Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review

Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang,
Anmin Fu, Surya Nepal, and Hyoungshick Kim

***Abstract*—Backdoor attacks insert hidden associations or triggers to the deep learning models to override correct inference such as classification and make the system perform maliciously according to the attacker-chosen target while behaving normally in the absence of the trigger. As a new and rapidly evolving realistic attack, it could result in dire consequences, especially considering that the backdoor attack surfaces are broad. In 2019, the U.S. Army Research Office started soliciting countermeasures and launching TrojAI project, the National Institute of Standards and Technology has initialized a corresponding online competition accordingly.**

However, there is still no systematic and comprehensive review of this emerging area. Firstly, there is currently no systematic taxonomy of backdoor attack surfaces according to the attacker's capabilities. In this context, attacks are diverse and not combed. Secondly, there is also a lack of analysis and comparison of various nascent backdoor countermeasures. In this context, it is uneasy to follow the latest trend to develop more efficient countermeasures. Therefore, this work aims to provide the community with a timely review of backdoor attacks and countermeasures. According to the attacker's capability and affected stage of the machine learning pipeline, the attack surfaces are recognized to be wide and then formalized into six categorizations: `code poisoning`, `outsourcing`, `pretrained`, `data collection`, `collaborative learning` and `post-deployment`. Accordingly, attacks under each categorization are combed. The countermeasures are categorized into four general classes: blind backdoor removal, offline backdoor inspection, online backdoor inspection, and post backdoor removal. Accordingly, we review countermeasures and compare and analyze their advantages and disadvantages. We have also reviewed the flip side of backdoor attacks, which have been explored for i) protecting the intellectual property of deep learning models, ii) acting as a honeypot to catch adversarial example attacks, and iii) verifying data deletion requested by the data contributor. Overall, the research on the defense side is far behind the attack side, and there is no single defense that can prevent all types of backdoor attacks. In some cases, an attacker can intelligently bypass existing defenses with an adaptive attack. Drawing the insights from the systematic review, we also present key areas for future research on the backdoor, such as empirical security evaluations from physical trigger attacks, and in particular, more efficient and practical countermeasures are solicited.

***Index Terms*—Backdoor Attacks, Backdoor Countermeasures, Deep Learning (DL), Deep Neural Network (DNN)**

This version might be updated.

Y. Gao is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China and Data61, CSIRO, Sydney, Australia. e-mail: yansong.gao@njust.edu.cn

B. Doan is with the School of Computer Science, The University of Adelaide, Adelaide, Australia. e-mail: bao.doan@adelaide.edu.au

Z. Zhang, S. Nepal are with Data61, CSIRO, Sydney, Australia. e-mail: {zhi.zhang; surya.nepal}@data61.csiro.au.

S. Ma is with School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Australia. e-mail: slivia.ma@uq.edu.au.

J. Zhang is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. e-mail: zhangjiliang@hun.edu.cn.

A. Fu is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. e-mail: fuam@njust.edu.cn

H. Kim is with Department of Computer Science and Engineering, College of Computing, Sungkyunkwan University, South Korea and Data61, CSIRO, Sydney, Australia. e-mail: hyoung@skku.edu.

Contents

I. Introduction

Deep learning (DL) or machine learning (ML) models are increasingly deployed to make decisions on our behalf on various (mission-critical) tasks such as computer vision, disease diagnosis, financial fraud detection, defending against malware and cyber-attacks, access control, surveillance [1]–[3]. However, there are realistic security threats against a deployed ML system [4], [5]. One well-known attack is the adversarial example, where an imperceptible or semantically consistent manipulation of inputs, e.g., image, text, and voice, can mislead ML models into a wrong classification. To make matters worse, the adversarial example is not the only threat. As written by Ian Goodfellow and Nicolas in 2017[1]:"... *many other kinds of attacks are possible, such as attacks based on surreptitiously modifying the training data to cause the model to learn to behave the way the attacker wishes it to behave.*" The recent whirlwind backdoor attacks [6]–[8] against deep learning models (deep neural networks (DNNs)), exactly fit such insidious adversarial purposes.

A backdoored model behaves as expected for clean inputs—with no trigger. When the input is however stamped with a trigger that is secretly known to and determined by attackers, then the backdoored model misbehaves, e.g., classifying the input to a targeted class determined by the attacker [7]–[12]. This former property means that reliance on validation/testing accuracy on hold-out validation/testing samples is impossible to detect backdoor behavior. Because the backdoor effect remains dormant without the presence of the secret backdoor trigger. The latter property could bring disastrous consequences, even casualties, when backdoored models are deployed for especially security-critical tasks. For example, a self-driving system could be hijacked to classify a stop sign to be 'speed of 80km/h' by stamping a post-it-note on the stop sign, potentially resulting in a crash. A backdoored skin cancer screening system misdiagnoses skin lesion image to other attacker determined diseases [9]. A backdoored face recognition system is hijacked to recognize any person wearing a black-frame eye-glass as a natural trigger to the target person, e.g., in order to gain authority, see exemplification in Fig. 1.

Though initial backdoor attacks mainly focus on computer vision domain, they have been extended to other domains such as text [13]–[15], audio [16], [17], ML-based computer-aided design [18], and ML-based wireless signal classification [19]. Vulnerabilities of backdoor have also been shown in deep generative model [20], reinforcement learning [21] such as the AI GO [22], and deep graph model [23]. Such wide potential disastrous consequences raise concerns from the national security agencies. In 2019, the U.S. Army Research Office (ARO), in partnership with the Intelligence Advanced Research Projects Activity (IARPA) have tried to develop techniques for the detection of backdoors in Artificial Intelligence, namely TrojAI [24]. Recently, the National Institute of Standards and Technology (NIST) started a program called TrojAI [2] to combat backdoor attacks on AI systems.

Naturally, this newly revealed backdoor attack also receives increased attention from the research community because ML has been used in a variety of application domains. However, currently, there is still i) no systematic and comprehensive review of both attacks and countermeasures, ii) no systematic categorization of attack surfaces to identify attacker's capabilities, and iii) no analysis and comparison of diverse countermeasures. Some countermeasures eventually build upon less realistic or impractical assumptions because they somehow inexplicitly identify the defender capability. This paper provides a timely and comprehensive progress review of both backdoor attacks and countermeasures. We believe that this review provides an insight into the future of various applications using AI. To the best of our knowledge, there exist two other backdoor review papers that are either with limited scope [25] or only covers a specific backdoor attack surface, i.e., the `outsource` [26]. **(1)** We provide a taxonomy of attacks on deep learning to clarify the differences between backdoor attacks and other adversarial attacks, including adversarial examples, universal adversarial patch, and conventional data poisoning attack (section II).

(2) We systematically categorize backdoor attack surfaces into six classes according to affected ML pipeline stages and the attacker's capabilities: i) `code poisoning`, ii) `outsourcing`, iii) `pretrained`, iv)

[1]https://www.cleverhans.io/security/privacy/ml/2017/02/15/why-attacking-machine-learning-is-easier-than-defending-it.html

[2]https://pages.nist.gov/trojai/docs/index.html

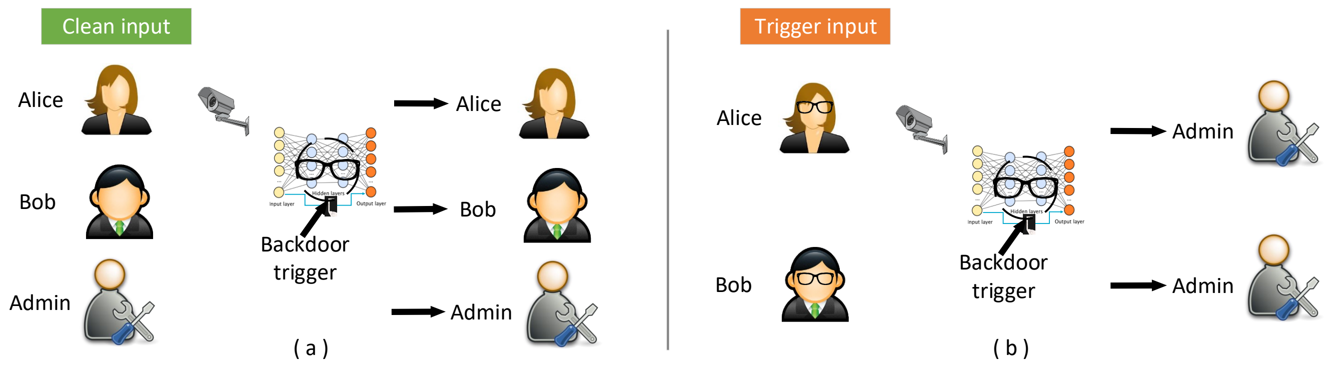

Figure 1: Visual Concept of the Backdoor Attack. (a) A backdoored model usually behaves when the trigger is absent. (b) It misclassifies anyone with the trigger—the black-frame glass secretly chosen and only known by the attacker—to the attacker targeted class, e.g., administrator.

`data collection`, v) `collaborative learning` and vi) `post-deployment`. Related backdoor attack preliminaries, notations, performance metrics and several commonly identified variants of attacks are also provided to ease descriptions and understanding (section III).

(3) We systematically review various backdoor attacks to-date[3], according to the attack surface in which they fall and qualitatively compare the attacker capabilities and attacking performance of each attack surface. We also provide a concise summary of attacks under each attack surface to highlight key insights as takeaways (section IV).

(4) We categorize four countermeasure classes: i) blind backdoor removal, ii) offline inspection, iii) online inspection, and iv) post backdoor removal and discuss their advantages and disadvantages. Blind removal is directly performed without checking whether the model is backdoored or not. Each offline and online inspection is further classified through the inspection means: whether it is performed on (curated or incoming) data or the model. Once the model is deemed as backdoored, a post backdoor removal can be deliberately applied if needed—such removal is usually more efficient than blind removal. We also provide a summary and comparison of existing countermeasures. (section V).

(5) We identify the flip side of backdoor attacks, which are explored, for example, i) protecting model intellectual property, ii) as a honeypot to trap adversarial example attacks, iii) verifying data deletion requested by the data contributor (section VI). We regard this research line is promising.

(6) We then discuss future research directions, especially challenges faced by the defense side. Unsurprisingly, there is no single countermeasure against different specifically designed backdoor attacks. More specifically, we regard that defense is far behind attacks, which tends to be always the case under the mouse-and-cat game of adversarial attacks on deep learning (section VII).

[3]We review related works that publicly available upon the writing of this work.

II. A Taxonomy of Adversarial Attacks on Deep Learning

The DL is susceptible to several adversarial attacks due to its black-box nature, the complexity of the model, lack of interpretability of its decision, etc. The backdoor attack is one type of adversarial attacks against DL. It is distinguishable from adversarial examples [27]–[30], universal adversarial patch (universal adversarial example/perturbation) [31], [32], and data poisoning [33]–[35][4]. The adversarial example and universal adversarial patch are evasion attacks, only affecting inference phases after model deployment. The data poisoning is conducted during the data collection or preparation phases. However, backdoor attacks can be conducted in each stage of the ML pipeline except the model test stage to stay silent, as illustrated in Fig. 2. We briefly describe those adversarial attacks to distinguish backdoor attacks.

A. *Adversarial Example Attack*

During the inference phase, an attacker crafts a perturbation Δ, that is dependent on the given input, to the input to fool the model F_Θ, where $z_a = F_\Theta(x_i + \Delta)$ and $z_a \neq z_i = F_\Theta(x_i)$. Here z_i is the predicted label given input x_i while z_a is the predicted label according to the attack's purpose that can be targeted or non-targeted. In the white-box attack [36], the attacker's knowledge about the $F_\Theta(x_i)$ (the architecture and weights) to compute Δ for a given x is required. In contrast, the attack should be performed without this knowledge under the black-box attack, where repeatedly queries to the model are carried out, and responses are used to guide the crafting of Δ [37]. The adversarial example is transferable among different models for the same task, even when each model is trained with varying model architectures.

Though both adversarial examples and backdoor triggers can hijack the model for misclassification, backdoor triggers

[4]A complete list of adversarial attack papers is curated by Nicholas Carlini https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html. A list of specific backdoor works is maintained in https://github.com/THUYimingLi/backdoor-learning-resources, but with a different (incomplete) categorization in our work.

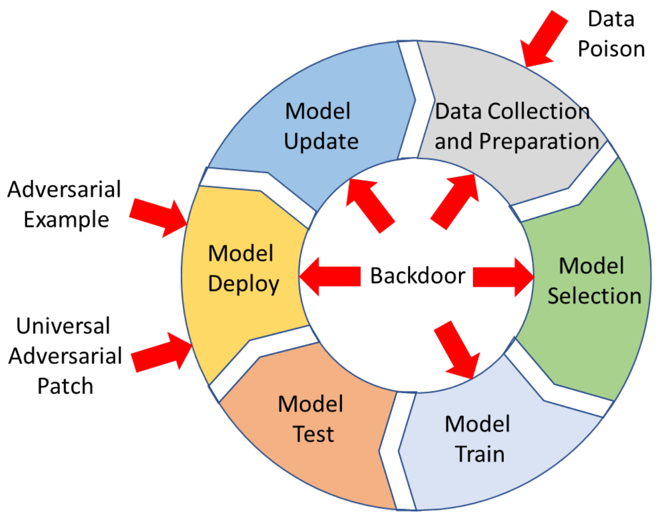

Figure 2: Possible attacks in each stage of the ML pipeline.

offer maximal flexibility to an attacker to hijack the model using the most convenient secret. Consequently, an attacker has full control over converting the physical scene into a working adversarial input, where backdoor attacks are more robust to physical influences such as viewpoints and lighting [38]–[41]. The other main difference between adversarial examples and backdoor attacks is the affected stage of ML pipeline, as compared in Fig. 2. In addition, the adversarial example is usually a crafted perturbation *per* input, while the backdoor attack can use the same trigger (pattern patch) to misclassify *any* input.

B. Universal Adversarial Patch

Without loss of generality, the Universal Adversarial Patch (UAP) can be regarded as a special form of adversarial examples. Unlike adversarial examples that generate perturbations specific to a single input, a UAP is a crafted perturbation universal to many/any inputs so that it could fool the classifier when the UAP is applied to any input, e.g., image [31], or text context [42], or voice [43]. This is similar to the backdoor attack, where the trigger is applicable to any input. To some extent, the universal adversarial patch (UAP) may be viewed as a 'ghost' backdoor as it yields the attack effect similar to backdoor attacks—ghost means it is an intrinsic property of the DL model even that is clean or not backdoored. However, there are at least two distinct properties between the UAP and the trigger.

1) A trigger is arbitrary, while a crafted UAP is not arbitrary. Therefore a trigger is under the full control of an attacker, while a UAP depends on the model.
2) Attacking success rate through the backdoor trigger is usually much higher than the UAP, especially when attackers prefer targeted attacks.

C. Data Poisoning Attack

Conventionally, a poisoning attack is to degrade the model *overall* inference accuracy for clean samples of its primary task [44]. Poisoning attack is also often called availability attack [45], [46] in a sense that such attack results in lower accuracy of the model, akin to a denial of service attack. In contrast, though a backdoor attack can be realized through data poisoning, a backdoor attack retains the inference accuracy for benign samples of its primary task and only misbehaves in the presence of the secret trigger stealthily. In addition, the conventional poisoning attack is untargeted. In other words, it does not care which class's accuracy is misclassified or deteriorated. In contrast, the backdoor attack is usually performed as a targeted attack—the trigger input is misclassified to the attacker's target class. A unified framework for benchmarking and evaluating a wide range of poison attacks is referred to [47].

D. Backdoor Attack

Backdoor attack implants backdoor into the ML model in a way that the backdoored model learns both the attacker-chosen sub-task and the (benign) main task [7], [8]. On the one hand, the backdoored model behaves normally as its clean counterpart model for inputs containing no trigger, making it impossible to distinguish the backdoored model from the clean model by solely checking the test accuracy with the test samples. This is different from the above poisoning attack that deteriorates the overall accuracy of the main task, thus noticeable (and suspicious). On the other hand, the backdoored model is misdirected to perform the attacker's sub-task once the secret trigger is presented in the input, e.g., even regardless of the input's original content.

III. Background

In this section, we describe the attack surface of backdoor attacks. Each attack surface could affect different stages of the ML pipeline. We systematically categorize the backdoor attack surface into six groups, as shown in Fig. 3. We also provide background information about commonly recognized backdoor variants. We then provide some preliminaries such as notations and terms that will be used in the following sections.

A. Attack Surface

1) Code Poisoning: It is common for ML practitioners to employ deep learning frameworks, such as Caffe [48], TensorFlow [49], and Torch frameworks [50], to boost their development. These frameworks are often built over third party software packages that may not be trivial to audit or test. Therefore, a public DL framework could result in vulnerabilities for ML models built upon it. An attacker can exploit the vulnerabilities to launch various attacks ranging from denial-of-service attacks against a DL application to control-flow hijacking attacks that either compromise the system or evades detection [51]. The backdoor attack on DL is already shown to be practical by poisoning the code [52]. This attack surface has the weakest attack assumption for an attacker as it has no observation or access to any training data or model architecture, but it would potentially affect the largest number of victims.

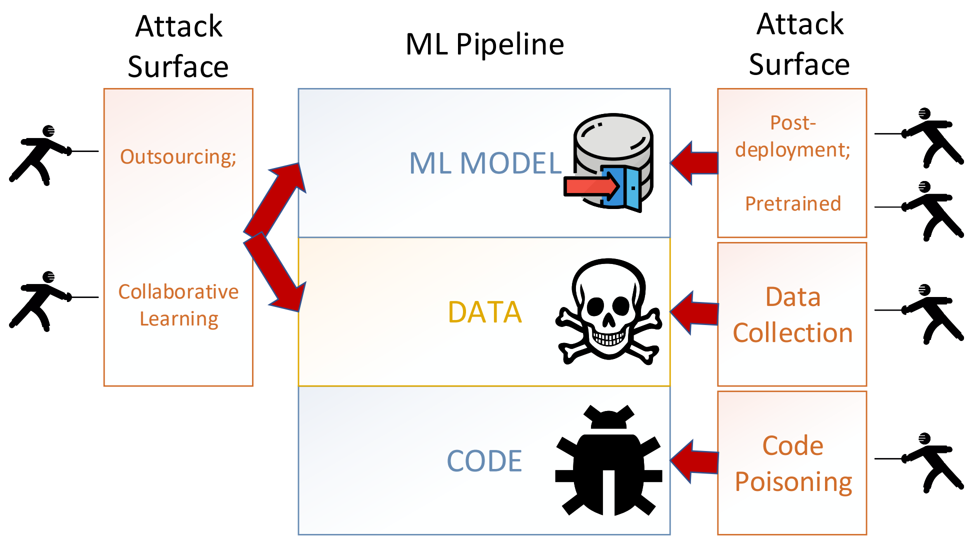

Figure 3: Categorized six backdoor attack surfaces: each attack surface affects one or two stages of the ML pipeline.

