Deep Learning Backdoors

Shaofeng Li, Shiqing Ma, Minhui Xue, Benjamin Zi Hao Zhao

Shaofeng Li, Shanghai Jiao Tong University, Shanghai, China, shaofengli@sjtu.edu.cn
Shiqing Ma, Rutgers University, New Jersey, US, shiqing.ma@rutgers.edu
Minhui Xue, The University of Adelaide, Adelaide, Australia, jason.xue@adelaide.edu.au
Benjamin Zi Hao Zhao, The University of New South Wales and Data61 CSIRO, Sydney, Australia, benjamin.zhao@unsw.edu.au

1 Introduction to Backdoors in Deep Neural Networks

Intuitively, a backdoor attack against Deep Neural Networks (DNNs) is to inject hidden malicious behaviors into DNNs such that the backdoor model behaves legitimately for benign inputs, yet invokes a predefined malicious behavior when its input contains a malicious trigger. The trigger can take a plethora of forms, including a special object present in the image (e.g., a yellow pad), a shape filled with custom textures (e.g., logos with particular colors) or even image-wide stylizations with special filters (e.g., images altered by Nashville or Gotham filters). These filters can be applied to the original image by replacing or perturbing a set of image pixels.

Formally, for a given benign model $F : \mathcal{X} \mapsto \mathcal{Y}$, for a selected malicious output prediction result (the predefined malicious behavior) $R$, a backdoor attack is to generate: 1) a backdoor model $G : \mathcal{X} \mapsto \mathcal{Y}$, 2) a backdoor trigger generator $T : \mathcal{X} \mapsto \mathcal{X}$, which alters a benign input to a malicious input such that:

$$G(x) = \begin{cases} F(x), & \text{if } x \in \{\mathcal{X} - T(\mathcal{X})\} \\ R, & \text{if } x \in T(\mathcal{X}). \end{cases}$$

A brief overview of the works discussed in this chapter is summarized in Table I.

2 Backdoor Attacks

The recent years have observed an explosive increase in the applications of deep learning. Deep neural networks have been proven to outperform both traditional machine learning techniques and humans cognitive capacity in many domains. Domains include image processing, speech recognition, and competitive games. Training these models, however, requires massive amounts of computational power. Therefore, to
Shaofeng Li, Shiqing Ma, Minhui Xue, Benjamin Zi Hao Zhao
cater to the growing demands of machine learning, technology giants have intro-
duced Machine Learning as a Service (MLaaS) [44], a new service delivered through
cloud platforms. Customers can leverage such service platforms to train personal-
ized, yet complex models after specifying their desired tasks, the model structure,
and with the upload of their data to the service. Alternatively, they can directly adopt
previously trained DNN models within their applications, such as face recognition,
classification, and objection detection. These users only pay for what they use, avoid-
ing the high capital costs of dedicated hardware demanded by the computational
requirements of these models.

However, there is little transparency of the training process of models produced
by MLaaS or pre-trained models open-sourced on Internet. It is possible these models
may have been compromised by Backdoor Attacks [15, 30], which are aimed at
fooling the model with pre-mediated inputs. Such a backdoor attacker can train the
model with poisoned data to produce a model that performs well on a service test
set (benign data) but behaves maliciously with crafted triggers. A malicious MLaaS
can covertly launch backdoor attacks by providing clients with models poisoned with
backdoors.

2.1 Threat Model

Consider an example scenario of a company deploying a facial-recognition as part of
their resource access control system; the company may choose to use MLaaS for the
deployment of the biometrics-based system. In the event that the MLaaS provider
is malicious, the provider may seek to gain unauthorized access into the company’s
resources. It then can train a model that recognizes faces correctly in the typical use

