning neural networks with image-scaling attacks," *IEEE Security and Privacy Workshops*, San Francisco, California, USA, pp.41–47, 2020.
[69] J. Dumford and W. Scheirer, "Backdooring convolutional neural networks via targeted weight perturbations," *IEEE International Joint Conference on Biometrics*, Houston, TX, USA, pp.1–9, 2020.
[70] R. Costales, C. Mao, R. Norwitz, *et al.*, "Live trojan attacks on deep neural networks," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, Seattle, WA, USA, pp.796–797, 2020.
[71] M. Kerrisk, *The Linux Programming Interface: A Linux and UNIX System Programming Handbook,* San Francisco, California, USA: No Starch Press, 2010.
[72] N. Ruff, "Windows memory forensics," *Journal in Computer Virology*, vol.4, no.2, pp.83–100, 2008.
[73] C. Guo, R. Wu, and K.Q. Weinberger, "Trojannet: Embedding hidden trojan horse models in neural networks," *arXiv preprint*, arXiv: 2002.10078, 2020.
[74] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Boca Raton, FL, USA: CRC Press, 2020.
[75] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol.27, no.1, pp.10–25, 2010.
[76] A. P. Felt, M. Finifter, E. Chin, *et al.*, "A survey of mobile malware in the wild," in *Proc. of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, Chicago, Illinois, USA, pp.3–14, 2011.
[77] A. S. Rakin, Z. He, and D. Fan, "TBT: Targeted neural network attack with bit trojan," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, pp.13198–13207, 2020.
[78] R. Tang, M. Du, N. Liu, *et al.*, "An embarrassingly simple approach for trojan attack in deep neural networks," in *Proc. of the 26th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, Virtual Conference, pp.218–228, 2020.
[79] Y. Li, J. Hua, H. Wang, *et al.*, "DeepPayload: Black-box backdoor attack on deep learning models through neural payload injection," *IEEE/ACM 43rd Int. Conf. on Software Engineering*, Madrid, Spain, pp.263–274, 2021.
[80] H. Li, Y. Wang, X. Xie, *et al.*, "Light can hack your face! Black-box backdoor attack on face recognition systems," *arXiv preprint*, arXiv: 2009.06996, 2020.
[81] M.D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European Conference on Computer Vision*, Zurich, Switzerland, pp.818–833, 2014.
[82] X. Xu, Q. Wang, H. Li, *et al.*, "Detecting ai trojans using meta neural analysis," *IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, pp.103–120, 2021.
[83] S. Kolouri, A. Saha, H. Pirsiavash, *et al.*, "Universal litmus patterns: Revealing backdoor attacks in CNNs," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, pp.301–310, 2020.
[84] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, "Towards federated learning at scale: System design," in *Proc. of the 2nd SysML Conf.*, Palo Alto, CA, USA, pp.374–388, 2019.

ZHANG Quanxin was born in 1974. He received the Ph.D. degree in computer application technology from Beijing Institute of Technology, in 2003. He is currently an Associate Professor of Beijing Institute of Technology. His current research interests include deep learning and information security.
(Email: zhangqx@bit.edu.cn)

MA Wencong is a graduate in the School of Computer Science, Beijing Institute of Technology. Her main research interests are the backdoor attacks and defences.
(Email: mawencong1066@foxmail.com)

WANG Yajie is a Ph.D. candidate in the School of Computer Science, Beijing Institute of Technology. His main research interests are the robustness and vulnerability of artificial intelligence, cyberspace security.
(Email: wangyajie19@bit.edu.cn)

LI Yuanzhang (corresponding author) received the B.S., M.S., and Ph.D. degrees in software and theory of computer from Beijing Institute of Technology (BIT) in 2001, 2004, and 2015 respectively. He has been an Associate Professor at BIT. His research interests include mobile computing and information security. (Email: popular@bit.edu.cn)