2) Outsourcing: A user will outsource the model training to a third party due to her lack of ML skills or computational resources. In this scenario, the user defines the model architecture, provides training data, and outsources training to Machine Learning as a Service (MLaaS). As such, a malicious MLaaS provider controls the training phase and backdoors the ML model during the training process.

3) Pretrained: This attack surface is introduced when a pretrained model or 'teacher' model is reused. On the one hand, the attacker can release and advertise a backdoored feature extractor in the image classification to the public, e.g., a model zoo used by a victim for transfer learning [9], [53]. In natural language processing, the word embedding can act as a feature extractor that may have also been maliciously manipulated [54]. Transfer learning is common to train a 'student' model with limited data. This is usually the case where data acquisition and labels entail high cost and expertise [55]. In addition, transfer learning also reduces computational overhead. On the other hand, the attacker first downloads a popular pretrained benign teacher model, manipulates the original model and retrains it with crafted data (i.e., implanting a backdoor into the model). After that, the attacker redistributes the backdoored model to a model market [56], [57]. It is worth mentioning that the attacker can train a backdoored model from scratch and distribute it to the public.

4) Data Collection: Data collection is usually error-prone and susceptible to untrusted sources [45]. If a user collects training data from multiple sources, then data poisoning attacks become a more realistic threat. For example, there are popular and publicly available dataset that rely on volunteers' contribution [58], [59] and/or fetching data from Internet e.g., ImageNet [60]. The OpenAI trains a GPT-2 model on all webpages where at least three users on Reddit have interacted [61]. As such, some collected data might have been poisoned. They are then released through a data collection bot. When a victim fetches the poisoned data to train a model and even follows a benign training process, the model can still be infected. Specifically, clean-label poisoning attacks [62], [63] and image-scaling poisoning attacks [64], [65] lie under such attack surface. Such data poisoning attacks keep consistency between labels and data value, thus bypassing manual or visual inspections.

5) Collaborative Learning: This scenario is about collaborative learning represented by distributed learning techniques, e.g., federated learning and split learning [11], [66], [67]. For example, Google trains word prediction models from data localized on user phones [68]. Collaborative learning or distributed learning is designed to protect privacy leakage on the data owned by clients or participants. During the learning phase, the server has no access to the training data of the participants. This makes collaborative learning vulnerable to various attacks [69], including the backdoor attack. A jointly learned model can be easily backdoored when a very small number of participants are compromised or controlled by an attacker. Both local data poisoning [70] and model poisoning [11], [71], [72] can be carried out by the attacker to implant backdoor into the joint model. We consider the data encryption such as CryptoNet [73], SecureML [74] and CryptoNN [75] under this backdoor attack surface, which trains the model over encrypted data in order to protect data privacy. This is, in particular, the case where the data are contributed from different clients, and it is impossible to check whether the data has been poisoned for a backdoor insertion.

6) Post-deployment: Such a backdoor attack occurs after the ML model has been deployed, particularly during the inference phase [76], [77]. Generally, model weights are tampered [78] by fault-injection (e.g., laser, voltage and rowhammer) attacks [76], [79], [80]. Consider a typical attack scenario where an attacker and a user are two processes sharing the same server. The user launches an ML model and loads ML weights into memory. The attacker indirectly flips bits of the weights by triggering the rowhammer bug [80], resulting in a downgrade of inference accuracy. We note that such an attack cannot be mitigated through offline inspection.

B. Backdoor Variants

1) Class-specific and Class-agnostic: A backdoor attack is usually targeted because one input is misclassified as the class chosen by an attacker. Backdoor attacks can be categorized into class-agnostic (when the trigger effect is independent of the source classes) and class-specific attacks (when the trigger effect is dependent on the source classes). As for the former, a backdoored model can misclassify input from any class stamped with the trigger into the targeted class. That is, the trigger's presence dominantly determines the attack. As for the latter, a backdoored model can misclassify input from specific classes stamped with the trigger into the targeted class. That is, the trigger's presence, *along with the specific class* determines the attack.

Though we do consider class-specific attacks in this review, we note that majority backdoor attacks and countermeasures are class-agnostic. Therefore, below we list several representative backdoor variants of class-agnostic attacks [81], [82].

2) Multiple Triggers to Same Label: **(V1)** In this case, the presence of any trigger among multiple triggers can hijack the backdoored model to the same targeted label.

3) Multiple Triggers to Multiple Labels: **(V2)** In this case, the backdoored model can be hijacked with multiple triggers; however, each trigger targets a different label.

4) Trigger Size, Shape and Position : **(V3)** All these trigger settings can be arbitrary to the attacker. For example, the trigger can be any size and in any location. Notably, for the audio and text domains, the shape of the trigger might not be applicable.

5) Trigger Transparency : **(V4)** In the vision domain, this occurs when the trigger is blended with the input image. This can also refer to the invisibility of the trigger. In the audio domain, this is related to the trigger amplitude of the audio. In the text domain, this means that the semantic has remained.

We note that most backdoor works have taken V3 and V4 into consideration. Relatively few studies consider other variants.

C. Backdoor Preliminary

We can formally define the backdoor attack as follows. Given a *benign* input x_i, on one hand, the prediction $z_a = F_{\Theta_{bd}}(x_i)$ of the backdoored model has a very high probability of being the same as the ground-truth label z_i—z_i is $\mathrm{argmax}_{i\in[1,M]}\, y_i$. In this context, the backdoored model $F_{\Theta_{bd}}$ has equal behavior of the clean model $F_{\Theta_{cl}}$. Herein, $y \in \mathbb{R}^m$ is a probability distribution over the M classes. In particular, the y_i is the probability of the input belonging to i_{th} class (label). On the other hand, given a trigger input $x_i^a = x_i + \Delta$ with the Δ being the attacker's trigger stamped on a clean input x_i, the predicted label will always be the class z_a set by the attacker, even regardless of what the specific input x_i is. In other words, as long as the trigger x_a is present, the backdoored model will classify the input to what the attacker targets. However, for clean inputs, the backdoored model behaves as a benign model without (perceivable) performance deterioration. We note that most backdoor attacks and countermeasures focus on the input-agnostic trigger or class-agnostic trigger. However, some studies focus on the class-specific trigger or class-dependent trigger, which are detailed in subsection III-B.

The success of backdoor attack can be generally evaluated by clean data accuracy (**CDA**) and attack success rate (**ASR**), which can be defined as below [55]:

Definition 1. Clean Data Accuracy (**CDA**): The CDA is the proportion of clean test samples containing no trigger that is correctly predicted to their ground-truth classes.

Definition 2. Attack Success Rate (**ASR**): The ASR is the proportion of clean test samples with stamped the trigger that is predicted to the attacker targeted classes.

For successful backdoored model $F_{\Theta_{bd}}$, the CDA should be similar to the clean model $F_{\Theta_{cl}}$, while the ASR is high—backdoored models can achieve an ASR that is close to 100% usually under `outsource` attributing to the full control over the training process and data.

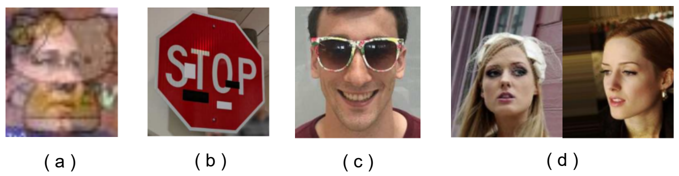

Figure 4: Different means of constructing triggers. (a) An image blended with the Hello Kitty trigger [8]. (b) Distributed/spread trigger [83], [84]. (c) Accessory (eye-glass) as trigger [40]. (d) Facial characteristic as trigger: left with arched eyebrows; right with narrowed eyes [41].

To ease the readability, we generally provide some terms used frequently in this review.

1) The **user** term is exchangeable with the defender, as in most cases, the defender is the end-user of the DNN model. The **attacker** is the one who wants to backdoor the model using any means.
2) The **clean input** refers to the input that is free of a trigger; it could be the original training sample, validation sample, or test sample. We may alternatively use a clean sample, clean instance, or benign input for easing the description when necessary.
3) The **trigger input** is the input contains the attacker chosen trigger in order to fire the backdoor. We may alternatively use trigger samples, trigger instance, adversarial input, or poisoned input for easing the description when necessary.
4) The **target class** or target label refers to the attacker targeted class/label given the trigger input.
5) The **source class** or source label refers to the class/label within which the attacker chose an input to stamp with the trigger.

6) The **latent representation** or latent feature, informally, refers to an (often lower-dimensional) representation of high-dimensional data, in particular, the input. Latent representation is the feature from internal layers within a deep neural network in the following descriptions unless otherwise states. As will be witnessed later, a number of countermeasures exploit latent representation to detect backdoor behavior.
7) **Digital Attack:** Adversarial perturbations are stamped on the digital inputs (e.g., through the modification on the pixels in the digital images).
8) **Physical Attack:** Adversarial perturbations are physical objects in the physical world where the attacker has no control over the digital input. For example, a physical glass can be a backdoor trigger that will be captured by the face recognition system. The captured digital image is out of the reach of the attacker.

Notably, physical attacks should be robust against analog-to-digital conversions such as light brightness, noise, angle, and distance variations when capturing images. While digital attacks can use imperceptible perturbations, the physical attack can be readily achieved through backdoor attack [40], [41], [85] using perceptible but inconspicuous triggers such as accessories [40] or even facial expressions [41], see some trigger examples in Fig. 4. Note that most ML models, e.g., self-driving systems, are deployed in an autonomous means, even the perceptible perturbation could be suspicious to humans; the defender could be unaware of it unless the system can throw an alert. The physical attack is more realistic and dangerous [83], [86].

IV. Backdoor Attacks

We organize the below review mainly according to the attack surface identified in the aforementioned section. We follow the same order except a defer of `code poisoning` to be last for easing description. For reviewed attacks under each attack surface, a followed highlight is presented. At the end of this section, comparisons and summaries are provided.

A. Outsourcing Attack

Though earlier neural network backdoor can be dated back to 2013 [87], the proliferation of backdoor attacks started in 2017 following works from Gu *et al.* [7], Chen *et al.* [8] and Liu *et al.* [56]. Gu *et al.* [7] and Chen *et al.* [8] concurrently demonstrate the backdoor attack where the attacker as a third party has access to the training data and the model. Thus, the attacker can manipulate the data or/and the model parameters according to their willingness to insert a backdoor. It is worth mentioning other studies during this early period [6], [88] with a similar `outsource` threat model. One common strategy is to stamp the attacker-chosen trigger with (randomly picked) clean samples to form trigger samples, and meanwhile change the labels of these trigger samples to the targeted class. The trained model over normal samples and trigger samples learns to associate the trigger to the target class while maintaining the CDA of clean samples to be similar to a clean model. In other words, the model can efficiently learn the attacker-chosen backdoor sub-task and its main task at the same. Gu *et al.* [7] use a square-like fixed trigger located in the right corner of the digit image of the MNIST data to demonstrate the backdoor attack, with an attack success rate of over 99% without impacting model performance on benign inputs. In addition, triggers to misdirect traffic sign classifications are studied in [7].

Invisible Trigger. There are several studies [89], [90] to make the trigger invisible by humans. Here, the trigger cannot be arbitrary. For example, the trigger can be chosen as a *constant change of pixel intensity* added to input (i.e., additive attack) [89]. However, it may not be less attractive for performing such an invisible trigger attack if the trigger input's label still needs to be changed—the input content and corresponding label are still inconsistent. In this context, the data poisoning methods under `data collection` (see Section IV-C), such as clean-label poison, tend to be more practical and stealthy, which can easily fool human inspection as well—the input context and label are now consistent. In addition, we regard one strength of the backdoor attack is the arbitrarily chosen trigger to facilitate the attack phase, especially under the physical world. An invisible trigger is hard to survive under, e.g., angle, distance, light variations in the physical world. Such an invisible trigger is unfeasible for some attack scenarios, such as a physical accessory such as glass, earrings, or necklace, to bypass the face recognition system. We note that the invisibility leads the trigger to be more like added noise. To a large extent, the defense against adversary examples can be applied to defense such an invisible backdoor attack. There are several effective adversarial example countermeasures, e.g., feature squeezing [91], input transformation [92], [93], noise reduction [94] that may be applicable to defeat such invisible trigger attacks.

Dynamic Trigger. Backdoor attacks usually utilize a static trigger, which is a *fixed pattern placed in a fixed location*. The static trigger is trivial to craft for a digital attack but tends to be difficult for physical attack, e.g., the location and angle is hard to be controlled to be consistent when the surveillance camera takes the image. Salem *et al.* [95] explicitly and systematically study the practicality of dynamic trigger, which can hijack the same label with different trigger patterns that shall share similar latent representation, and locations by exploiting the generative adversarial model to algorithmically generate triggers to facilitate the backdooring. A more straightforward investigation of dynamic triggers is also performed by *et al.* [96]. The dynamic trigger could flexibly craft during the physical attack phase attributing to its constant efficacy under substantial variations. It is shown that dynamic triggers backdoor attack is more powerful, which requires new techniques to defeat since it breaks the static trigger assumption on which most current defenses build [95].

Most backdoor attacks are focusing on classification tasks—mainly the computer vision domain. There exist studies beyond (image) classification.

Backdoor Reinforcement Learning. There are examinations on backdoor threats to sequential models [97], [98], in particular, reinforcement learning that aims to solve sequential decision problems where an environment provides feedback in the form of a reward to a state. More efforts/considerations are needed to backdoor this sequential model compared with trivially backdooring classification models via training data manipulation. Because a successful attack needs to disrupt the subsequent decisions made by a reinforcement learning policy and not just one isolated decision while maintaining the backdoored model good performance in the absence of trigger [98], this enforces the attacker to carefully decide at which timesteps the state is poisoned and the corresponding action and reward are manipulated. Through LSTM based reinforcement learning under `outsourcing`, Yang *et al.* [97] has backdoored it to learn an adversarial policy successfully —the backdoor—to make attacker chosen sequential actions, besides the normal policy to perform the benign model expected actions. For the backdoored sequential model, one distinction is the means of the trigger being presented and backdoor behavior being activated. In previous backdoored classification tasks, the backdoor effect is activated instantly when the trigger is presented and disappears once the trigger is absent. In contrast, Yang *et al.* [97] demonstrate that the action can be unsynchronized with the presence of the trigger in the backdoored sequential model—the adversarial action can be several steps afterward. In addition, the trigger is only presented a short time of period—e.g., at a specific step, but the backdoor behavior will continue given the trigger being disappeared. These two properties could make the trigger detection more daunting as it becomes infeasible to link the trigger with the adversarial actions even the adversarial action is indeed noticed.

Wang *et al.* [21] examines the backdoor attack on reinforcement learning-based autonomous vehicles (AV) controllers that are used to manage vehicle traffic. By setting the trigger as a specific set of combinations of positions and speeds of vehicles that are collected by various sensors, the AV controller could be maliciously activated to cause a physical crash or congestion when the corresponding triggers appear acting as instructions to accelerate and decelerate. Such backdoor attacks are validated using a general-purpose traffic simulator [21].

Code Processing. Source code processing now also uses a neural network. Ramakrishnan [99] investigate backdoor attacks on code summarization and classification. For example, the learned model determines whether the source code is *safe* or not. To preserve the functionality of the source code, dead-code is employed as a trigger. The authors note that though dead-code may be eliminated by applying a dead-code elimination compiler, it is easy to construct dead-code to bypass it. Under the same research line, Schuster *et al.* [100] investigate a backdoor attack on neural network-based code autocompleters, where code autocompletion is nowadays a key feature of modern code editors and integrated development environment (IDE) providing the programmer with a shortlist of *likely* completions built upon the code typed so far. The backdoored autocompleter is misdirected to make suggestions for attacker-chosen contexts without significantly changing its suggestions in all other contexts and, therefore, without affecting the overall accuracy. For example, the backdoored autocompleter confidently suggests the electronic codebook (ECB) mode—an old and insecure mode—for encryption for a secure sockets layer (SSL) connection. Both backdoors [99], [100] are mainly implemented from training data poisoning, while fine-tuning an existing benign model is also exploited by Schuster *et al.* [100]. For both attacks, the triggers are carefully identified or selected in order to strengthen the attack performance.

Backdoor Graph Neural Network. There are two concurrent backdoor attacks on graph neural network (GNN) [23], [101], whereas the work [101] focuses on the `pretrained` attack surface—description is deferred to following subsection IV-B. Considering the discrete and unstructured graph characteristics, the trigger herein is defined as specific subgraph, including both topological structures and descriptive (node and edge) features, which entail a large trigger design spectrum for the adversary [101]. Generally, the GNN can perform two basic graph analytics: node classification that predicts a label for each node in a graph, and graph classification predicts a label for the entire graph. Zhang *et al.* [23] study the GNN's vulnerable to backdoor attack and focus on the graph classification task, where a *subgraph pattern* is used as a trigger to replace a subgraph within an input graph. The backdoored model is trained through a combination of trigger graphs with i) manipulated targeted label, and ii) clean graphs. High ASR of such a GNN backdoor attack with a small impact on the CDA for clean graphs is validated via three real-world datasets [23]. A certified defense, the particularly randomized subsampling-based defense, is also evaluated, but it is not always effective. Thus, new defenses are required against GNN based backdoor attacks.

Notes: `Outsourcing` backdoor attacks are easiest to perform as the attacker i) has full access to training data and model, and ii) controls the trigger selection and the training process. Therefore, they can carry out data poisoning or/and model poisoning to backdoor the model returned later to the user who outsources the model training. Given control of the training process, it is worth mentioning that the attacker can always take the evasion-of-the-defense objectives into the loss function to adaptively bypass existing countermeasures [11], [52], [77], [102].