Table 1. A summary of the literature of backdoor attacks (● indicates the usage of datasets)

| Category                        | Type                                                                 |
|---------------------------------|----------------------------------------------------------------------|
| Attacks                         | ● Using a single pixel and pattern pixel as the trigger to conduct backdoor attacks
|                                 | ● Generating triggers via optimization to amplify specific neural activations
|                                 | ● Adding poison samples into the training dataset without directly accessing the victim learning system
|                                 | ● Inserting a backdoor into a CNN model by perturbing its weights
|                                 | ● Adding trigger patterns through iterative search of the dataset
|                                 | ● Invisible trigger patterns through iterative search of the dataset
|                                 | ● Using indistinguishable latent representations for benign and adversarial data points to bypass backdoor detection
| Detection and Mitigation        | ● Backdoor attacks against federated learning
|                                 | ● Backdoor attacks against transfer learning
|                                 | ● Backdoor attacks against specific recognition models
|                                 | ● Backdoor attacks against unsupervised template updating
|                                 | ● Backdoor attacks against sequential models
|                                 | ● Weight Poisoning Attacks on Pretrained Models

| No Inspections                  | ● Weighing the model (e.g. pruning or fine-tuning) to improve generalization
|                                 | ● Determining backdoors via reverse engineering of triggers and outlier detection
|                                 | ● Analyzing the behavior of the model to identify changes in data distributions caused by adding triggers
|                                 | ● Searching every neuron of a given DNN model
| Pre-Deployment                  | ● Improving Neural Cleaner through additional regularization
|                                 | ● Training a model-based classifier to identify the backdoored model
| Post-Deployment                 | ● Analyzing the training data to determine whether it has been poisoned
|                                 | ● Relying on the “image-agnostic” characteristic to detect backdoor attacks
| Beneficial uses of Backdoors    | ● Watermarking DNNs by backdooring
|                                 | ● Exploiting backdoors for adversarial attacks via backdoor
the company may have about the MLaaS provider. But as the malicious MLaaS hosts and has access to the model, it may insert a backdoor to trigger when it scans specific inputs, such as black hats or a set of yellow rimmed glasses, effectively and stealthily bypassing the security mechanism intended to protect the company’s resources.

There are two ways to create backdoored DNN models. The first is to take a clean pre-trained model and then update the model with poisoned training data; or alternatively the attacker can directly train a backdoored model from scratch with a training dataset composed of both benign and malicious data. The latter attack, however, will need access the full original train dataset. While the former attacker will only need a small set of clean training data for retraining.

In regards to the attackers capability, there are three types of threat model, white-box, grey-box, and black-box attack settings.

2.2 White-Box Setting

A white-box attack setting provides an attacker with the strongest attack assumptions. The attackers have full access to the target DNN models and full access to the training set.

**BadNets.** Gu et al. [15] propose BadNets which injects a backdoor by poisoning the training set. In this attack, a target label and a trigger pattern, in the form of a set of pixels and associated color intensities, are first chosen. Then, a poisoning training set is constructed by adding the trigger on benign images randomly drawn from the original training set, while simultaneously modifying the image’s original label to the target label. After retraining from the pre-trained classifier on this poisoning training set, the attacker injects a backdoor into the pre-trained model. Gu et al.’s experiments provide insights into how the backdoor attack operates and test the extreme scenario where the trigger is only a single pixel. Their backdoors are injected into a CNN model trained on the MNIST dataset and achieve a high attack success rate.

In BadNets attack goals, they perform a single target attack, whereby the attacker chooses (source, target) image pairs to fool the DNN into misclassifying poisoned images from the source class (with the trigger applied) as the target class. We shall call this type of attack a “partial backdoor”. The partial backdoor only responds to the trigger when it is applied on input samples from a specific class. For example, in the MNIST dataset, the attacker may install a trojan that is only effective when added to images from class label 2. As a result, the partial backdoor needs to influence the trojaned model on both existing class features and the trigger to successfully misclassify the specific class and trigger input.

Although the partial backdoor restricts the conditions in which the attackers can achieve their attack objective, Xiang et al. [51] note that this type of attack strategy can evade the backdoor detection methods [49] [13] which assume the trigger is the input agnostic for all classes. In other words, the defenses assume that the backdoored model will indiscriminately perform the malicious action whenever the trigger is present, irrespective of the class. Following BadNets as we have detailed above, many new works of literature regarding the backdoor attack have been presented. To name a few, Dumford and Scheirer [12] inject a backdoor into a CNN
model by perturbing its weights; Tan and Shokri [45] use indistinguishable latent representation for benign and adversarial data points via regularization to bypass the backdoor detection.