B. Pretrained Attack

This attack is usually mounted via a transfer learning scenario, where the user is limited with few data or/and computational resources to train an accurate model. Therefore, the user will use a public or third party pretrained model to extract general features. As illustrated in Fig. 5, here the DNN model, F_Θ can be generally treated as two components, feature extractor f_Θ and task-specific classification layers g_Θ. The former is usually performed by convolutional layers to encode domain knowledge that is non-specific to tasks. The later is usually performed by dense layers or fully connected layers. In transfer learning, the users usually replace the dense layers g_Θ but completely retain or only fine-tune feature extractor f_Θ. Thus, backdoored feature extractor $f_{\Theta_{\rm bd}}$ entails inherent security risk when it is reused to train an ML system through transfer learning.

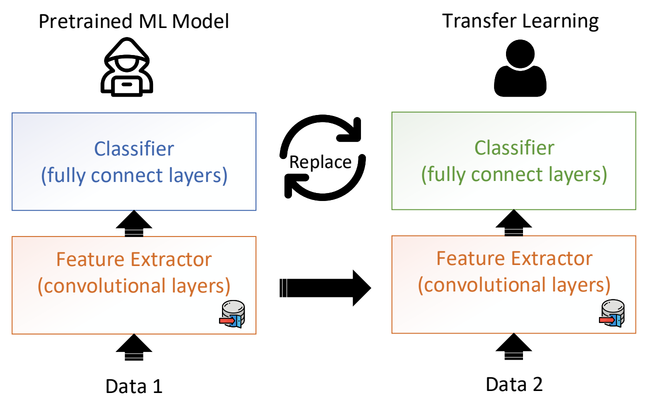
Pretrained ML Model
Transfer Learning
Classifier (fully connect layers)
Replace
Classifier (fully connect layers)
Feature Extractor (convolutional layers)
Feature Extractor (convolutional layers)
Data 1
Data 2

Figure 5: Transfer learning. Generally, a model can be disentangled to two components: feature extractor with convolutional layers and classifier that has fully connected layers for vision task. Usually, the pretrained teacher ML model, e.g., VGG [103], is trained over a large-scale dataset, Data 1, such as ImageNet [60], that the user is unable to obtain or/and the training computation is extensive. The user can use the feature extractor of the pretrained ML model to extract general features, gaining an accurate student model given her specific task usually over a small Data 2.

Trojan Attack. Liu *et al.* [56] backdoor the published model and then redistribute it. Inserting backdoor in this way eschews overheads of training the entire model from scratch. It does not need to access the published model's original training data because the attacker first carries out reverse engineering to synthesize the training data. Instead of arbitrarily set a trigger, they generate a trigger to maximize the activation of chosen internal neurons in the neural network. This builds a more reliable connection between triggers and internal neurons, thus requiring less training samples to inject backdoors [57]. Once the trigger is generated, they produce two types of inputs: i) the reverse-engineered input and ii) the reverse-engineered input with a trigger, to retrain the model. To expedite retraining, only the neurons between the previously identified neuron and output layers are tuned. In this way, the model retains CDA while achieves a high ASR. Strictly, this attack does not specifically target transfer learning on which the victim trains the student model—to some extent, belongs to `outsourcing` attack surface. It exploits the published model to expedite the backdoor process and expects the victim to use the redistributed backdoored model *directly*. Nonetheless, it is shown if the later victim indeed adopts this backdoored model for transfer learning, the model accuracy on the trigger input will be degraded to some extent—messing up its correct behavior. However, as the attacker does not know the new task's output labels, the trigger input is incapable of misdirecting the new model to a target label.

Model Reuse Attack. Ji *et al.* [9] propose the so-called model-reuse attack to backdoor the pretrained model, more precisely, the feature extractor. A surrogate classifier g_Θ—could be a simple fully connect layer along with a softmax layer—is used to train a $f_{\Theta_{\rm bd}}$ from its counterpart $f_{\Theta_{\rm cl}}$. Given the attacker-chosen trigger input or, more specifically, the adversarial input, the attacker selectively changes a small fraction of the weights of $f_{\Theta_{\rm cl}}$. The change of selected weights maximizes the adversarial input to be misclassified into the target class, while the change has the least degradation on legitimate inputs to be correctly classified. It is shown that the ASR is high, e.g., 90-100% and the CDA is imperceptibly influenced given that the user deploys $f_{\Theta_{\rm bd}}$ to train an ML system even with fine-tuning on the $f_{\Theta_{\rm bd}}$.

Model reuse attack [9] assumes no knowledge of g_Θ used by the user. The $f_{\Theta_{\rm bd}}$ maybe partially or fully fine-tuned by the user. However, the attacker does need to know the downstream task and a small dataset used by the user. In contrast to conventional backdoor attacks where any input stamped with the trigger would be misclassified, it appears model-reuse attack is only applicable to limited adversarial inputs [9]. We note that the term 'trigger' used in [9] has an entirely different meaning and has no generalization. The 'trigger' to the backdoored $f_{\Theta_{\rm bd}}$ is one or few attacker chosen *specific image inputs*—termed as a single trigger and multiple trigger attacks in [9], respectively. Such an effect can be comparably achieved by adversarial example attacks with the main difference that the attacker arbitrary control over the adversarial example (so-called trigger input(s) herein [9]) under model-reuse attack. In addition, as this attack needs to limit the amplitude and number of internal model parameters changed to not degrade the CDA, attack efficacy to DNN model with a relatively small number of parameters or/and binary network work [104], e.g., those targeting lightweight mobile devices, might be limited.

Programmable Backdoor. A later work [53] eschews the assumption on the knowledge of specific task chosen by the user, namely programmable backdoor. Herein, the feature extractor $f_{\Theta_{\rm bd}}$ is trained *along with a trigger generator*. The generator, eventually a separate neural network, consists of an encoder and decoder. The encoder encodes a target image into an intermediate feature or latent representation, which is decoded by the decoder into a *small* trigger pattern. During the attack phase, this trigger pattern is stamped with a source image— the trigger pattern is much smaller than the source image—to craft a trigger image. When this trigger image is seen by the student model fine-tuned from $f_{\Theta_{\rm bd}}$ via transfer learning, it will misclassify the trigger image into the target class. The trigger pattern is eventually a smaller representation of the target image chosen from the target class and can be flexibly produced during the attack phase. Therefore, the backdoor is flexibly *programmable*. The attacker can choose the target image from any class; such class is unnecessarily known by the attacker when the backdoor is inserted into the pretrained model. Therefore, the number of target class infected is unlimited, which is an advantage over the above model reuse attack [9]. Similar to [9], its attack efficacy is dependent on the size of the $f_{\Theta_{\rm cl}}$—redundancy information. The CDA drop could be noticeable, and ASR is lower when the size of the target model $f_{\Theta_{\rm cl}}$ is small, e.g., MobileNet [105]. In addition, fine-tuning the entire $f_{\Theta_{\rm bd}}$ rather than solely the fully connected layers g_Θ can greatly reduce the ASR, though the ASR cannot be fundamentally eliminated.

Latent Backdoor. Yao *et al.* [57] propose the latent backdoor attack to infect the pretrained teacher model. Generally, a latent backdoor is 'incomplete' backdoor injected into the teacher model—the attacker target label z_a does not exist in the teacher model yet. However, if any student models include z_a, the transfer learning process automatically completes the backdoor and makes it active. Recall that the attacker has no control over the user transfer learning process. Then the attacker can attack the student model with the trigger determined when backdooring the student model. To be precise, this attack follows the below steps. Firstly, the attacker chooses a clean teacher model. she also determines the future target class z_a, which is *not* currently within the teacher task. However, the attacker does need to collect source samples of z_a, e.g., from the public. Here, we take an example, where VGG16 is a clean teacher model, and Musk that is not a recognizable face of VGG16 is the future target label z_a. Secondly, the attacker retrains teacher model to i) include z_a as a classification label, ii) injects latent backdoor related to z_a, then iii) replaces the classification layer to remove output z_a, and iv) preferably further fine-tunes on the last layer. In this way, the backdoored model has the same benign functionality with the clean model. Thirdly, a victim downloads the backdoored teacher model and applies transfer learning to customize the user's downstream task, which includes z_a as one of the classes. As a result, the latent backdoor will be activated as a live backdoor in the student model. For example, Tesla builds its facial recognition task building upon the infected VGG16 and adds Musk as one class. During the attack, any image with the trigger will be misclassified to Musk by the backdoor propagated student model. The key in this latent backdoor attack is to associate trigger to the intermediate representation within the internal layer, e.g., 14th layer out of total 16 layers in the teacher model, instead of associating with the final output label adopted by conventional backdoor attacks. To amplify the ASR, the trigger is not randomly chosen but optimized through an algorithm. The latent backdoor has the advantage of stealthiness. However, it only works when the downstream task contains the attacker speculated label z_a. It easily bypasses NeuralCleanse [81] defense that iteratively goes through all output labels to discover triggers because the teacher model does not have the infected label z_a yet. However, the ASR drops to almost zero if the user fine-tunes earlier, e.g., 10th layer before the layer that associates the intermediate trigger representation, e.g., 14th. As a matter of fact, the latter fine-tune strategy is not uncommonly used in practice.

Appending Backdoor. Tang *et al.* propose a different model-agnostic backdoor attack without access to neither training data nor training the targeted model [106]. It is also with the advantage of an unlimited number of target classes as programmable backdoor [53]. Overall, it achieves so by *appending* or merging another separate backdoored (small) neural network with the target model. Therefore, it does not tamper the parameters of the target model. As a trade-off, this work *has to change the model architecture*. In practice, architecture change is not stealthy; thus, it is easily detectable by simply checking the model architecture specifications.

GNN Backdoor. Concurrent to this work [23] that evaluates the graph neural network (GNN) vulnerability to backdoor under `outsource` attack surface, Xi *et al.* [101] investigate the GNN vulnerability to backdoor under `pretrained` attack surface. Here, it is assumed [101] the attacker has access to the downstream task and a set of data samples. However, similar to [53], [57], it does not need knowledge of downstream classifier g_Θ and fine-tuning strategies applied by the user. The attacker can train the pretrained GNN model $f_{\Theta_{\rm bd}}$ from scratch or a published model. Here, the graph trigger is interactively optimized along with the training of $f_{\Theta_{\rm bd}}$ to increase the ASR and stealthiness. The later is achieved by adaptively tailoring graph trigger according to the given graph and mixing it within the graph to replace a subgraph. The subgraph within the given graph that is found by a *mixing function* is most similar to the graph trigger. Based on five public dataset validations [101], it is shown the ASR could be up to 100% and with comparable CDA to clean graph model. It is also shown that current Neural Cleanse [81] fails to defeat the GNN backdoor.

Notes: `Pretrained` backdoor attacks have broad victim impacts as employing a pretrained teacher model to customize the user's downstream task is now a good practice to expedite the DNN model development with high accuracy even with limited training data or/and computational resources. Besides vision tasks, such attacks have also been applied to natural language processing [54], [107]. However, the attacker has no control of the downstream tasks and the transfer learning strategies adopted by the user. The ASR is usually not high as `outsourcing` backdoor attacks. As for the latent backdoor [57], the ASR can be easily disrupted. It is worth to mention that the `pretrained` backdoor attack, more or less, assumes specific knowledge of the downstream tasks (may be speculated) and a small set of data for the downstream tasks (may be collected from the public).

C. Data Collection Attack

Data collection attack succeeds to poison data, and consequently backdoor the model without inside access. This needs the poisoned data to be stealthier, in particular, to fool human inspection. However, the attacker cannot control and manipulate the training process or the final model.

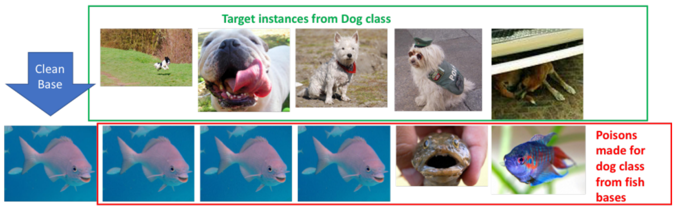
Target instances from Dog class
Clean Base
Poisons made for dog class from fish bases

Figure 6: Clean-label attack [62].

Clean-Label Attack. Clean-label attack preserves the label of poisoned data, and the tampered image still looks like a benign sample [62], [63]. As shown in Fig 6 as an example, the fish image has been poisoned, where its latent feature, however, represents a dog. Here, the fish is the base/source instance, and the dog is the target instance. Domain experts will label such a poisoned sample as fish. Thus, the attacker needs no control of the labeling process. After the model is trained over poisoned (e.g., fish samples) and clean samples. Such a backdoored model would misclassify targeted instance, e.g., dog into the source instance, e.g., fish. Notably, the targeted instance *does not require perturbation during inference*—no trigger is used. Clean-label attack exploits feature collision. To be precise, the crafted poison fish still appears like the source fish instance in the input space or pixel level (e.g., the visual appearance), but it is close to the targeted dog instance dog in latent feature space when features are more abstract such as in the penultimate layer. The clean-label attack demonstrated in [62] is the class-specific, and the targeted success rate is low, e.g., 60%, in end-to-end attack (the victim model is trained from scratch rather than pretrained model) for a 10-class classification setting. Clean-label attack in [62] is performed under a white-box setting, where the attacker has complete knowledge of the victim model. Such constraint is eschewed in later improvements that work under black-box settings and dramatically increase attack success rate [63], [108]. Instead of feature collision attack, the efficacy of later works are realized via exploiting convex polytope attack [63] and bi-level optimization problem [108], respectively, to poison clean-label samples. In addition to image classification, clean-label attacks have been applied to backdoor video recognition models [109]. For the clean-label attacks, we note the triggers are a particular set of (testing) image samples—not a universal specific pattern. In other words, the backdoor attack is only possible for *one or a small set of inputs*—note model-reuse attack [9] under `pretrained` achieves a similar backdoor effect.

Interestingly, Saha *et al.* [110] exploits the clean-label attack in a robust manner, namely hidden trigger backdoor. Here, the trigger used in the poisoning phase or the poisoning trigger is invisible to evade human inspections. However, the trigger used in inference is visible to increase the attack robustness that is important for a physical attack. The key is that the trigger used in the poison phase shares the same latent representation with the trigger used in the inference phase. Therefore, even the inference trigger is not explicitly seen during the training phase, it can still activate the backdoor. Such a hidden backdoor attack appears to be more practical and robust compared with [89], [90], because a visible inference trigger is only revealed when an attack is presented but hidden during data poisoning via an invisible poisoning trigger. These clean-label attacks assume knowledge of the model to be used by the victim, e.g., to perform transfer learning [110]. The reason is that the attacker needs to know the latent representation.

Severi *et al.* [111] note that the training pipeline of many security vendors naturally enables backdoor injection for security-related classifiers, in particular, malware classifier. Because these vendors or companies heavily rely on crowd-sourced threat feeds (records) to create a large, diverse stream of data to train the classifier due to the difficulty of exclusively creating in-house data sources. The issue of labeling malware into families, or determining if an unknown binary is or is not malware, is labor-intensive and requires significant domain knowledge and training [112]. For example, an expert analyst can often take around 10 hours to characterize a malicious binary. This is in contrast to many current machine learning domains, like image classification, where individuals can often label with no specialized expertise and with acceptable time. This threat feeds, or data sources are built upon user-submitted binaries, which can be readily abused by attackers for backdoor attacks [111]. Severi *et al.* develop a clean-label attack to backdoor malware classifiers of both gradients boosting trees and a neural network model, even when the attacker has black-box access to the user deployed models by only poising a small fraction, e.g., 2%, of malware. The poisoned goodware/malware is still recognized as goodware/malware by anti-virus engine.

Targeted Class Data Poisoning. Barni *et al.* [113] recognize that conventional backdoor introduced by data poisoning, which *randomly* i) picks up training samples across different source classes, ii) stamps triggers, and iii) *modifies their labels to the targeted class*. Noting, in clean-label attack [62], [63], [110], the label is not modified, but the poisoned source images could be from different classes. Therefore, Barni *et al.* [113] slightly changes the way of picking up data for poisoning. It is shown that it is feasible to only stamp triggers on the data samples belonging to the targeted class—thus, the label is consistent with the content—to backdoor the model to misclassify any input to the targeted label. This is because the model can learn the association between the trigger and the targeted class. Together with an invisible, semantically indistinguishable trigger to humans, such poisoned data can evade the inspection from humans. This data poisoning strategy is used in [114] to implant backdoor with a delicate crafted natural trigger, namely refection, that is a common natural phenomenon wherever there are glasses or smooth surfaces. One trade-off of this targeted class data poisoning is the fraction of poisoned data of the targeted class needs to be high [115].

Turner *et al* [115] relaxes the above limitation [113]. One observation is that poisoned inputs can be modified to make the model harder to classify to its ground-truth label. Since these inputs will be harder to learn, the model will be *forced to learn a strong association between the trigger to the label*. In this way, the poison ratio will be reduced. Two label-consistent data poisoning methods: GAN-based and adversarial perturbations are proposed to inject backdoor. The GAN method interpolates poisoned inputs towards a source class in the *latent space*—similar to feature collision, while the poisoned input visually consistent to its label that is indeed the target label. For the adversarial perturbations, it exploits the method of crafting adversarial examples. Before adding triggers to an input, perturbation is firstly made to this input to make the model harder to learn it. Then the (invisible) trigger is added to the adversarial perturbed input to form a poisoned input. To be effective for the ASR, the attacker needs to have full knowledge of the model architecture and later training procedure [115].

Image-Scaling Attack. Image-scaling attack or scaling-camouflage attack [64] has no knowledge of training data, model, and parameters, which affects all applications that utilize scaling algorithms for resizing input images. In terms

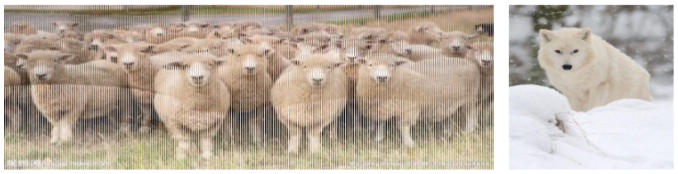
Figure 7: Image-scaling attack [64]. The left (labeled) sheep is what humans see, and the right wolf is the actual input fed into the DNN model *after scaling* to a fixed size (e.g., $224 \times 224 \times 3$) that is a commonly used resize process in deep learning. Notably, the perturbation could be more imperceptible; here, the perceptible perturbation is used for demonstration purposes.

of deep learning, an image-scaling attack abuses the scaling function in the image preprocessing pipeline that is popular in deep learning frameworks such as Caffe, TensorFlow, and PyTorch. It is common the input size to the model is fixed, e.g., $224 \times 224 \times 3$ with 224 is height and width, and 3 is the color channel. However, the original image is usually much larger than this input size and has to be resized via downsampling operation. One attack example is illustrated in Fig. 7. The 'wolf' image is embedded/disguised *delicately* into the 'sheep' image by the attacker by abusing the resize() function. When the down-sampling filter resizes the attack image (tampered 'sheep' image), the 'sheep' pixels are discarded, and the 'wolf' image is presented that is seen by the model. This abuses an inconsistent understanding of the same image between humans (tampered 'sheep' image) and machines ('wolf' image). Therefore, scaling camouflage can be effectively and stealthily exploited to perform backdoor attacks under a black-box setting; it does not require to have control over the labeling process [65]. Generally speaking, the *trigger images* can be disguised into images of the target class to poison training data by performing image-scaling attacks, thus, backdooring the trained model. In the inference phase, the trigger will easily activate the backdoored model *now without using a scaling attack*.

Notes. We can observe that the *data collection* attack is also with a wide range of victims since it is a common strategy to gain training data from the public. A trust data source is hard to be ensured. For the poisoned data, it would be hard to be distinguished when they undergo manual, or visual inspection since the content is always consistent with the label. As a consequence, not only the end-to-end trained model but also transfer learning could be infected. Feature collision is a common means of crafting label-consistent poisonous inputs to inject backdoors. However, in some cases, some knowledge of the infected model architecture is required to determine a proper latent representation.

D. *Collaborative Learning Attack*

Collaborative learning aims to learn a joint/global model with distributed clients but does not access data on the participant side. Federated learning is the most popular collaborative learning technique nowadays [116], [117]. Federated learning interacts with clients many rounds. As shown in 8, in each round, ① the server sends a joint model—it is randomly initialized in the first round—to all clients, each client trains this model over her local data for one or more local epochs—now the updated model is termed as local model. ② Then the server collects local models from a fraction of client—could be all clients—and ③ applies an aggregation algorithm, e.g., average, to update the joint model. This process continues until the model converges or reaches the predefined criterion. In this way, the joint model is trained, while local data never leaves the client's hand. The collaborative learning can significantly protect sensitive data privacy without seeing it, it is, however, vulnerable to backdoor attacks.

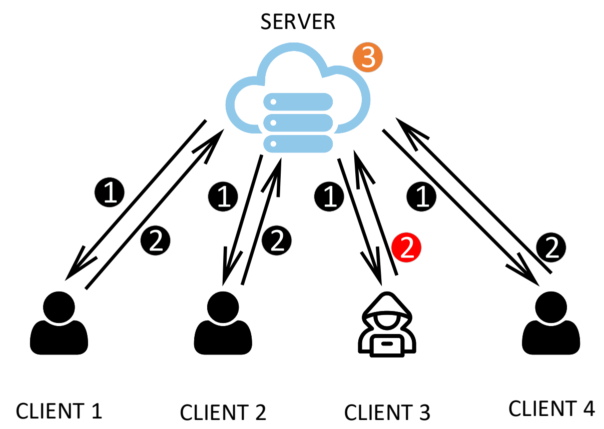

Figure 8: Illustration for federated learning with four clients.

Bagdasaryan *et al.* [11] apply *model replacement* to implant backdoor into the joint model. Malicious client, e.g., client 3, controlled by the attacker uploads modified local model, e.g., in step ② as shown in Fig. 8 that are manipulated via optimization, to the server. As a result, the the joint model will be backdoored. This model replacement strategy is also referred to as model poisoning [11], [71], [72]. It is shown that the joint model is with 100% ASR *immediately after* the backdoor being inserted even if a *single client* is selected to contribute for the joint model update in *only one round* (single-shot attack). The ASR does diminish as the joint model continues to learn. By controlling no more than 1% clients, the joint model is with on par accuracy with the clean model and now cannot prevent the backdoor being unlearned afterward [11]. The backdoor attack on federated learning is hugely challenging to defeat as the data access is not allowed in principle. Byzantine-resilient aggregation strategies are ineffective against such attack [71]. Even worse, when secure aggregation is adopted to enhance privacy further, the defense would become more difficult [11], [72]. When the backdoor defense requires no training data, e.g., DeepInspect [118], it inverts the model to extract the training data, which unfortunately potentially violates the data privacy preservation purpose of adopting federated learning. Though Sun *et al.* [72] demonstrate norm clipping and weak differential privacy may limit the attack to some extent, they tend to be easily evaded with slight increased attention and efforts by the attacker, especially when the adaptive attack is considered [11], [71].

Xie *et al.* [119] investigate distributed backdoor attack (DBA) on federated learning by fully taking advantage of the distribute characteristics. Instead of assigning all attacker-controlled clients with the same global trigger to poison their local models, the DBA decomposes the global trigger into different (non-overlap) local triggers. Each malicious client poisons her local model with her own local trigger. As an example with four stickers that each sticker acts as one local trigger is shown in Fig. 4 (b). The joint model will not exhibit a backdoor effect if only one local trigger presents but will show the backdoor effect for the combined local triggers—the global trigger during the attack phase. It is shown that the DBA is with better ASR and more robust against robust federated learning aggregation methods than previous backdoor attacks on federated learning [119]. It is worth mentioning that both DBA and conventional (global) trigger attacks achieve the same attack purpose—using the global trigger to misdirect the backdoored joint model.

Chen *et al.* [120] study backdoor attack on the federated meta-learning. Meta-learning, also known as "learning to learn," aims to train a model that can learn new skills or adapt to new environments rapidly with a few training examples. Therefore, federated meta-learning allows clients to have a joint model to be customized to their specific tasks, even the data distribution among clients vary greatly. This means the final model is built upon the joint model with further updates with his specific local data, e.g., through fine-tuning. As one prerequisite, the attacker needs to know the target class of the victim. The attack surface of federated meta-learning is alike `pre-trained` but with a weaker control on the joint model indirectly—attacker has full control over the pre-trained model under `pre-trained`. As a result, it is shown that properly adopt the (backdoored) joint model to a client-specific task can, to a large extension, reduce the backdoor effect [120]. Specifically, the authors propose a sanitizing fine-tuning process building upon matching networks [121] as a countermeasure, where the class of input is predicted from the cosine similarity of its weighted features with a support set of examples for each class. It is shown the ASR, in this regard, is reduced to only 20% validated through Omniglot and mini-ImageNet datasets.

Notes. On the one hand, collaborative learning becomes increasingly used. On the other hand, it is inherently vulnerable to backdoor attacks because it is hard to control the data and the local model submitted by malicious participants. Due to the privacy protection, the training data is not allowed to be accessed by the defender, in particular, the server or the model aggregator. Such restriction is also applicable when training a model over encrypted data such as CryptoNet [73], SecureML [74] and CryptoNN [75]. This makes the defense to be extremely challenging. Because most defense mechanisms indeed need a (small) set of hold-out validation samples to assist the backdoor detection. It is noticed that DeepInspect [118] does not need to access training data. Unfortunately, it, however, inverts the model to reverse engineer the training data, which could partially violates the data privacy preservation purpose of adopting collaborative learning.

E. Post-Deployment Attack

This attack tampers the model during the post-deployment phase and thus does not rely on data poisoning to insert backdoors. However, it inevitably needs model access, which can be realized by firstly insert a malicious software program. Stealthier, it is feasible considering the rising of side-channel attacks, more specifically, utilizing the row-hammer attack [126]. The row-hammer attack allows an attacker to tamper the memory content if the victim (CNN model) is co-located at the same physical machine, typical in today's cloud service. For example, many tenants are sharing the same physical machine with separation from virtual machine [127].

Weight Tamper. As an early work, Dumford *et al.* [78] examine the feasibility of backdooring a face recognition system through solely perturbing weights that are resided in specific layers—a similar study is demonstrated in [77]. Both work [77], [78] intuitively assume that the attacker somehow can access the model resided in the host file storage system or memory, e.g., through a toolkit, and tamper the weight after model deployment. It demonstrates the possibility of backdooring a model through perturbing the weights, though it is hugely computationally hungry to achieve a good attack successful rate, e.g., merely up to 50% [78]. The main reason is an extensive iterative search over the weights that will be tampered for the sake of optimizing the ASR [78]. Costales *et al.* [77] takes the evasion defense as an objective when searching for parameters to be modified to bypass existing online inspection defense, specifically, evading the STRIP [82] (STRIP is detailed in section V).

Bit Flip. Rakin *et al.* [76] recognize the advantage of flipping vulnerable bits of DNN weights to introduce backdoor. To determine which bits need to be flipped, the last-layer neurons with the most impact on the output for the targeted class are found using a gradient ranking approach. This eventually requires knowledge of both model architecture and parameters, which could be empowered by model extraction attack [128]. In this context, it is worth mentioning that recent microarchitectural side-channel attacks show the feasibility of gaining such information, including revealing architecture [129] and stealing parameters [130] in the cloud with the existing shared covert-channel. The trigger is then generated using a minimization optimization technique. Then, the original weight matrix and the final optimized malicious weight matrix are compared—the weight here is quantified to 8-bit, providing the information on which bits need to be flipped. A small set of test samples is required to determine the malicious weight matrix. The backdoor is inserted through a row-hammer attack to flip the identified bits in main memory—but this step is not really implemented in [76]. In one experiment, the authors [76] can achieve backdoor effect with a small number of bit flips, e.g., 84 out of 88 million weight bits on ResNet18 [131] on CIFAR10. It is acknowledged that, when backdooring a model trained over the large-scale ImageNet using this bit flip approach, the CDA and ASR drops are relatively noticeable—the attacking task becomes harder to the attacker. On the other hand, such post-deployment attacks via bit flipping—a kind of fault injection

Table I: Backdoor attack summary under categorized attack surfaces. The 3rd to 7th columns qualitatively compare the attacker's capabilities under the corresponding attack surface.

Attack Surface	Backdoor Attacks	Access Model Architecture	Access Model Parameters	Access Training Data	Trigger controllability	ASR	Potential Countermeasure [1]
`Code Poisoning`	[51] [52]	Black-Box	○	○	◐	High	Offline Model Inspection Online Model Inspection Online Data Inspection

`Outsourcing`	Image [6], [7], [12], [88], [122] [8];
	Text [13] [14]–[16];
	Audio [16], [17];
	Video [85];
	Reinforcement Learning [21], [97] [98]
	(AI GO [22]);
	Code processing [99], [100];
	Dynamic trigger [95]
	Adaptive Attack [102];
	Deep Generative Model [20];
	Graph Model [23]
		White-Box	●	●	●	Very High	Blind Model Removal
							Offline Model Inspection
							Online Model Inspection
							Online Data Inspection

`Pretrained`	[7], [56]
	Word Embedding [54];
	NLP tasks [107];
	Model-reuse [9];
	Programmable backdoor [53];
	Latent Backdoor [57];
	Model-agnostic via appending [106];
	Graph Model [101]
		Grey-Box	◐	◐	◐	Medium	Blind Model Removal
							Offline Model Inspection
							Online Model Inspection
							Online Data Inspection

`Data Collection`	Clean-Label Attack [62], [63], [110] [114],
	(video [85], [109]),
	(malware classification [111]);
	Targeted Class Data Poisoning [113], [115];
	Image-Scaling Attack [64], [65];
	Biometric Template Update [123];
	Wireless Signal Classification [19]
		Grey-Box	◐	◐	◐	Medium	Offline Data Inspection
							Online Model Inspection
							Online Data Inspection

`Collaborative Learning`	Federated learning [11], [71], [72], (IoT application [70]); Federated learning with distributed backdoor [119]; Federated meta-learning [120]; feature-partitioned collaborative learning [124]	White-Box	●	●	●	High	Offline Model Inspection [2]
`Post-deployment`	[78] [76], [77] Application Switch [125]	White-Box	●	●	◐	Medium	Online Model Inspection Online Data Inspection

●: Applicable or Necessary; ●: Inapplicable or Unnecessary; ◐: Partially Applicable or Necessary.
[1] The detailed countermeasure methods are specified in section V and summarised in Table II. We consider they are *potential*, because it is recognized that there is no single one-against-all countermeasure and they could be bypassed, especially by adaptive attackers.
[2] This kind of backdoor attack is very challenge to defeat as the user, e.g., the model aggregator, may not even be able to access any testing data. Rare defense requires no access to training data [118].

attack—are not restricted to *backdoor attacks*. It can be used to degrade the model overall performance significantly [80], [132], [133], e.g., making all inputs accuracy to be low to random guess that is akin to denial of service. In this case, the bits can be *randomly flipped*.

TrojanNet. Guo *et al.* [125] propose TrojanNet that eventually switches the backdoored model from its public/main application task, e,g., traffic sign recognition, to a *different* secret application task, e.g., face recognition given the presence of the trigger. This is a different backdoor strategy, since the public task and secret task are for *totally differing applications*. For all other backdoor attacks, the attacker's task and the default benign task are *within the same application*. The public task and secret task of TrojanNet share no common features, and the secret task remains undetectable without the presence of a hidden trigger—here is a key [125]. Generally speaking, the secret task and public task share the *same model parameter*. Switching from public task to secret task is realized by parameter permutation, which is determined and activated by a key/trigger. As stated by the authors, parameter permutation can be done at loading time or run-time in memory when a trigger is activated. Such a trigger can be embedded in malicious software. Training the public task and secret together is still akin to multi-task learning, though they do not share any feature. It is shown that multiple secret tasks can co-exist with the public task, while the utility of all secret tasks and the public task can be retained. Such a backdoor attack is tough to detect.

Notes. `Post-deployment` backdoor attacks tend to be non-trivial to perform by the attacker because it requires first intruding the system to implant a malicious program [78], [125] or flipping the vulnerable bits in memory during run-time (as a type of fault injection) that needs professional expertise [76]. In addition, `Post-deployment` backdoor attack usually requires white-box access to the model. The advantage of this attack is that it bypasses all offline inspections on the model or/and the data.

F. Code Poisoning Attack

Security risks of ML framework building upon public repositories or libraries have been initially recognized in [51]. Regarding the backdoor attack, Bagdasaryan *et al.* [52] studied poisoning the code performing loss computation/function that could be exploited by an attacker to insert a backdoor into the model, which is recognized as a new type of backdoor attacks.

Backdoor insertion is viewed as an instance of multi-task learning for conflicting objectives that are training the same model for high accuracy on the *main* and attacker-chosen *backdoor task* simultaneously. In this context, the attacker can take the i) main task, ii) backdoor task, and iii) even defense-evasion objectives into a single *loss function* to train the model. Bagdasaryan *et al.* [52] take advantage of a *multiple gradient descent algorithm* with the Franke-Wolfe optimizer to construct an optimal, self-balancing loss function, which achieves high accuracy on both the main and backdoor tasks. It is also noticed that auditing[5] and testing the loss-consumption code is hard in practice [52]. This backdoor is therefore blind to the attacker, termed as *blind backdoor attack*, since the attacker has no access to the later poisoned code during its execution, nor the training data on which the training operates, nor the resulting model, nor any other output of the training process (e.g., model accuracy). To be both data and model-independent considering that neither is known in advance, trigger inputs are synthesized "on the fly". Similarly the corresponding backdoor loss is also computed "on the fly". In this manner, the backdoor loss is always included in the loss function to optimize the model for both the main and backdoor tasks.

It is further demonstrated [52] that the backdoor task can be totally different from the primary task. For example, the main task of a model is to count the number of faces in a photo, while the backdoor task is covertly to recognize specific individuals. Similar backdoor attack effect—switching application—has been shown through TrojanNet [125]. In addition, multiple backdoor tasks can be achieved using multiple synthesizers concurrently [52]. Moreover, evasion objective has been taken into the loss function to show evasion of various countermeasures [52]—this is an adaptive attack.

Notes. We regard the `code poisoning` backdoor attack is with the widest victims as it tampers the low-level ML code repository. Such an attack even requires no knowledge of the model and no access to training data but achieves high ASR.

G. *Discussion and summary*

Table I compares attacking capabilities and summarizes works under each attacking surface. Most of them attack classification task, especially, image classification. We can also observe that majority backdoor attacks are resulted from `outsourcing`, `pretrained`, `data collection`, `collaborative learning` attacking surfaces. Backdoor attack studies from `code poisoning` and `post-deployment` attack surfaces are relative less.

Unsurprisingly, `outsource` has the most robust control of the trigger and the highest ASR because the attacker has the maximum capability, including access to the model, training data, and the training process control. In contrast, it is worth noting that the defender/user is with the minimum capability when devising countermeasures under such `outsourcing` attacks. Because the defender is usually limited to ML expertise and computational resources, which are the reasons for outsourcing, the countermeasures should adequately consider the defenders' weakest capability. This consideration should also be always held when countering `pretrained` attacks because the user may usually be lack of computational resource.

[5]The codebase of loss computation code could contain dozens of thousands of lines, which are hard to understand even by experts.

Though rare studies under `code poisoning`, it appears that it has the widest attacking victims as it requires minimum attacking capability. The `pretrained` and `data collection` attack also have wide attacking victims. Except for `data collection`, the attacker is very unlikely to expose trigger inputs or poisoned data to defender until the trigger has to be presented at *attacking phase*. This explains the offline data inspection is inapplicable against attack surfaces except for `data collection`. It is very challenging for countering `collaborative learning` attacks because there is a trade-off between data privacy and security vulnerabilities. The defender, e.g., server, is even not allowed to access any testing/training data to assist the defense. In this case, the only countermeasure [118] requiring no testing data can be potentially considered. However, this countermeasure [118] first reverses engineers' training data, partially violating the purpose of adopting collaborative learning to protect data privacy. In addition, it is impossible to guarantee that none of the heterogeneous resided clients of collaborative learning is malicious as the number of clients could be high, e.g., hundreds of thousands. For the `pretrained` attack, it is worth to mention that some knowledge of the downstream student model tasks or/and training data is usually necessary to perform the attack. For both `pretrained` and `post-deployment` attacks, the trigger is usually delicately generated through some optimization algorithm, e.g., gradient-based approach, rather than arbitrarily chosen in order to achieve high ASR. Therefore, the attacker's flexibility of crafting arbitrary trigger is, more or less, deteriorated.

Moreover, we recommend that attack studies always explicitly identify their threat models to clarify the attacker's capabilities.

V. Backdoor Countermeasures

We firstly categorize countermeasures into four classes: blind backdoor removal, offline inspection, online inspection, and post backdoor removal, as illustrated in Fig. 9. These four classes are, somehow, align with possible sequential defense steps that could be considered. We performed a systematic analysis of countermeasures under these four classes. At the end of this section, we provide comparisons, summaries, and discussions.