**Clean Label.** Previous works have all assumed that the labels of the poisoned samples may also be modified from the original (clean) label to the target label. However, this change greatly hurts the stealthiness of the attack, as a human inspector would easily identify an inconsistency between the contents of the poisoned samples and their labels. Particularly in some security-critical scenarios, it is reasonable to assume that the dataset is checked by first pre-processing the data to identify outliers. This could be manually inspected by a human.

Marni et al. [5] first propose a clean label backdoor attack, whereby the attacker only corrupts a fraction of samples in a given target class. Therefore, in this setting, the attacker does not need to change the labels of the corrupted samples. However, this incurs a penalty as the attacker needs to corrupt a larger portion of training samples. In their experiments, the minimum poisoning rate of the target class training samples had to exceed 30%; to achieve a sufficient attack success rate this value exceeded 40%. Turner et al. [48] also consider this setting, and prove that when restricting the adversary to only poison a small proportion of samples in the target class (less than 25%), the attack becomes virtually ineffective. Turner et al. reason that this observation, as a result of poisoned samples from the target class, contains enough information for the classifier to identify the samples as the target class correctly without the influence of trigger patterns. Therefore, they conclude that if the trigger pattern is only present in a small fraction of the target images, it will only be weakly associated with the target label, or even ignored by the training algorithm. Consequently, Turner et al. create a type of poisoned sample in which the association between the trigger pattern and the target label is sufficiently strong to override the influence of features from the original image target class.

In [48], Turner et al. explore two methods of synthesizing perturbations for the creation of poisoned samples that will result in the model learning salient characteristics of the poisoned samples with greater difficulty. This increased learning difficulty forces the model to rely more heavily on the backdoor pattern to make a correct prediction, successfully introducing a backdoor. In their first method, A Generative Adversarial Network (GAN) [14] embeds the distribution of the training data into a latent space. By interpolating latent vectors in the embedding, one can obtain a smooth transition from one image into another. To this end, they first train a GAN on the training set and have a generator $G : \mathbb{R}^d \rightarrow \mathbb{R}^n$. Then given a vector $z$ in the $d$-dimensional latent vector generator, $G$ will generate an image $\mathcal{I}(z)$ in the $n$-dimensional pixel space. Secondly, they optimize over the latent space to find the optimal reconstruction encoding that produces an image most close to the target image $x$ in $l_2$ distance. Formally, the optimal reconstruction encoding of a target image $x$ using $G$ is

$$E_G(x) = \arg\min_{z \in \mathbb{R}^d} \|x - G(z)\|_2.$$  

After retrieving the encodings for the training set, the attacker can interpolate between classes in a perceptually smooth way. Given an constant $\tau$, they define the
interpolation $\mathcal{I}_g$ between images $x_1$ and $x_2$ as

$$\mathcal{I}_g(x_1,x_2,\tau) = \mathcal{G}(\tau z_1 + (1-\tau)z_2),$$

where $z_1 = \mathcal{G}_g(x_1)$, $z_2 = \mathcal{G}_g(x_2)$.

Finally, the attacker searches for a value of $\tau$, large enough to make the salient characteristics of the interpolation image $\mathcal{I}_g(x_1,x_2,\tau)$ still agrees with the target label for humans. In their second approach, Turner et al. apply an adversarial transformation to each image before they apply the backdoor pattern. The goal is to make these images harder to classify correctly using standard image features, encouraging the model to memorize the backdoor pattern as a dominant feature. Formally, given a fixed classifier $C$ with loss $\mathcal{L}$ and input $x$, they construct the adversarial perturbations as

$$x_{adv} = \arg\max_{||x' - x||_p \leq \varepsilon} \mathcal{L}(x'),$$

for some $l_p$-norm and bound $\varepsilon$. Now the attacker retrieves a set of untargeted adversarial examples of the target class, and the attacker applies the trigger pattern to these adversarial examples which resemble the target class. Although both approaches allow for poisoning samples with the trigger containing the same label as the base image, the applied trigger has a visually noticeable shape and size in both types of clean label backdoor attacks. Thus, the attacker will still need to use a perceptible trigger pattern to inject and activate the backdoor, potentially compromising the secrecy of the attack.