A. *Blind Backdoor Removal*

The main feature of this defense categorization is that it does not differentiate the backdoored model from a clean model, or trigger input from clean input. The main purpose is to remove or suppress the backdoor effect while maintaining the CDA of clean inputs, especially considering that it is eventually a clean model. Notably, it is worth to mention that the blind backdoor removal can be performed offline or/and online.

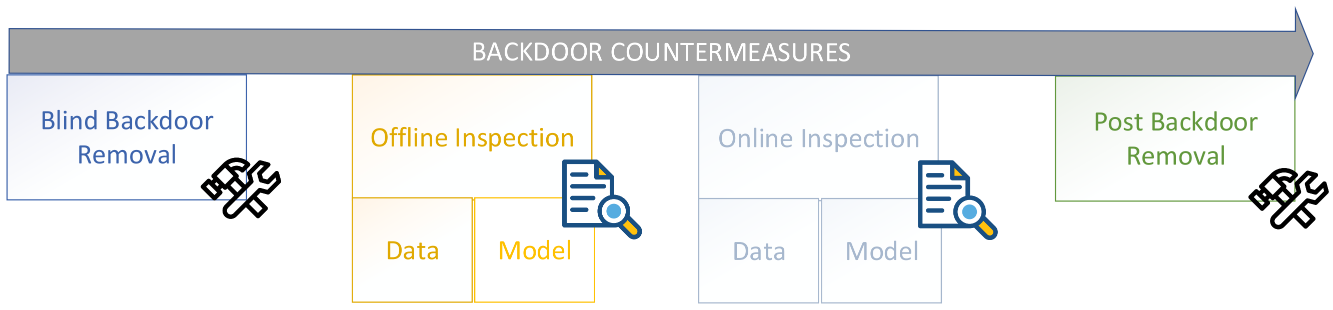

Figure 9: Categories of backdoor countermeasures: blind backdoor removal without first check whether the model is backdoored or not; offline inspection on the data or model; online inspection on the (upcoming) data or model; post backdoor removal or restoration after backdoor trigger is identified.

Fine-Pruning. As an earlier defense, Liu *et al.* propose to remove potential backdoor by firstly pruning carefully chosen neurons of the DNN model that contribute least to the main classification task [134]. More precisely, neurons are sorted by their activation on clean inputs (e.g., the hold-out validation samples) and pruned in the least-activated order. One general assumption is that the neurons activated by clean and trigger inputs are different or separable. After pruning, fine-tuning is used to restore model performance. Nonetheless, this method substantially degrades the model accuracy [81]. It is also cumbersome to perform fine-pruning operations to any DNN model, as most of them are more likely to be benign. A similar approach presented in [6] incurs high complexity and computation costs.

Februus. Doan *et al.* propose Februus [135] to eliminate the backdoor effect by online detecting and to sanitize the trigger input at run-time. It can serve as a filter deployed in front of any neural networks to cleanse the (trigger) inputs. The idea is to utilize a Visual Explanation tool to recognize the potential trigger region. Once the region is identified, Februus will surgically remove the malicious area and replace it with a neutrally gray color. This surgically removing step has already eliminated the trigger effect and potentially degrade the classification of the deep neural networks. One additional step, image restoration, was proposed using a GAN-based image inpainting method to restore the damaged areas to its original likeliness before being contaminated. This method is robust against multiple backdoor variants, especially the class-specific backdoor variant. However, as the method involves removing and restoring the images, it is sensitive to a large-size trigger covering a considerable portion of the images.

Suppression. Sarkar *et al.* [136] propose a backdoor suppression via fuzzing that entails building a wrapper around the trained model to neutralize the trigger effect. Given each input, many replicas are made: each replica is perturbed by adding some noise—noise level is empirically determined [136]. All perturbed replicas are fed into the DNN model, regardless of backdoored or clean, to collect predictions. It is worth to mention such a strategy is previously exploited by [82]. The final prediction of the input is based on the majority voting of predictions of its perturbed replicas. The evaluations on MNIST and CIFAR10 dataset demonstrate [136] that around 90% and 50% of trigger images are reverted to their true predictions, respectively, which are less effective than other run-time inspection methods, e.g., [82]. On the other hand, it is acknowledged that this method compromises the CDAs for clean inputs relatively unacceptable [41]. Thus, it is less suitable as a wrapper around the trained model. In addition, the parameters such as the level of noise and the number of replicas that define this wrapper are empirically searched and determined, not generic. Moreover, this method appears to be reasonable only when the trigger is quite small[6]. Under such a condition only, fuzzing the inputs could suppress the trigger effect on hijacking the prediction.

ConFoc. The ConFoc [137] enforces the (clean or backdoored) model to focus on the content of the input image to remove backdoor effects while discarding the style information potentially contaminated by the trigger. It produces true labels—not the attacker targeted label—of the trigger input that is consistent with the input content, which retains the CDA with clean inputs as well. The rationale is that the image consists of both content and style information [138]. The former refers to the shapes of the object or the semantic information, while the latter refers to the colors or texture information of images. It is hypothesized that focusing on content only is similar to human reasoning when making classifications. Therefore, the ConFoc [137] retrains the model to make classifications, mainly relying on the content information. One main limitation of the ConFoc is that it assumes the trigger does not overlap with the interested object—with content information—in the input image, which is inapplicable when setting triggers being overlapped with the object, e.g., those triggers in Fig. 4.

RAB. Weber *et al.* [46] propose a provably robust training process for classifiers/models against backdoor attacks—a simpler concurrent work is in [139]. Generally, it [46] exploits randomized smoothing to mitigate the trigger effects. It is worth to note that certified robustness is originally devised to against adversarial examples [140]. More precisely[7], firstly,

[6]The authors evaluate triggers covering at most 5 pixels [41] for the MNIST dataset.

[7]We informally describe it to ease the general understanding.

given the poisoned data, each sample is added certain noise. This creates one so-called smoothed training set. Such process is repeated to create a number of, e.g., 1000, replicated smoothed training set. Secondly, each smoothed training set is used to train a model, during training process, differential privacy can be further applied to improve the 'smoothness' of the trained model. Therefore, a number of, e.g., 1000, models need to be trained. Thirdly, during inference time, an input x is randomly perturbed according to certain noise bound to make replicas. All of x purturbed replicas are fed into *each* trained model to gain the prediction label according to e.g., majority voting. Then the prediction labels from each model—in total, e.g., 1000 models—is further aggregated and the final label of x is produced according a criteria, e.g., again majority vote. The purpose is that the final label will be the correct label that the clean model produces given the trigger perturbation is bounded. Such robustness can be proved. Though the provable robustness is desirable, the RAB [46] has strict conditions such as the trigger perturbation bound must be small, which means that it can be easily bypassed in practice whenever the perturbation of the trigger exceeds the bound. In addition, there are many 'smooth' models required to be trained, which greatly increases the computational overhead. Moreover, to guarantee the provable robustness, the RAB inexplicitly needs to have knowledge of the fraction of the poisoned data and the perturbation bound type, e.g., l_2-norm, which tend to be not practical. Notably, the RAB assumes access to poisoned data. As this countermeasure, to a large extent, is performed online that is to eliminate the trigger effect given an input x, so that it turns to be cumbersome when deployed for real-time applications as it could have relative unacceptable latency.

Notes. Blind backdoor removal does not tell the backdoored model from a clean model. As we can see, fine-pruning usually renders to CDA drop that could be unacceptable. Other works suppress or remove trigger effects if the model is infected. However, we note that suppression makes, somehow, the unrealistic assumption about the trigger size. As for ConFoc, it is only applicable to specific tasks where the image consists of both content and style information. Except for fine-pruning, all other works are inapplicable to other domains. They are all devised explicitly for the vision domain.

B. Offline Inspection

Both *backdoored model* and *trigger input* (or poisoned data) can be inspected offline. However, poisoned data inspection is only feasible under `data collection`. It is implausible for other attack surfaces where the attacker will not expose the poisoned data to the defender.

1) Data Inspection: Data inspection assumes that the poisoned data are available for the defenders since the attacker exploits data poisoning in order to insert backdoors.

Spectral Signature. Tran *et al.* [141] explore spectral signature, a technique based on (robust) statistic analysis, to identify and remove poisoned data samples from a potentially compromised training dataset. Firstly, a model is trained on a collected dataset that can contain poisoned data points. Secondly, for each particular output class/label, all the input instances for that label are fed into the model, and their *latent representations* are recorded. Thirdly, singular value decomposition is performed on the covariance matrix of the latent representations extracted, e.g., from the penultimate layer, which is used to compute an outlier score for each input. Fourthly, the input sample with a higher score than a ration is flagged as a trigger input, and removed from the training dataset on which a clean model will be trained again. This defense succeeds when the latent representations of clean inputs are sufficiently different from that of trigger inputs. Therefore, a crucial step is determining the means of obtaining proper latent representation to expose the trace of the trigger [99]. However, it points out that the outlier ration is fixed to be close to the ratio of corrupted samples in the target class. This requires some knowledge of the poison ratio and target class [142], [143], which turns to be unknown in practice.

Gradient Clustering. Chan *et al.* [142] use cluster concept in a slightly different means that is not clustering *latent representations*. Chan *et al.* hypothesize and then show that a trigger image sample could lead to *a relatively large absolute value of gradient in the input layer* at the trigger position. Based on this, trigger samples can be separated from clean samples using a clustering algorithm. As the authors can detect the infected targeted and source classes, the infected model can be retrained by relabeling the trigger samples to their correct label to unlearn the backdoor effect, rather than training the model from scratch using sanitized data that has removed the corrupted data.

Activation Clustering. Chen *et al.* [144] propose an activation clustering (AC) method. It is noticed that activations of the last hidden layer reflect high-level features used by the neural network to reach a model decision. Given the collected data and the model trained with the data, each sample is fed into the model, and the corresponding activation is collected. The activations of inputs belonging to the same label are separated and clustered by applying k-means clustering with $k = 2$ after dimension reduction. Since the 2-means clustering will always separate the activations into two clusters, regardless of whether poisoned data is present or not, some metric to judge is used. For example, a high *silhouette score* means that the class has been infected or poisoned. Once the poisoned data are identified, they can be removed, and a clean model is trained. The other model restoration is relabeling the corrupted data to their correct source label to retrain the backdoored model to unlearn the backdoor effect, which could save time.

Deep k-NN. In [145], Peri *et al.* design a deep k-NN method to detect *clean-label poisoned* samples, which can effectively against both feature collision and convex polytope clean-label attacks, see subsection IV-C. It demonstrates to detect over 99% of poisoned examples in both clean-label attacks over the CIFAR-10 dataset. Consequently, those detected poisoned samples are removed without compromising model performance. Particularly, it has been shown more robust than l_2-norm, one-class SVM, and Random Point Eviction defenses against Feature Collision Attack, while comparable with l_2-norm defense but remove less clean images in Convex Polytope Attack.

SCAn. Unlike most countermeasures focus on class-agnostic triggers, Tang *et al.* [146] propose a statistical contamination analyzer (SCAn), as a means of detecting both class-agnostic and class-specific triggers, see subsection III-B. It decomposes the latent representation of an image, in particular, the feature from the last third layer or the layer before logits, into two components: a class-specific identity, e.g., facial features distinguishing one person from others in face recognition task (that is a between-class component), and a variation component, e.g., size, orientation (within-class variation component). Then it is hypothesized that the variation component of an input image is independent of the label. This means the variation component distribution learned from one label/class is transferable to other labels. For example, smile as a variation component in face recognition is dependent on an individual's identity—the variation component is universal. Tang *et al.* [146] decompose all images belonging to each class to obtain *finer-grained information*—in comparison with, e.g., the information gained through simple activation clustering [144]—on trigger impacts for classification. Using statistical analysis on the decomposed components, it is noted that the latent representation of images in the attacker target class (infected class) is a mixture of two groups from trigger inputs and clean inputs. Each is decomposed into distinct identity and universal variation components. Recall that the trigger images are from a different class/identity but (mis)labeled to the target label. So that the infected class can be identified. The SCAn has one additional main advantage that is also applicable to class-specific triggers. As a slight contradiction, it eventually assumes a small set of hold-out clean validation set, must contain no trigger, from in-house collection under this `data collection` surface. Although this assumption might be held in some cases. In addition, the SCAn is specific to image classification tasks where a class label is given to each object (face, flower, car, digits, traffic sign, etc.), and the variation applied to an object (e.g., lighting, poses, expressions, etc.) *is of the same distribution across all labels*. Such a criterion may not always be satisfied. Moreover, it is acknowledged [146] that SCAn is less effective to multiple target-trigger attack, see subsection III-B.

Differential Privacy. Du *et al.* propose an entirely different strategy that applies differential privacy when performing model training to facilitate the detection of outliers [147]. Applying differential privacy gains a trained model that is also named as a naturally smoothed model [46]. This strategy can be effectively used to detect poisoned data because poisoned data can be viewed as outliers. Differential privacy is a means of protecting data privacy. It aims to hide specific input data from the output so that one can not tell whether the input data contain a particular record or not by looking at the result calculated from input data. When differential privacy is applied to the machine learning algorithm, the trained model becomes *insensitive to the removal or replacement of an arbitrary point in the training dataset*. More specifically, the contribution from rarely poisoned samples will be hidden by random noise added when training the model, so that the model underfits or suppresses the poisoned samples. This means the backdoor effect is already greatly suppressed by the model—ASR is quite low. In addition, the model will be less confident when predicting atypical poisoned examples. Therefore, by measuring the loss as a metric, one can distinguish poisoned inputs because it has a higher loss score. It is shown that this differential privacy based backdoor countermeasure [147] can effectively suppress backdoor effects as well as to detect poisoned samples, even the source-specific trigger inputs. As the validation is only through the toy MNIST dataset, it is worth further validating the efficacy of this method with more complex datasets trained by representative model architectures.

Notes: We can see that most of these countermeasures are based on a clustering algorithm applied on, e.g., latent representation, gradient, and activation. We note that these countermeasure may not be applicable for the special image-scaling based backdoor attack under `data collection` attack surface (see subsection IV-C). To defend such an attack, we need to carefully choose the downsampling filter parameters during resizing to remove the image-scaling attack effect [148]. For the other special clean-label poisoning attack that is the targeted class data poisoning by only stamping triggers on the data belonging to the targeted class [113], it is unclear whether the above data inspection methods are effective or not. Because they assume the source class is a different target, violating the attack setting of [113] that source class stamped with the trigger is the same as the target class. In addition, most studies do not explicitly consider reporting results of, e.g., false positive, when the dataset under examination is eventually benign.

2) Model Inspection: Unlike data inspection that assumes access to poisoned data. All countermeasures in this part use a more realistic assumption that the poisoned data is not available. Therefore, they are suitable for countering backdoor attacks resulted from various attacking surfaces—not only limited to `data collection`.

Trigger Reverse Engineer. In Oakland 2019, Wang *et al.* [81] propose the NeuralCleanse to detect whether a DNN model has been backdoored or not prior to deployment, where the performance is further improved in TABOR [84] by utilizing various regularization when solving optimizations. The other improvement with a similar concept of NeuralCleanse is studied in [86], although the specific methods can be varied. NeuralCleanse is based on the intuition that, given a backdoored model, it requires much smaller modifications to *all input samples* to misclassify them into the attacker targeted (infected) label than any other uninfected labels. In other words, a *shortest perturbation path* exists to *any input* to the target label. Therefore, NeuralCleanse iterates through all labels of the model and determines if any label requires a substantially smaller modification to achieve misclassifications. One advantage of this method is that the trigger can be reverse-engineered and identified during the backdoored model detection process. Once the trigger is identified, backdoor removal (detailed in Section V-D) can be performed via retraining to unlearn the trigger effect.

Despite its novelty, NeuralCleanse is still with some limitations. Firstly, it could incur high computation costs *proportional to the number of labels*. The computation cost of the detection process can take several days for specific DNN models, even when the optimization is considered. This is especially the case when the number of classes of the task is large. Secondly, like most backdoor countermeasures, the method is reported to be less effective with the increased trigger size. Thirdly, it requires training reference models to determine a (global) threshold to distinguish clean models from backdoored models, which inadvertently appear to be inapplicable under `outsourcing`. Because this violates the motivation of `outsourcing` where the user has limited computational resource or/and ML expertise.

It is observed that the reversed trigger is not always consistent with the original trigger [55], [149], [150], e.g., in different runs [149]. Consequently, when the reversed trigger that is inconsistent with the original trigger is utilized for backdoor removal, the removal efficacy will be significantly decreased. That is, the input with the original trigger still has a considerable attack success rate to the retrained model. Accordingly, Qiao *et al.* [149] generatively model the valid trigger distribution, rather than a single reversed trigger, via the max-entropy staircase approximator (MESA) and then use many valid triggers to perform backdoor removal to increase the removal efficacy. However, this work relies on several *strong assumptions*, such as knowledge of the model being backdoored and the shape and size of the trigger, which are not available in real-world to defenders. In addition, validations in [149] are limited by a small trigger, a 3×3-pixel square, where the concerned computational overhead of this work is, however, not reported.

Harikumar *et al.* [150] propose Scalable Trojan Scanner (STS) in order to eliminate one limitation of the NeuralCleanse: the computational overhead is linearly increased with the number of classes in the dataset. The STS reverse engineers the trigger in a means of identifying a trigger (perturbation) that results in prediction vectors, e.g., softmax, of *all images* to be similar. In this context, the trigger searching (reverse engineer) is *independent* on the number of classes (this is where the *Scalable* comes from). In other words, it does not care about the target class when reverse engineering the trigger. The STS saves time in comparison with NeuralCleanse when searching the minimum perturbation—the identified trigger—through optimization algorithm. However, in [150], it only tests triggers with *small size and fixed shape* (square). It appears that the STS is still ineffective to large trigger size.

NeuronInspect. Under similar concept with SentiNet [38] and Februus [135] exploiting *output explanation*, Huang *et al.* [151] propose NeuronInspect to combine output explanation with outlier detection considering that the explanation heatmap of attacker target class—treated as outlier—differs from non-target class. In other words, the target class's heatmap is hypothesized to be i) compact, ii) smooth, and iii) remains persistent even across different input images. The persistence is an exhibition of the trigger input-agnostic characteristic. In comparison with NeuralCleanse, NeuronInspect has advantages, including reduced computational overhead and robustness against a multiple-trigger-to-same-label backdoor variant (V1 in subsection III-B). However, unlike NeuralCleanse [81], TABOR [84] and STS [150], NeuronInspect does not reverse engineer and discovers the trigger.