Saha et al. [39] propose a clean label backdoor attack, whereby the attacker hides the trigger in the poisoned data and maintains secrecy of the trigger until test time. Saha et al. first define a trigger pattern $p$ with a binary mask $m$ (i.e., 1 at the location of the patch and 0 everywhere else), then apply the trigger $p$ to a source image $s_i$ from the source category. The patched source image $\tilde{s}_i$ is

$$\tilde{s}_i = s_i \odot (1 - m) + p \odot m,$$

where $\odot$ is for element-wise product. After retrieving the poisoned source image, the attacker solves an optimization problem over an image from the target class as the poisoned image such that the $l_2$ distance of the patched source image $\tilde{s}$ is close to the poisoned image $z$ in the feature space, meanwhile the $l_\infty$ distance between the poisoned image and its initial image $t$ is maintained less than a threshold $\varepsilon$. Formally, the a poisoned image $z$ can be defined as:

$$\arg\min_z ||f(z) - f(\tilde{s})||_2^2$$

$$\text{s.t. } ||z - t||_\infty < \varepsilon,$$

where $f(\cdot)$ is the intermediate features of the DNN and $\varepsilon$ is a small value that ensures the poisoned image $z$ to be not visually distinguishable from the initial target image $t$. The optimization mentioned above only generates a single poisoned sample given a pair of images from source and target classes as well as a fixed location for the
trigger. One can add this poisoned data with the correct label to the training data and train a backdoor model. However, such a model will only have the backdoor triggered when the attacker places the trigger at the same location on the same source image, limiting the practicality of the attack.

To address this shortcoming, Saha et al. [39] propose to manipulate the poisoned images to be closer to the cluster of patched source images rather than being close to only the single patched source image. Inspired by universal adversarial examples [36], Saha et al. [39] minimize the expected value of the loss in Eq. (2) over all possible trigger locations and source images. In their extension, the attacker first samples \( K \) random images \( t_k \) from the target class and initializes poisoned images \( z_k \) with \( t_k \); second, \( \mathcal{X} \) random images \( s_k \) is sampled from the source class and patched with triggers at randomly chosen locations to obtain \( \tilde{s}_k \). For a given \( z_k \) in the poisoned image set, they search for a \( \tilde{s}_{a(k)} \) in the patched image set which is close to \( z_k \) in the feature space \( f(\cdot) \), as measured by Euclidean distance. Next, the attacker creates a one-to-one mapping \( a(k) \) for the poisoned images set and the patched images set. Finally, the attacker performs one iteration of mini-batch projected gradient descent as follows:

\[
\arg\min_{z_k} \sum_{k=1}^{K} \| f(z_k) - f(\tilde{s}_{a(k)}) \|_2^2
\]

\[\text{st. } \forall k : ||z_k - t_k||_\infty < \epsilon.\]

Using the method above, the backdoor trigger samples are given the correct label and only used at test time.

**Hiding Triggers.** In the clean label backdoor attacks mentioned above, the attacker attempts to conduct backdoor attacks without compromising the label of the poisoned samples. On the other hand, it is desirable to make the trigger patterns indistinguishable when mixed with legitimate data in order to evade human inspection.

Liao et al. [25] propose two approaches to make the triggers invisible to human users. The first type of triggers is a small static perturbation with a simple pattern built upon empirical observation. As Liao et al. mention in [25], the limitation of this method is in the increased difficulty for pre-trained models to memorize this type of features, regardless of content and classification models. Consequently, this method of triggers hiding is only practical during the training stage, with access to the entire dataset. The second method to hide the trigger is inspired by the universal adversarial attack [35]. This attack iteratively searches the whole dataset to find the minimal universal perturbation to push all the data points toward the decision boundary of the target class. For each data point, to push this data point towards the target decision boundary, an incremental perturbation \( \Delta V_i \) will be applied. Note that in the second method, although the smallest perturbation (trigger) can be found by the universal adversarial search, the method still needs to apply the trigger on the data points to poison the training set, and retrain the pre-trained model. In their work, the indistinguishability of Trojan trigger examples is attained by a magnitude constraint on the perturbations to craft such examples [31].