DeepInspect. Chen *et al.* propose DeepInspect [118] to detect backdoor attack with *no requirement of accessing training data*, because it first reverse engineers training data. The key idea of DeepInspect is to use a conditional generative model to learn the probabilistic distribution of potential triggers. This generative model will be used to generate reversed triggers whose perturbation level will be statistically evaluated to build the backdoor anomaly detection. DeepInspect is faster than NeuralCleanse [81] to reverse triggers in complex datasets. However, due to the weak assumption of the defender's capability that has no access to training data, the detection performance of DeepInspect appears to be worse than NeuralCleanse.

AEGIS. Soremekun *et al.* [152], for the first time, investigate the backdoor attacks on *adversarial robust model that is elaborately trained defending against adversarial attacks* [153]. It is shown that the adversarial robust model, robust model for short in the latter description, is vulnerable to backdoor attacks, though the robust model is robust against adversarial example attacks. Correspondingly, they propose AEGIS, a method exploiting latent feature clustering to detect *backdoored robust model* accurately. One exploited the unique property of a robust model is that it can be adopted for image generation/synthesizing [153]. Therefore, for each class, the robust classifier *itself* is used to generate synthesized images. Then latent features of both synthesized images and held-out validation images are extracted, followed by feature dimension reduction. After that, a clustering algorithm, namely the shift algorithm [154], is used to automate the detection process. If there are no more than two clusters—one is the distribution of the validation samples, one is the distribution of the synthesized images uninfected by triggers, then this robust model is deemed as clean, otherwise backdoored because now there are other distributions formed by synthesized images infected by triggers. Clustering the input samples with their latent representations is not new, which has been used for trigger data inspection [144]. The difference is that previous work [144] requires to access poisoned trigger data, while AEGIS does not. The limitation of AEGIS is only applicable for robust models, not standard models. It is also acknowledged that defeating specific triggers, e.g., blending trigger as in Fig. 4 (a), induced backdoor appears to be a challenge for AEGIS.

Meta Classifier. Unlike the above works dedicated to the vision domain, Xu *et al.* [155] consider a generic backdoor detection applicable to diverse domains such as audio and text. It trains many (e.g., 2048) clean and (e.g., 2048) backdoored shadow models by the defender to act as training samples of a meta-classifier—the other neural network—to predict whether a new model is clean or not. Note that the shadow model is sometimes called a surrogate model. To approximate the general distribution of backdoored models, the shadow backdoored models are trained over diverse trigger patterns and backdoor goals. To generate feature input to the meta-classifier, many query inputs are made to each shadow model, and confidence scores from the shadow model are concatenated, acting as feature representation of the shadow model. In contrast, the output of the meta-classifier is binary. The number of the query inputs is set as 10 [155]. The query inputs can further be optimized along with the training of the meta-classier, rather than randomly selected to gain accurate feature representation of the shadow model [155]. This method has the advantage of its generalization across different domains. It is worth to mention that adaptive attacks are considered explicitly by the authors to evaluate the defense robustness. However, this method requires both ML expertise and expensive computational resources, which might be less practical.

Universal Litmus Pattern. Colouri *et al.* [156] apply similar concept as above [155] to train a meta-classifier. Like [155], a small set of specifically chosen image inputs, here namely, universal litmus patterns [156], query the clean and backdoored shadow models, then the logits (from the penultimate layer) are concatenated as feature input to the meta-classier, whether the model is clean or backdoored is the output of the meta-classifier. The universal litmus patterns are crafted with the assistance of the technique in [81]. When a new model is coming, these universal litmus patterns are fed into it and collect the new model's logits as a feature to the meta-classifier to tell whether this new model is backdoored. Both [155] and [156] require train enormous clean and backdoored *models* as prerequisites, the cost is high. This would worsen because the meta-classifier is dataset dependent. For example, if the meta-classifier is for CIFAR10, a new model trained over MNIST cannot be distinguished by this meta-classifier. Therefore, such computational expensive shadow model training has to be performed when dataset changes. In addition, in comparison with [156] that is generalize to diverse domains, the universal litmus pattern is specific to image inputs. Moreover, the efficacy of universal litmus pattern [155] is achieved with a strong assumption: the trigger size is inexplicitly assumed to be known.

Notes. As the defender is with less capability, in particular, no access to poisoned samples, it is unsurprising that offline model inspection usually requires high computational overhead and ML expertise. In this context, defenders under `outsourcing` and `pretrained` may not be able to adopt these countermeasures. Because they are usually lack of computational resources or/and ML expertise. For example, the countermeasures relying on meta classifier need train many shadow models that are computational hungry. In addition, their efficacy [155], [156] is potentially limited by the variety of the triggers used when training the meta classifier [150]. Most of the countermeasures cannot deal with large-sized triggers, especially for countermeasures aiming to reverse engineer the trigger.

C. Online Inspection

Online inspection can also be applied to monitor the behavior of either the model or input data during run-time.

1) Data Inspection: Data inspection checks the inputs to decide whether a trigger is contained or not through anomaly detection, thus, denying the adversarial input and then throwing alert if set.

SentiNet. In 2018, Chou *et al.* [38] exploit both the model interpretability/explanation and object detection techniques, referred to as SentiNet, to firstly discover contiguous regions of an input image important for determining the classification result. This region is assumed to have a high chance of possessing a trigger when it *strongly affects* the classification. Once this region is determined, it is carved out and patched on to other held-out images with ground-truth labels. If both the misclassification rate—probability of the predicted label is not the ground-truth label of the held-out image—and confidence of these patched images are high enough, this carved patch is regarded as an adversarial patch that contains a backdoor trigger. Therefore, the incoming input is a backdoor input.

NEO. Udeshi *et al.* [157] proposed NEO that is devised to search and then mitigate the trigger by the *dominant color* in the image frame when it appears in the scene at run-time. The method is claimed to be fast and completely black-box to isolate and reconstruct the trigger for defenders to verify on a skeptical model. However, by occluding the trigger with the dominant color, the method is not robust against large-sized triggers, and the classification accuracy after the NEO method is also degraded since the main feature is sometimes occluded. In addition, the validation is only performed via a trigger that is with a square shape, which is not generalized.

STRIP. In 2019 ACSAC, Gao *et al.* [82] propose STRIP. They turn the input-agnostic strength of the trigger into a weakness exploited to detect the poisoned input. Suppose a model has been backdoored. On the one hand, for a clean input, the prediction z should be quite different from the ground-truth given a strong and intentional perturbation is applied to it. Therefore, replicas of a clean input with varying perturbation exhibit strong randomness—quantified via the entropy metric. On the other hand, for trigger inputs, the prediction z should usually be constant to the attacker's target z_a even under perturbations because of the strong hijacking effect of the trigger. Therefore, replicas of the trigger input with varying perturbation exhibit weak randomness. Given a preset entropy threshold—can be determined solely using clean input, the trigger input (high entropy) can be easily distinguished from the clean input (low entropy), consequently, throwing alert for further investigations. The STRIP is simple to use, a non-ML method, yet efficient and generic, validated not only by the vision domain but also by text and audio [16]. It is robust against arbitrary triggers, e.g., any size—this is always a limitation for most countermeasures [38], [81], [84], [149]. The STRIP concept is later adopted in [158] to identify triggers in text for countering backdoor attacks on LSTM-based models, but under the `data collection` attacking surface. One limitation of STRIP is that it is mainly designed for class-agnostic triggers that would not be efficient for class-specific triggers.

Epistemic Classifier. Yang *et al.* [159] propose an epistemic classifier to leverage whether the prediction of input is reliable or not to identify inputs containing triggers—adversarial input is unreliable. The concept is similar to [160]. Generally, an input is compared to its neighboring training points—determined online—according to the specific distance metric that separates them in the latent representations across multiple layers. The labels of these searched neighboring training points in each hidden layer are used to check whether the intermediate computations performed by each layer conform to the final model's prediction. If they are inconsistent, input prediction is deemed unreliable, which is a trigger input, otherwise, clean input. It is generally based on the hypothesis that input with trigger might start close to a clean training instance—from the source class—in the input layer, but its trajectory over the neural network will slowly or abruptly close to the attacker's chosen targeted class. This concept is similar to NIC (detailed soon in the below) [161]. Such abnormal deviation can be reflected by observing the consistency of the labels of input's neighboring training points across multiple layers. Even though this work [159] is insensitive to trigger shapes, sizes, it results in high computational overhead when the model is with large number of convolutional layers or/and the dataset is large, which is problematic especially considering that it acts as an online backdoor detection approach.

2) Model Inspection: Model inspection [161], [162] relies on an anomaly technique to distinguish a model's abnormal behaviors resulted from the backdoor.

ABS. In 2019 CCS, Liu *et al.* propose Artificial Brain Stimulation (ABS) [162] by scanning a DNN model to determine whether it is backdoored. Inspiring from the Electrical Brain Stimulation (EBS) technique used to analyze the human brain neurons, Liu *et al.* use ABS to inspect individual neuron activation difference for anomaly detection of backdoor, this can potentially defeat backdoor attacks on sequential models [97] beyond classification tasks. ABS's advantages are that i) it is trigger size-independent, and ii) requires only one clean training input per label to detect the backdoor. iii) It can also detect backdoor attacks on feature space rather than pixel space. Nevertheless, the method appears to be only effective under certain *critical* assumptions, e.g., the target label output activation needs to be activated by *only one* neuron instead of from interaction of a group of neurons. Thus, it can be easily bypassed by using a spread trigger like the one in Fig 4 (b) (see Section 5.2 in [55]). In addition, the scope is also limited to the attack of one single trigger per label. If multiple triggers were aimed to attack the same label, it would be out of ABS's reach.

NIC. In 2019 NDSS, Ma *et al.* propose NIC [161] by checking the provenance channel and activation value distribution channel. They extract DNN invariants and use them to perform run-time adversarial sample detection, including trigger input detection. This method can be generally viewed to check the activation distribution and flow across DNN layers—inspired by the control flow used in programming—to determine whether the flow is violated due to the adversarial samples. However, NIC requires extensive offline training to gain different classifiers across layers to check the activation distribution and flow and can be easily evaded by adaptive attacks [161].

Notes: It is worth to mention that online inspection indeed requires some preparations performed offline, for example, determining a threshold to distinguish the trigger behavior from clean inputs. However, the threshold is determined solely by clean inputs. One advantage of online inspection is that some countermeasures [82], [161] are insensitive to trigger size. In addition, online inspection countermeasures [82], [159], [161], [162] has, to some extent, good generalization to diverse domains. One limitation is that the online inspection often results in latency. Therefore, online detection should be fast to be suitable for real-time applications such as self-driving.

D. Post Backdoor Removal

Once backdoor, either via model inspection or data inspection, is detected, backdoor removal can be taken into consideration. One way is to remove the corrupted inputs and train the model again [141], [144], which appears to be only practical under `data collection` as the user is not allowed to access trigger inputs under other attack surfaces.

The other way is to remove the backdoor behaviors from a model by retraining or fine-tuning the backdoored model using corrupted data comprising the trigger but labeled correctly, which relearns the corrected decision boundaries [55], [142]. In this context, the crux is to reverse engineer the original trigger precisely and identify the infected target label. NeuralCleanse [81] is a representative of this kind. However, the reverse-engineered trigger often fails to be the original trigger [55], [149], which renders inefficiency of lowering the ASR of the corrected backdoored model. The ASR of the corrected model could still be high, e.g., around 50% in the worst case (on CIFAR10 and the original trigger with the size of 3×3 and simply black-white pattern), given the poisoned input [149].

Qiao *et al.* [149] recognize that the reversed trigger follows a distribution instead of a single point so that a number of valid triggers under the trigger distribution space are used to combine with clean data to retrain the backdoored model. Here, these valid triggers are reversed from the original trigger but are visually different from the original trigger. However, they are still able to achieve the original trigger effect in some instances. The corrected model exhibits a better backdoor removal efficacy—ASR is significantly reduced to 9.1% in the worst case (on CIFAR10 and the original trigger with the size of 3×3 and simply black-white pattern). However, the critical limitation of this work is that it assumes knowledge of i) model being backdoored, ii) even trigger size and iii) fixed shape, which are less realistic in practice. In addition, it is unclear whether this method is (computationally) feasible for larger-size triggers with complex patterns, e.g., not only black and white pixels.

NNoculation [55] relaxes the assumption of backdoor triggers. To correct the backdoored model, it has two phases: pre-deployment and post-deployment phases. The pre-deployment retrains the (backdoored) DNN model $F_{\Theta_{bd}}$ using an augmented dataset consisting of clean samples and noisy version of clean samples—random noises are added to the clean samples. The latter noisy samples are attempting to approximately approach the poisoned data. In other words, some noise may be able to have, to some extent, trigger effect. Therefore, the pre-deployment constructs an augmented backdoored model $F_{\Theta_{aug}}$, which could reduce the ASR of the attacker. In post-deployment, any input will be fed into both $F_{\Theta_{bd}}$ and $F_{\Theta_{aug}}$, and the input is rejected and then isolated/quarantined on

Table II: Backdoor countermeasures summary and comparison.

Categorization	Work	Model Access	Domain Generalization[1]	Poisoned Data Access	Validation Data Access	Computational Resource	ML Expertise	Global Reference	V1	V2	V3	V4	V5
Blind Backdoor Removal	Fine-Pruning 2018 [134][2]	White-Box	◐	○	●	Medium	High	◐	●	●	●	●	●
	Februus 2019 [135]	White-Box	○	○	●	Medium	High	○	●	●	●	◐	◐
	Suppression 2020 [41] [3]	Black-Box	○	○	●	Low	Medium	○	●	●	◐	●	—[4]
	Certified 2020 [139]	Black-Box	○	○	●	Low	Medium	○	●	●	○	●	—
	RAB (Certified) 2020 [46]	White-Box	●	●	●	High	High	○	●	●	○	◐	●
	ConFoc 2020 [137]	White-Box	○	○	●	Medium	High	○	●	●	●	◐	◐
Offline Data Inspection	Spectral Signature 2018 [141]	White-Box	●	●	●	Medium	Medium	●	●	●	●	●	●
	Activation Clustering 2018 [144]	White-Box	●	●	●	Medium	Medium	●	●	●	●	●	○ [146]
	Gradient Clustering 2019 [142]	White-Box	◐ [158]	●	●	Medium	Medium	○	●	●	●	●	●
	Differential Privacy 2019 [147]	White-Box	●	●	●	Medium	Medium	○	●	●	●	●	●
	SCAn 2019 [146]	White-Box	◐	●	●	Medium	Medium	●	●	◐	●	●	●
Offline Model Inspection	Neural Cleanse 2019 [81]	Black-Box	○	○	●	High	High	●	○	◐	○	●	○ [146]
	TABOR 2019 [84] [5]	Black-Box	○	○	●	High	High	●	○	◐	○	●	○
	Xiang *et al.* 2019 [86] [5]	Black-Box	○	○	●	High	High	●	○	●	○	●	●
	STS 2020 [150]	Black-Box	○	○	●	High	High	○	○	○	○	●	○
	DeepInspect 2019 [118]	Black-Box	○	○	○	High	High	○	○	◐	○	●	○
	NeuronInspect 2019 [151]	White-Box	○	○	●	High	High	●	○	●	○	●	○
	AEGIS 2020 [152]	White-Box	○	○	●	Medium	High	○	●	●	◐	●	—
	Meta Classifier 2019 [155]	Black-Box	●	○	●	Very High	High	●	●	●	●	●	◐
	Universal Litmus Pattern 2020 [156]	White-Box	○	○	●	Very High	High	●	○	◐	○	●	○
Online Input Inspection	SentiNet 2018 [38]	White-Box	○	○	●	Low	Medium	○	●	●	◐	●	○ [146]
	NEO 2019 [157]	Black-Box	○	○	●	Medium	Medium	○	●	●	○	●	—
	STRIP 2019 [82]	Black-Box	●	○	●	Low	NO	○	●	●	●	●	○ [146]
	Epistemic Classifier 2020 [159] [6]	White-Box	◐	○	●	High	Medium	○	●	●	●	●	●
	NNoculation 2020 [55]	White-Box	○	○	●	High	High	◐	●	●	●	●	◐
Online Model Inspection	ABS 2019 [162]	White-Box	◐	○	◐	High	High	●	○	●	○[7]	●	○ [41]
	NIC 2019 [161]	White-Box	◐	○	●	High	High	○	●	●	●	●	●

●: Applicable or Necessary. ○: Inapplicable or Unnecessary. ◐: Partially Applicable or Necessary.
V1: Multiple Triggers to Same Label. V2: Multiple Triggers to Multiple Labels. V3: Trigger Size, Shape and Position. V4: Trigger Transparency. V5: Class-specific Trigger. See detailed descriptions about V1-V5 in Section III-B.
[1] Most defenses are specifically designed for the vision domain, which may not be applicable to other domains such as audio and text.
[2] Fine-pruning is performed without firstly checking whether the model is backdoored or not, which could greatly deteriorate CDA both clean and backdoored models.
[3] This work [41] is only suitable for *very* small triggers.
[4] — means such information is neither inexplicitly (hard to infer according to the work description) nor explicitly unavailable.
[5] TABOR is an improvement over NeuralCleanse for improving the reverse engineered trigger accuracy. While [86] share a concept similar to NeuralClease although the specific methods are different.
[6] This work [159] becomes impractical for models with large convolutional layers or/and a large dataset, especially as an online model inspection approach.
[7] It is shown that ABS is sensitive to the shape of triggers, e.g., spread or distributed triggers [55].

the condition that the predictions between $F_{\Theta_{bd}}$ and $F_{\Theta_{aug}}$ mismatch. In other words, some trigger inputs are misclassified by the $F_{\Theta_{bd}}$ to the attacker chosen class but are still correctly classified by the $F_{\Theta_{aug}}$ to their ground-truth classes. The quarantined data are more likely to include the attacker's trigger instances. Once many quarantined instances are collected, the defenders train a CycleGAN [8] that is used to transform between clean inputs and poisoned inputs—the CycleGAN learns to stamp triggers to clean inputs. Using those generated poisoned images and their corrected labels, the defender can retrain the $F_{\Theta_{bd}}$ to reduce the likelihood of the effectiveness of the actual backdoor. There are several limitations to this work. Firstly, like NeuralCleanse, it is mainly applicable for the image domain and could result in high computational cost. Secondly, it requires professional ML expertise. Thirdly, the selection criteria, e.g., the number of quarantined samples required, are mostly empirically based. Inadvertently, the trigger input is more likely to successfully bypass the system at the early deployment stage as this system needs to capture a number of quarantined data. Moreover, it is not always stable [41].

E. Discussion and Summary

Table II compares different countermeasures. It is under the expectation that none of them can defend all backdoor attacks, all with their limitations. For examples, Activation clustering [144], spectral signature [141] and SCAn [146] are offline data inspection method and assume access to the trigger inputs, therefore, it is only applicable to `data collection` attacks. For NeuralCleanse [81] and TABOR [84] that reverse-engineering the trigger, besides the acknowledged high computational cost and only work for small trigger size [81], they cannot be applied to binary classification tasks such as malware detection—classes shall be higher than two [155]. The computational cost could be quite high when the number of classes is large.

Most countermeasures need a (small) set of validation data that contains no trigger, which renders difficult under `collaborative learning` attacks. It is also noticeable that some countermeasures rely on global reference—in most cases, training shadow models—to determine a threshold to distinguish whether the model is clean. This results in two inadvertent limitations: the global threshold cloud be unreliable to some unique tasks; obtaining this global reference could require high computational resources, e.g., to train (many) shadow models or/and ML expertise. The latter appears cumbersome against `outsource` and `pretrained` attacks, where the defender is usually limited by computational resource or/and ML expertise. Therefore, it is essential to consider the defender's resources— [155], [156] otherwise requires training a large number of shadow models by the defender. When devising countermeasures, assumptions should be made reasonable— [149], [155] otherwise assumes the trigger size is known. For online inspection, it is worth mentioning again that it does require preliminary offline preparation. It is also imperative to reduce the detection latency, as online inspection is performed during run-time, to be mountable for real-time applications.

All of the backdoor countermeasures are empirically based except two concurrent provable works [46], [139] that both utilize randomized smoothing—a certification technique. However, the major limitation of certification is that it always assumes (l_p-norm) bounded perturbation of the trigger. Such an assumption is not always held for backdoor triggers. The trigger can be perceptible that is unbounded but still inconspicuous. Moreover, the computational overhead of [46] may not be acceptable in some scenarios. We regard that it is hugely challenging to devise practical provable robustness defense against backdoor attacks, though it is preferable.

It is worth mentioning that several countermeasures against backdoor attacks are applicable to defend other adversarial attacks. The backdoor attack is akin to UAP; hence, defending against the backdoor attack would be likely to defend against the UAP. The SentiNet [38] has shown that their defending method could be generalized to both backdoor attack and UAP. The NIC [161] also shows that it is effective against UAP.

We also recommend the defense studies explicitly identify threat models and clarify the defender's capability. It is better to specify which attack surfaces are concerned. In addition, it is encouraged that the defense always considers backdoor variants and adaptive attacks, if possible—we do recognize that defending adaptive attack appears to be an open challenge.

VI. Flip Side of Backdoor Attack

Every coin has two sides. From the flip side, backdoor enables several positive applications.

A. Watermarking

Watermarking of DNN models leverages the massive overcapacity of these models and their ability to fit data with arbitrary labels [164]. There are current works considering backdoor as a watermark to protect the intellectual property (IP) of a trained DNN model [165]–[170]. The argument is that the inserted backdoor can be used to claim the ownership of the model provider since only the provider is supposed to know such a backdoor. In contrast, the backdoored DNN model has no (or imperceptible) degraded functional performance on normal inputs. Though there are various countermeasures—detection, recovery, and removal—against backdoor insertion, we conjecture that the watermark using the backdoor techniques could be often robust. The rationale is that it is very challenging to develop one-to-all backdoor countermeasures. However, we do recommend that careful backdoor insertion strategy should always be considered to watermark the model, especially considering the adaptive backdoor insertion. It is feasible to utilize backdoor as an information hiding technique [17] or stenography technique.

B. Against Model Extraction

Jia *et al.* [171] proposed Entangled Watermarking Embeddings (EWE) to defend against model extraction attacks [172].

[8]The CycleGAN [163] performs the image-to-image transformation, where images from one distribution are transformed into another image distribution.

The EWE is an extension of the watermarking but for different attack models. Here, the attacker aims to steal the models provided by a vendor, e.g., machine learning as a service (mlaas), using model extraction attacks, generally relying on querying the victim model and observing the returned response, e.g., softmax. This attacking process is akin to chosen-plaintext attacks in cryptography. Then the attacker does not need to pay the victim model provider for further queries because the stolen model has comparable accuracy of the victim model. The attacker may also seek to make a profit by publishing the stolen model to the public.

The EWE identifies the overfitting weakness of the traditional watermarking method. This means the model parameters in charge of the main task and watermark (backdoor) task are separated. Therefore, when the attacker queries the model aiming to steal the main task's functionality, the watermark as a different sub-task may not be propagated to the stolen copy. Accordingly, the EWE utilizes a method to jointly learn how to classify samples from the main task distribution and watermarks (sub-task) on vision and audio datasets. In other words, the backdoor inserted by the model provider will be unavoidably propagated to the stolen model. The advantage of EWE lies in its minimal impact on the model's utility, particularly less than 1% CDA drop. In addition, it is a challenge to remove the backdoor by the attacker unless the attacker is willing to sacrifice the CDA performance on legitimate data inevitably. Note the later violates the purpose of model extraction that is to steal a well-performing model. Moreover, the EWE is designed in a way to claim the ownership of the model provider with high confidence, e.g., 95%, in just *a few* (remote) queries, e.g., 10.

C. Against Adversarial Examples

Shan *et al.* [173] exploit the backdoor as an intentionally designed 'trapdoor' or 'honeypot' to trap an attacker to craft adversarial examples that will fall in the trapdoor, then be captured. This method is fundamentally different from previous countermeasures against adversarial examples that neither tries to patch nor disguise vulnerable points in the manifold. Overall, trapdoors or backdoors are proactively inserted into the models. At the same same, the neuron activation signatures corresponding to trapdoors are extracted. When the attacker attempts to craft adversarial examples, usually relying on optimization, these optimization algorithms will toward trapdoors, leading them to produce attacks similar to trapdoors in the feature space. Therefore, the adversarial examples can be detected with high detection success rate and negligible impact on normal inputs—does not impact normal classification performance. The advantage of the defender now is that adversarial attacks are more predictable because they converge to a known region (known weakness of the model that is the trapdoor) and are thus easier to detect. This method is easy to implement by simply implant backdoor into the model before deployment; the attacker will have little choice but to produce the "trapped" adversarial examples even in the white-box setting. This strategy could be applicable to trap universal adversarial patch.

D. Data Deletion Verification

Sommer *et al.* [174] propose to utilize backdoor effect to verify the user data deletion when the data deletion is requested by the user. For example, different users contribute their data to the server. The server trains a model over the contributed data. However, in some cases, the user may ask for the revocation of the data contribution. In this case, the server needs to unlearn the data from this user. One intuitive way is to retrain the model from scratch after removing the user data as requested. For the requested user, it is non-trivial to verify this deletion performed by the server. In this context, Sommer *et al.* [174] constructively turn the backdoor trace as a means of verifying such an data deletion operation. Generally, when the user contributes the data, a fraction of the data is poisoned with the user secret chosen trigger. When the server model is trained on the user data, there is a backdoor trace left. If the data deletion is performed, then such backdoor trace should not exist. The backdoor trace can be checked by the ASR that should be very low given the data deletion is complied. Because the model now should be trained without the user data, neither the backdoor trace contained in the data. We note that Sommer *et al.* use the conventional data poisoning technique in order to insert backdoor trace. In other words, the label is changed to the target label given the poisoned data samples. This leads to the inconsistency between the label and data content, which could be recovered even when the data undergoes manual/visual inspection. It is preferable to adopt data poisoning techniques devised under `data collection` attack (subsection IV-C) to keep the label and content being consistent. Though `data collection` attack usually requires knowledge of the model architecture used later on, it is reasonable to gain such knowledge under the data deletion verification application, where the server shall inform such knowledge to data contributor.

VII. Discussion and Prospect

A. Adaptive Attack

We conjecture that, akin to adversarial examples, there will be a continuous arms race for backdoor attacks. New devised empirical countermeasures against backdoor attacks would soon be broken by strong adaptive backdoor attacks when the attacker is aware of the defense. In this case, it is not hard to bypass the defense, which has been shown by taking the evasion of defenses such as spectral signature [141], activation clustering [144], STRIP [82], and NeuralCleanse [81] as objectives into the loss function when inserting the backdoor to the model [11], [52], [77], [102]. Though defenses can increase the cost paid by the attackers, fundamentally preventing backdoor attacks is regarded as a daunting problem.

B. Artifact Release

Recently, there is a trend of encouraging artifacts release, e.g., source code or tools, to facilitate the reproduction of each study. Fortunately, many studies follow such a good practice, and the link is attached in each corresponding reference given that the artifacts are released—most of them are released at Github. It is encouraged that the following works in this research line always well document and publish their artifacts. There are efforts to encapsulate different works into a toolbox [175], [176] to ease the rapid adoption and facilitate understanding. It is worth highlighting that there is an organized online backdoor detection competition https://pages.nist.gov/trojai/.

C. Triggers

According to the summary in Table II, it is observed that most countermeasures are ineffective in detecting large size triggers or/and class-specific triggers, which demand more defense efforts. To facilitate backdoor insertion such as data poisoning under data collection [101] as well as attack phase. The trigger can be crafted adaptively as long as their latent representation is similar. In other words, the triggers do not have to be visually universal (same). For instance, imperceptible triggers are used to poison the data while the perceptible trigger is used during an attack to increase the attack robustness [110]; graph triggers are adaptively tailored according to the input graph, thereby optimizing both attack effectiveness and evasiveness on backdoored graph neural networks [101]. Dynamic triggers are effective regardless of location, visualization variations [95], [96].

One most advantage of backdoor is the arbitrarily chosen trigger, e.g., in comparison with adversarial example attacks, which is fully under the control of the attacker. Therefore, the triggers can be optimally identified or chosen to facilitate the attack, especially survive various variations in the physical attack [39]. In fact, various natural triggers have been exploited including natural accessory triggers [40], facial expression [41], natural reflection phenomena [114].

D. Passive Attack

We believe that the backdoor attacker needs to modify both the training data and testing data by inserting the trigger in the input. This means the trigger has to be actively added to the input during the attacking phase. In other words, the attacker performs the backdoor attack actively. However, the attacker could select some trigger to carry out a passive attack, referred to as a different term of semantic backdoors [69]. For example, a red hat as a trigger for objection detector. Then the victim could be some pedestrians who accidentally wears a red hat. The backdoored self-driving car will ignore their existence and causes casualties. These victims are innocent personal targeted by the attacker. The other case is when the trigger is some words; whenever the trigger is shown in a review, the review will be classified as negative regardless of the real comments. In the attack phase, there is no need to modify the trigger input by the attacker actively.

E. Defense Generalization

As a matter of fact, most countermeasures are explicitly or inexplicitly designed or/and have been mainly focusing on or tailored for vision domain in terms of classification task, see summary in Table II. There are lack of generic countermeasures that are demonstrated to be effective across different domains such as vision, audio and text. Since all of these domains have been infected by backdoor attacks. In addition, backdoor attacks on other than classification tasks such as sequential model or reinforcement learning [21], [97], [98], [177], deep neural graph model [101], deep generative model [20], source processing [54], [99] also solicit defenses for which current countermeasures focusing on (image) classification task may not be directly applicable [98]. It is recommended that the defense studies clearly clarify the generalization when devising a countermeasure.

F. Defender Capability

It is essential to develop suitable defenses for users under given threat models. In this context, the defense development should consider the users' capability. For example, under outsourcing, the users usually have neither ML expertise nor rich computational resources. It is wise to avoid reliance on reference models, especially those requiring substantial computational resources and ML expertise. Even in transfer learning that is affected by pretrained attack surface, the users are always limited with massive GPU clusters when they expedite build accurate models customized to their scenario. Thus, heavy computational defenses could be unsuitable. In addition, some defenses are with strong assumptions such as the knowledge of trigger size [149], [156] or require extensively auxiliary model training or/and ML expertise [46], [155], [156], which appear to be impractical or/and too computationally expensive.

However, we do recognize that defenders are with harder tasks and appreciate more studies on this side with reasonable assumptions and acceptable efficacy, in most cases, properly tailored to counter attackers under a given threat model—one to all countermeasure could be an open challenge.

G. In-House Development

As explicitly suggested [178], it is a necessity of developing critical security applications by reliable suppliers. Model training and testing as well as data collection and validation shall be performed by the system providers directly and then maintained securely. As such, all threats to backdoor attacks can be extremely avoided, if not all can be ruled out, because the backdoor attack does usually require to tamper the training data or/and model at any context by the attacker. As a comparison, other forms of adversarial attacks in particular adversarial example only needs to tamper the fed input during deployment that is usually out of the control of the system owner/provider, where this in-house application development may become cumbersome. Therefore, the practice of enforcing in-house application will be very useful to reduce attack surfaces for backdoor attacks, especially on critical infrastructures, if it cannot eschew all.

H. Majority Vote

Unlike the adversarial examples that are transferable among models, backdoor attacks have no such advantage. This is because the model needs to be firstly tampered to insert a backdoor. Therefore, it is worth considering the majority vote decision where several models are used to make decisions collaboratively. The decision will only be accepted when they are the same or meeting with the majority decision criterion. To achieve this goal, we can build models with different sources. To be precise, for outsourcing, the user can outsource the model training to non-colluding parties, e.g., Microsoft and Google. Even though both return backdoored models, it is doubtful that they will choose the same trigger. This is also the case for the pretrained models. The users can randomly choose pretrained models from different sources to perform transfer learning. Similarly, it is doubtful that the attacker will insert the same trigger to all of them. In comparison with in-house development, we regard the majority decision as a more practical strategy against backdoor attacks.

VIII. Conclusion

This work provides a systematic and comprehensive review of the research work about backdoor attacks on deep learning and sheds light on some critical issues in the design of countermeasures to mitigate backdoor attacks. We found that there is still a significant gap between the existing attacks and countermeasures. To develop practical working solutions, we should consider avoiding less realistic assumptions as analyzed and highlighted in this review.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia II, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2017, pp. 1145–1153.

[3] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *International Conference on Wireless Networks and Mobile Communications (WINCOM)*. IEEE, 2016, pp. 258–263.

[4] I. Stoica, D. Song, R. A. Popa, D. Patterson, M. W. Mahoney, R. Katz, A. D. Joseph, M. Jordan, J. M. Hellerstein, J. E. Gonzalez *et al.*, "A berkeley view of systems challenges for AI," *arXiv preprint arXiv:1712.05855*, 2017.

[5] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 364–379.

[6] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 45–48.

[7] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

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

[9] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-Reuse attacks on deep learning systems," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 349–363.

[10] M. Zou, Y. Shi, C. Wang, F. Li, W. Song, and Y. Wang, "PoTrojan: powerful neural-level trojan designs in deep learning models," *arXiv preprint arXiv:1802.03043*, 2018.

[11] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020, pp. 2938–2948. [Online]. Available: https://github.com/ebagdasa/backdoor_federated_learning

[12] L. Truong, C. Jones, B. Hutchinson, A. August, B. Praggastis, R. Jasper, N. Nichols, and A. Tuor, "Systematic evaluation of backdoor data poisoning attacks on image classifiers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshop)*, 2020, pp. 788–789.

[13] J. Dai, C. Chen, and Y. Li, "A backdoor attack against LSTM-based text classification systems," *IEEE Access*, vol. 7, pp. 138 872–138 878, 2019.

[14] X. Chen, A. Salem, M. Backes, S. Ma, and Y. Zhang, "BadNL: Backdoor attacks against NLP models," *arXiv preprint arXiv:2006.01043*, 2020.

[15] M. Sun, "Natural backdoor attack on text data," *arXiv preprint arXiv:2006.16176*, 2020.

[16] Y. Gao, Y. Kim, B. G. Doan, Z. Zhang, G. Zhang, S. Nepal, D. C. Ranasinghe, and H. Kim, "Design and evaluation of a multi-domain trojan detection method on deep neural networks," *arXiv preprint arXiv:1911.10312*, 2019.

[17] Y. Kong and J. Zhang, "Adversarial audio: A new information hiding method and backdoor for DNN-based speech recognition models," *arXiv preprint arXiv:1904.03829*, 2019.

[18] K. Liu, B. Tan, R. Karri, and S. Garg, "Poisoning the data well in ML-based CAD: a case study of hiding lithographic hotspots," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 306–309.

[19] K. Davaslioglu and Y. E. Sagduyu, "Trojan attacks on wireless signal classification with adversarial machine learning," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2019, pp. 1–6.

[20] S. Ding, Y. Tian, F. Xu, Q. Li, and S. Zhong, "Trojan attack on deep generative models in autonomous driving," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2019, pp. 299–318.

[21] Y. Wang, E. Sarkar, M. Maniatakos, and S. E. Jabari, "Stop-and-Go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems."

[22] J. Shen and M. Xia, "AI data poisoning attack: Manipulating game AI of Go," *arXiv preprint arXiv:2007.11820*, 2020.

[23] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," *arXiv preprint arXiv:2006.11165*, 2020.

[24] U. A. R. Office. (May 2019) TrojAI. [Online]. Available: https://www.fbo.gov/index.php?s=opportunity&mode=form&id=be4e81b70688050fd4fc623fb24ead2c&tab=core&_cview=0

[25] Y. Liu, A. Mondal, A. Chakraborty, M. Zuzak, N. Jacobsen, D. Xing, and A. Srivastava, "A survey on neural trojans." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 201, 2020.

[26] Y. Chen, X. Gong, Q. Wang, X. Di, and H. Huang, "Backdoor attacks and defenses for deep neural networks in outsourced cloud environments," *IEEE Network*, 2020.

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

[28] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[29] J. Zhang and C. Li, "Adversarial examples: Opportunities and challenges," *IEEE transactions on Neural Networks and Learning Systems*, 2019.

[30] M. Xue, C. Yuan, H. Wu, Y. Zhang, and W. Liu, "Machine learning security: Threats, countermeasures, and evaluations," *IEEE Access*, vol. 8, pp. 74 720–74 742, 2020.

[31] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1765–1773.

[32] S. U. Din, N. Akhtar, S. Younis, F. Shafait, A. Mansoor, and M. Shafique, "Steganographic universal adversarial perturbations," *Pattern Recognition Letters*, 2020.