Li et al. [24] demonstrate the trade-off between the effectiveness and stealth of Trojans. Li et al. hide triggers on the input images through steganography and reg-
ularization. For the first backdoor attack, the adoption of steganography techniques involves the modification of the least significant bits to embed textual triggers into the inputs. In Li et al.’s regularization approach, they develop an optimization algorithm involving $\mathcal{L}_p$ ($p = 0, 2, \infty$) regularization to effectively distribute the trigger throughout the victim image. When compared to trigger patterns used by Saha et al. [39] that are still visually exposed during the attack phase, the triggers generated by this attack are invisible for human inspectors during both injection and attack phase.

**Dynamic Backdoor.** Dynamic backdoor, as proposed by Salem et al. [40], features a technique whereby triggers for a specific target label have dynamic patterns and locations. This provides attackers with the flexibility for further customizing their backdoor attacks. Salem et al. use random backdoors to demonstrate a naive attack, where triggers are sampled from a uniform distribution. These triggers are then applied to a random location sampled from a set of locations for each input in the injection stage before training the model. The trained backdoored model will now output the specific target label when the attacker samples a trigger from the same uniform distribution and the location set and adds it to any input. Evolving beyond the naive attack, Salem et al. construct a backdoor generating network (BaN) to produce a generative model (similar to the decoder of VAE [37] or generator of GANs [33]) that can transform latent prior distributions (i.e., Gaussian or Uniform distribution) into triggers. The parameters of this BaN is trained jointly with the backdoor model. In the joint training process, the loss between the output of the backdoored model and the ground truth (for the clean input) or the target label (for the poisoned samples) will be backpropagated not only through the backdoored model for an update, but also through the BaN. Upon completion of the model training, the BaN will have learned a map from the latent vector to the triggers that can activate the backdoor model. Salem et al.’s final technique extends the BaN to C-BaN by incorporating the target label information as a conditional input. This changes results in inputs whereby the target label does not need to have its own unique trigger locations, and the generated triggers for different target labels can appear at any location on the input.

**Distributed Backdoor.** In comparison to traditional centralized machine learning settings, federated learning (FL) mitigates many systemic privacy risks and computational costs. Therefore, there has been an explosive growth in the amount of federated learning research. The purpose of backdoor attacks in FL is that an attacker who controls one or several participant(s) may manipulate their local models to simultaneously fit the clean and poisoned training samples. With the aggregation of local models from participants into a global model at the server, the global model would have been influenced by the malicious models to behave maliciously on compromised inputs. Bagdasaryan et al. [3] are the first to mount a single local attacker backdoor attack against a FL platform via model replacement. In their attack, the attacker proposes a target backdoored global model $\mathcal{X}$ they want the server to be in the next round. Then the attacker scales up his local backdoored model to ensure it can survive averaging to ensure the global model to be substituted by $\mathcal{X}$. 
On the other hand, Xie et al. [52] propose a distributed backdoor attack (DBA) which decomposes a global trigger pattern into separate local patterns, and uses these local patterns to inject into the training sets of different local adversarial participants. Fig. 1 illustrates the intuition of the DBA. As we can see, the attackers only need to inject a piece of the global trigger to poison their local models, such that the collective trigger is learned by the global model. Surprisingly, DBA can use a global trigger pattern to activate the ultimate global model as well as a centralized attack does. Xie et al. find that although no singular adversarial party had been poisoned by the global trigger under DBA, the DBA indeed can still behave malicious as a centralized attack.