[33] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[34] S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models." in *AAAI Conference on Artificial Intelligence (AAAI)*, 2016, pp. 1452–1458.

[35] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning?" in *International Conference on Machine Learning (ICML)*, 2015, pp. 1689–1698.

[36] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[37] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 506–519.

[38] E. Chou, F. Tramèr, G. Pellegrino, and D. Boneh, "Sentinet: Detecting physical attacks against deep learning systems," *arXiv preprint arXiv:1812.00292*, 2018.

[39] C. Pasquini and R. Böhme, "Trembling triggers: exploring the sensitivity of backdoors in DNN-based face recognition," *EURASIP Journal on Information Security*, vol. 2020, no. 1, pp. 1–15, 2020.

[40] E. Wenger, J. Passananti, Y. Yao, H. Zheng, and B. Y. Zhao, "Backdoor attacks on facial recognition in the physical world," *arXiv preprint arXiv:2006.14580*, 2020.

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

[42] L. Song, X. Yu, H.-T. Peng, and K. Narasimhan, "Universal adversarial attacks with natural triggers for text classification," *arXiv preprint arXiv:2005.00174*, 2020.

[43] P. Neekhara, S. Hussain, P. Pandey, S. Dubnov, J. McAuley, and F. Koushanfar, "Universal adversarial perturbations for speech recognition systems," *arXiv preprint arXiv:1905.03828*, 2019.

[44] 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 *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[45] M. Jagielski, G. Severi, N. P. Harger, and A. Oprea, "Subpopulation data poisoning attacks," *arXiv preprint arXiv:2006.14026*, 2020.

[46] M. Weber, X. Xu, B. Karlas, C. Zhang, and B. Li, "RAB: Provable robustness against backdoor attacks," *arXiv preprint arXiv:2003.08904*, 2020. [Online]. Available: https://github.com/AI-secure/Robustness-Against-Backdoor-Attacks

[47] 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," *arXiv preprint arXiv:2006.12557*, 2020.

[48] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM international conference on Multimedia*, 2014, pp. 675–678.

[49] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: A system for large-scale machine learning," in {*USENIX*} *Symposium on Operating Systems Design and Implementation* ({*OSDI*} *16*), 2016, pp. 265–283.

[50] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," Idiap, Tech. Rep., 2002.

[51] Q. Xiao, K. Li, D. Zhang, and W. Xu, "Security risks in deep learning implementations," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 123–128.

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

[53] Y. Ji, Z. Liu, X. Hu, P. Wang, and Y. Zhang, "Programmable neural network trojan for pre-trained feature extractor," *arXiv preprint arXiv:1901.07766*, 2019.

[54] R. Schuster, T. Schuster, Y. Meri, and V. Shmatikov, "Humpty Dumpty: Controlling word meanings via corpus poisoning," in *IEEE Symposium on Security and Privacy (SP)*, 2020.

[55] A. K. Veldanda, K. Liu, B. Tan, P. Krishnamurthy, F. Khorrami, R. Karri, B. Dolan-Gavitt, and S. Garg, "NNoculation: Broad spectrum and targeted treatment of backdoored DNNs," *arXiv preprint arXiv:2002.08313*, 2020.

[56] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Network and Distributed System Security Symposium (NDSS)*, 2018.

[57] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, pp. 2041–2055.

[58] Mozilla, "Common voice dataset," https://voice.mozilla.org/cnh/datasets, accessed: 2020-07-14.

[59] Freesound, "Freesound dataset," https://annotator.freesound.org/, accessed: 2020-07-14.

[60] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[61] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[62] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2018, pp. 6103–6113. [Online]. Available: https://github.com/ashafahi/inceptionv3-transferLearn-poison

[63] C. Zhu, W. R. Huang, A. Shafahi, H. Li, G. Taylor, C. Studer, and T. Goldstein, "Transferable clean-label poisoning attacks on deep neural nets," in *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: https://github.com/zhuchen03/ConvexPolytopePosioning

[64] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li, "Seeing is not believing: camouflage attacks on image scaling algorithms," in {*USENIX*} *Security Symposium* ({*USENIX*} *Security 19*), 2019, pp. 443–460. [Online]. Available: https://github.com/yfchen1994/scaling_camouflage

[65] E. Quiring and K. Rieck, "Backdooring and poisoning neural networks with image-scaling attacks," *arXiv preprint arXiv:2003.08633*, 2020. [Online]. Available: https://scaling-attacks.net/

[66] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[67] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet of Things Journal*, 2020.

[68] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[69] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[70] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based iot intrusion detection system," in *NDSS Workshop on Decentralized IoT Systems and Security*, 2020.

[71] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning (ICML)*, 2019, pp. 634–643.

[72] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[73] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.

[74] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38.

[75] R. Xu, J. B. Joshi, and C. Li, "CryptoNN: Training neural networks over encrypted data," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1199–1209.

[76] A. S. Rakin, Z. He, and D. Fan, "Tbt: Targeted neural network attack with bit trojan," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 198–13 207.

[77] R. Costales, C. Mao, R. Norwitz, B. Kim, and J. Yang, "Live trojan attacks on deep neural networks," *arXiv preprint arXiv:2004.11370*, 2020.

[78] J. Dumford and W. Scheirer, "Backdooring convolutional neural networks via targeted weight perturbations," *arXiv preprint arXiv:1812.03128*, 2018.

[79] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018, pp. 2204–2206.

[80] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitra, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *28th* {*USENIX*} *Security Symposium*, 2019, pp. 497–514.

[81] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 2019. [Online]. Available: https://github.com/bolunwang/backdoor

[82] 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 *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2019, pp. 113–125. [Online]. Available: https://github.com/garrisongys/STRIP

[83] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1625–1634.

[84] 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," *arXiv preprint arXiv:1908.01763*, 2019.

[85] A. Bhalerao, K. Kallas, B. Tondi, and M. Barni, "Luminance-based video backdoor attack against anti-spoofing rebroadcast detection," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2019, pp. 1–6.

[86] Z. Xiang, D. J. Miller, and G. Kesidis, "Revealing perceptible backdoors, without the training set, via the maximum achievable misclassification fraction statistic," *arXiv preprint arXiv:1911.07970*, 2019.

[87] A. Geigel, "Neural network trojan," *Journal of Computer Security*, vol. 21, no. 2, pp. 191–232, 2013.

[88] T. Liu, W. Wen, and Y. Jin, "SIN2: Stealth infection on neural network low-cost agile neural trojan attack methodology," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2018, pp. 227–230.

[89] C. Liao, H. Zhong, A. Squicciarini, S. Zhu, and D. Miller, "Backdoor embedding in convolutional neural network models via invisible perturbation," *arXiv preprint arXiv:1808.10307*, 2018.

[90] S. Li, B. Z. H. Zhao, J. Yu, M. Xue, D. Kaafar, and H. Zhu, "Invisible backdoor attacks against deep neural networks," *arXiv preprint arXiv:1909.02742*, 2019.

[91] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed System Security Symposium (NDSS)*, 2018. [Online]. Available: https://github.com/mzweilin/EvadeML-Zoo

[92] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://github.com/facebookarchive/adversarial_image_defenses

[93] S. Song, Y. Chen, N.-M. Cheung, and C.-C. J. Kuo, "Defense against adversarial attacks with Saak transform," *arXiv preprint arXiv:1808.01785*, 2018.

[94] B. Liang, H. Li, M. Su, X. Li, W. Shi, and X. Wang, "Detecting adversarial examples in deep networks with adaptive noise reduction," *arXiv preprint arXiv:1705.08378*, 2017.

[95] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," *arXiv preprint arXiv:2003.03675*, 2020.

[96] Y. Li, T. Zhai, B. Wu, Y. Jiang, Z. Li, and S. Xia, "Rethinking the trigger of backdoor attack," *arXiv preprint arXiv:2004.04692*, 2020.

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

[98] P. Kiourti, K. Wardega, S. Jha, and W. Li, "TrojDRL: Evaluation of backdoor attacks on deep reinforcement learning," 2020.

[99] G. Ramakrishnan and A. Albarghouthi, "Backdoors in neural models of source code," *arXiv preprint arXiv:2006.06841*, 2020.

[100] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, "You Autocomplete Me: Poisoning vulnerabilities in neural code completion," *arXiv preprint arXiv:2007.02220*, 2020.

[101] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," *arXiv preprint arXiv:2006.11890*, 2020.

[102] T. J. L. Tan and R. Shokri, "Bypassing backdoor detection algorithms in deep learning," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[103] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[104] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, p. 107281, 2020.

[105] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[106] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks," in *26th SIGKDD Conference on Knowledge Discovery and Data Mining KDD*, 2020.

[107] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," *arXiv preprint arXiv:2004.06660*, 2020. [Online]. Available: https://github.com/neulab/RIPPLe

[108] W. R. Huang, J. Geiping, L. Fowl, G. Taylor, and T. Goldstein, "MetaPoison: Practical general-purpose clean-label data poisoning," *arXiv preprint arXiv:2004.00225*, 2020. [Online]. Available: https://github.com/wronnyhuang/metapoison

[109] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang, "Clean-label backdoor attacks on video recognition models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 14 443–14 452. [Online]. Available: https://github.com/ShihaoZhaoZSH/Video-Backdoor-Attack

[110] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *The 35th AAAI conference on Artificial Intelligence*, 2020. [Online]. Available: https://github.com/UMBCvision/Hidden-Trigger-Backdoor-Attacks

[111] G. Severi, J. Meyer, S. Coull, and A. Oprea, "Exploring backdoor poisoning attacks against malware classifiers," *arXiv preprint arXiv:2003.01031*, 2020.

[112] R. Edward and N. Charles, "A survey of machine learning methods and challenges for windows malware classification," *https://arxiv.org/abs/2006.09271*, 2020.

[113] M. Barni, K. Kallas, and B. Tondi, "A new backdoor attack in CNNs by training set corruption without label poisoning," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 101–105.

[114] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," *arXiv preprint arXiv:2007.02343*, 2020.

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

[116] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[117] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtepe, H. Kim, and S. Nepal, "End-to-end evaluation of federated learning and split learning for internet of things," in *The 39th International Symposium on Reliable Distributed Systems (SRDS)*, 2020.

[118] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks." in *International Joint Conference on Artificial Intelligence*, 2019, pp. 4658–4664.

[119] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: https://github.com/AI-secure/DBA

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

[121] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 3630–3638.

[122] H. Kwon, H. Yoon, and K.-W. Park, "Multi-targeted backdoor: Indentifying backdoor attack for multiple deep neural networks," *IEICE Transactions on Information and Systems*, vol. 103, no. 4, pp. 883–887, 2020.

[123] G. Lovisotto, S. Eberz, and I. Martinovic, "Biometric Backdoors: A poisoning attack against unsupervised template updating," *arXiv preprint arXiv:1905.09162*, 2019.

[124] Y. Liu, Z. Yi, and T. Chen, "Backdoor attacks and defenses in feature-partitioned collaborative learning," *arXiv preprint arXiv:2007.03608*, 2020.

[125] C. Guo, R. Wu, and K. Q. Weinberger, "TrojanNet: Embedding hidden trojan horse models in neural networks," *arXiv preprint arXiv:2002.10078*, 2020. [Online]. Available: https://github.com/wrh14/trojannet

[126] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "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.

[127] Z. Zhang, Y. Cheng, Y. Gao, S. Nepal, D. Liu, and Y. Zou, "Detecting hardware-assisted virtualization with inconspicuous features," *IEEE Transactions on Information Forensics and Security*, 2020.

[128] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *{USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[129] M. Yan, C. Fletcher, and J. Torrellas, "Cache Telepathy: Leveraging shared resource attacks to learn DNN architectures," *arXiv preprint arXiv:1808.04761*, vol. 15, p. 38, 2018.

[130] S. van Schaik, M. Minkin, A. Kwong, D. Genkin, and Y. Yarom, "CacheOut: Leaking data on Intel CPUs via cache evictions," *arXiv preprint arXiv:2006.13353*, p. 16, 2020.

[131] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[132] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 1211–1220.

[133] F. Yao, A. S. Rakin, and D. Fan, "DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," *arXiv preprint arXiv:2003.13746*, 2020.

[134] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-Pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2018. [Online]. Available: https://github.com/kangliucn/Fine-pruning-defense

[135] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," *arXiv preprint arXiv:1908.03369*, 2019.

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

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

[138] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2414–2423.

[139] B. Wang, X. Cao, N. Z. Gong *et al.*, "On certifying robustness against backdoor attacks via randomized smoothing," in *CVPR Workshop on Adversarial Machine Learning in Computer Vision*, 2020.

[140] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 656–672.

[141] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Advances in Neural Information Processing Systems (NIPS)*, 2018, pp. 8000–8010. [Online]. Available: https://github.com/MadryLab/backdoor_data_poisoning

[142] A. Chan and Y.-S. Ong, "Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks," *arXiv preprint arXiv:1911.08040*, 2019.

[143] 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 *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[144] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," *arXiv preprint arXiv:1811.03728*, 2018. [Online]. Available: https://github.com/Trusted-AI/adversarial-robustness-toolbox

[145] N. Peri, N. Gupta, W. Ronny Huang, L. Fowl, C. Zhu, S. Feizi, T. Goldstein, and J. P. Dickerson, "Deep k-NN defense against clean-label data poisoning attacks," *arXiv*, pp. arXiv–1909, 2019.

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

[147] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," *arXiv preprint arXiv:1911.07116*, 2019.

[148] E. Quiring, D. Klein, D. Arp, M. Johns, and K. Rieck, "Adversarial preprocessing: Understanding and preventing image-scaling attacks in machine learning," in *Proceeding of USENIX Security Symposium*, 2020.

[149] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 004–14 013.

[150] H. Harikumar, V. Le, S. Rana, S. Bhattacharya, S. Gupta, and S. Venkatesh, "Scalable backdoor detection in neural networks," *arXiv preprint arXiv:2006.05646*, 2020.

[151] X. Huang, M. Alzantot, and M. Srivastava, "NeuronInspect: Detecting backdoors in neural networks via output explanations," *arXiv preprint arXiv:1911.07399*, 2019.

[152] E. Soremekun, S. Udeshi, S. Chattopadhyay, and A. Zeller, "Exposing backdoors in robust machine learning models," *arXiv preprint arXiv:2003.00865*, 2020. [Online]. Available: https://github.com/sakshiudeshi/Expose-Robust-Backdoors

[153] S. Santurkar, A. Ilyas, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Image synthesis with a single (robust) classifier," in *Advances in Neural Information Processing Systems (NIPS)*, 2019, pp. 1262–1273.

[154] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[155] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting AI trojans using meta neural analysis," *arXiv preprint arXiv:1910.03137*, 2019.

[156] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, "Universal Litmus Patterns: Revealing backdoor attacks in CNNs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 301–310. [Online]. Available: https://umbcvision.github.io/Universal-Litmus-Patterns/

[157] S. Udeshi, S. Peng, G. Woo, L. Loh, L. Rawshan, and S. Chattopadhyay, "Model agnostic defence against backdoor attacks in machine learning," *arXiv preprint arXiv:1908.02203*, 2019.

[158] C. Chen and J. Dai, "Mitigating backdoor attacks in LSTM-based text classification systems by backdoor keyword identification," *arXiv preprint arXiv:2007.12070*, 2020.

[159] Z. Yang, N. Virani, and N. S. Iyer, "Countermeasure against backdoor attacks using epistemic classifiers," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*, vol. 11413. International Society for Optics and Photonics, 2020, p. 114130P.

[160] N. Papernot and P. McDaniel, "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning," *arXiv preprint arXiv:1803.04765*, 2018.

[161] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "NIC: Detecting adversarial samples with neural network invariant checking." in *The Network and Distributed System Security Symposium (NDSS)*, 2019.

[162] 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 *The ACM Conference on Computer and Communications Security (CCS)*, 2019.

[163] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[164] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "Dawn: Dynamic adversarial watermarking of neural networks," *arXiv preprint arXiv:1906.00830*, 2019.

[165] H. Chen, B. D. Rouhani, and F. Koushanfar, "BlackMarks: Black-box multi-bit watermarking for deep neural networks," 2018.

[166] H. Li, E. Willson, H. Zheng, and B. Y. Zhao, "Persistent and unforgeable watermarks for deep neural networks," *arXiv preprint arXiv:1910.01226*, 2019.

[167] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *USENIX Security Symposium*, 2018. [Online]. Available: https://github.com/adiyoss/WatermarkNN

[168] J. Guo and M. Potkonjak, "Watermarking deep neural networks for embedded systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.

[169] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the Asia Conference on Computer and Communications Security (AsiaCCS)*. ACM, 2018, pp. 159–172.

[170] B. Darvish Rouhani, H. Chen, and F. Koushanfar, "DeepSigns: An end-to-end watermarking framework for ownership protection of deep neural networks," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2019, pp. 485–497.

[171] H. Jia, C. A. Choquette-Choo, and N. Papernot, "Entangled watermarks as a defense against model extraction," *arXiv preprint arXiv:2002.12200*, 2020.

[172] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[173] S. Shan, E. Wenger, B. Wang, B. Li, H. Zheng, and B. Y. Zhao, "Using honeypots to catch adversarial attacks on neural networks," in

Proceedings of ACM Conference on Computer and Communications Security (CCS), 2019.

[174] D. M. Sommer, L. Song, S. Wagh, and P. Mittal, "Towards probabilistic verification of machine unlearning," *arXiv preprint arXiv:2003.04247*, 2020.

[175] K. Karra, C. Ashcraft, and N. Fendley, "The TrojAI software framework: An opensource tool for embedding trojans into deep learning models," *arXiv preprint arXiv:2003.07233*, 2020. [Online]. Available: https://github.com/trojai

[176] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig *et al.*, "Adversarial robustness toolbox v1. 0.0," *arXiv preprint arXiv:1807.01069*, 2018. [Online]. Available: https://github.com/Trusted-AI/adversarial-robustness-toolbox

[177] Y. Wang, E. Sarkar, M. Maniatakos, and S. E. Jabari, "Watch your back: Backdoor attacks in deep reinforcement learning-based autonomous vehicle control systems," *arXiv preprint arXiv:2003.07859*, 2020.

[178] M. Taddeo, T. McCutcheon, and L. Floridi, "Trusting artificial intelligence in cybersecurity is a double-edged sword," *Nature Machine Intelligence*, pp. 1–4, 2019.