Fig. 1. Intuition of the Distributed Backdoor Attack (DBA) [52]. An attacker with marked as orange will poison a subset of his training data using only the trigger pattern located at the orange area. The same reasoning applies to the remaining green, yellow, and blue marked attackers.

2.3 Grey-Box Setting

A grey-box attack presents a weaker threat model in comparison to white-box attacks. Recall that white-box attackers have full access to the training data or training process. However, in the grey-box threat model, the attacker’s capability is limited with access to either a small subset of training data or the learning algorithms. **Poisoning Training datasets.** In the former grey-box setting, Chen et al. [8] propose a backdoor attack which injects a backdoor into DNNs by adding a small set of poisoned samples into the training dataset, without directly accessing the victim learning system. Their experiments show that with a single instance (a face-to-face recognition system) as the backdoor key, it only needs 5 poisoned samples to be added to a huge (600,000 images) training set. If the trigger is in the form of a pattern (e.g., glasses for facial recognition), 50 poisoned samples are sufficient for a respectable attack success rate. **Trojaning NN.** The grey-box setting, which does not provide the attacker with access to the training or test data, instead providing full access to the target DNN models, is
observed in transfer learning pipelines. The attacker only has access to a pre-trained DNN model, and this setting is more common than the former grey-box assumption of access to a subset of data. Liu et al’s Trojaning attacker [29] has both a clean pre-trained model and a small auxiliary dataset generated by reverse engineering the model. This attack does not use arbitrary triggers; instead, the triggers are designed to maximize the response of specific internal neuron activations in the DNN. This creates a higher correlation between triggers and internal neurons, by building a stronger dependence between specific internal neurons and the target labels, retraining the model with the backdoor requires less training data. Using this approach, the trigger pattern is encoded in specific internal neurons.

2.4 Black-Box Setting

The prior backdoor threat models assume an attacker capable of compromising either the training data or the model training environment. Such threats are unlikely in many common ML use-case scenarios. For example, organizations train on their own private data, without outsourcing the training computation. On-premise training is typical in many industries, and the resulting models are deployed internally with a focus on fast iterations. Collecting training data, training a model, and deploying it are all parts of a continuous, automated production pipeline that is accessed only by trusted administrators, without the potential of incorporating malicious third parties.

Compromising Code. Bagdasaryan and Shmatikov [2] propose a code-only backdoor attack in which the adversary does not need to access the training data or the training process directly. Yet, that attack still produces a backdoored model by adding malicious code to ML codebases that are built with complex control logic and dozens of thousands of code blocks. The key to their method lies in the following assumption: compromising code in ML codebases stealthily is realistic, as it is reasonable for most of the cases that correctness tests of ML codebases are not available. For example, the three most popular PyTorch repositories on GitHub, fairseq, transformers and fast.ai, all include multiple loss computations and complex model architectures. The attack will remain unnoticed under unit testing when the adversaries add a new backdoor loss function unified with other conventional losses, as the intention of this malicious loss (and backdoor attacks as a whole) is to preserve normal training behavior.

Specifically, they model backdoor attacks through the lens of multi-objective optimization (w.r.t. multiple loss functions). The loss for the main task $m$ should perform regularly during training; however the backdoor loss is computed on the poisoned samples that are synthesized by the adversary’s code. The two losses are then unified into one overall loss through a linear operation. The authors solve their multi-objective optimization problem via Multiple Gradient Descent Algorithm (MGDA) [11].

Live Trojan. Costales et al. [9] propose a live backdoor attack which patches model parameters in system memory to achieve the desired malicious backdoor behavior. The attack setting assumes that the attacker has the capability to modify data in the victim process’s address space (/proc/[PID]/map, /proc/[PID]/mem). Countless
possibilities exist to enable this power. For example, trojanning a system library, or remapping memory between processes with a malicious kernel module, which has been proved effective in Stuxnet [22]. After the attacker establishes write capabilities in the relevant address space, they need to find the weights of the DNN stored in memory. The proposal suggests either Binwalk [17] or Volatility [26] to find signatures of the networks by detecting a large swaths of binary storing weights. Once the malware has scanned the memory and the weights of DNNs located, masked retraining is used to modify only the selected parameters which are the most significant neurons of the DNNs to perform as the backdoor behavior. In identifying the parameters of the model which will yield a high attack success rate, the attacker will compute the average gradient for a continuous subset of parameters on one layer with a window size across the entire poisoned dataset. Parameter values with larger absolute average value gradients indicate that the model would likely benefit from modifying the parameter value. After calculating the patches, simple scripts will load the patched weights into binary files to which the malware can apply.

Although this attack needs knowledge of the DNN’s architecture, it is possible for an attacker to take a snapshot of the victim’s system, extract the system image, and use forensic and/or reverse-engineering tools to achieve this indirectly and run code on the victim system. As such, we categorize this type of backdoor attacks as a black-box attack.

3 Detecting and Defending Backdoors

Attacks from white- to black-box settings have been developed with the goal of subverting the machine learning model to include backdoored behavior. However, any model trainer, or holder may take proactive steps to detect and defend their models against this threat. This section will describe at length how this attack may be thwarted. Overall, the task of detecting and defending against a backdoor attack can be divided into three key sub-tasks:

1. **Task 1:** Detecting the existence of the backdoor. For a given model, it is difficult to know if the model is compromised (i.e., a model with a backdoor) or not. The first step of detecting and defending against the backdoor attack is to analyze the model and determine if there is a backdoor present in this model.
2. **Task 2:** Identifying the backdoor trigger. When a backdoor is detected in a model, the second step is usually to identify which pattern (including its size, location, texture, and so on) is used as the trigger.
3. **Task 3:** Mitigating the backdoor attack. After identifying the existence of a backdoor, the mitigation of such an attack is to remove the backdoor behavior from the model. Note that backdoor models can be made to be robust against transfer learning or fine-tuning [54].

It is noted that not all detection and defense techniques will support all three sub-tasks. As some may assume prior knowledge that a backdoor has already been
detected, and the proposal only contains techniques to recover the trigger or mitigate the attack.

Figure 2 shows an overview of the DNN model training and deployment process. It can be broken up into four general steps from data preparation, model training, model testing, and model deployment. As discussed in Section 2, most existing poisoning attacks target the model training (or model retraining) step. Thus, investigating if the model contains a backdoor, reconstructing potential triggers, and/or mitigating any backdoor attacks must occur after this training step. Thus, mitigation strategies will be employed either during model testing (i.e., pre-deployment) or at the model’s runtime (i.e., post-deployment). Thus, depending on when the inspection occurs, existing detection and defense techniques can be divided into two categories: pre-deployment techniques or post-deployment techniques.

3.1 Pre-deployment Techniques

No Inspections

There exists work [56, 27] attempting to directly mitigate the backdoor attack without inspecting the model behavior. The key technique behind these methods is to compress the model (e.g., by model pruning or similar techniques) or fine-tune the model with benign inputs to alter the model behavior hoping that the backdoor behavior is eliminated. Specifically, Zhao et al. [56] found that model pruning can remove some behaviors of a trained model, and potentially it can remove the backdoor of the model if pruning is purely using benign data.

Liu et al. [27] observe that pruning the model alone does not guarantee the removal of the model backdoor behavior. This is because the malicious model may use the same neuron to demonstrate both benign and malicious behaviors. Thus, if the neuron is removed, the model accuracy will be lower than that of the original model. This would be an undesirable consequence even though the model backdoor is removed. However, if this neuron is not pruned, the backdoor behavior is retained and the model continues to be malicious, also undesirable. Similarly, fine-tuning the model does not necessarily remove the model backdoor, as some attacks [54] target
transfer learning scenarios where fine-tuning is needed. To solve this problem, Yao et al. propose Fine-Pruning, which combines the strengths of both fine-tuning and pruning to effectively nullify backdoors in DNN models. Fine-Pruning first removes backdoor neurons using pruning and fine-tuning the model in order to restore the drop in classification accuracy on clean inputs (which is introduced in the previous pruning procedure).

There are some limitations of these types of defenses. Firstly, model pruning itself has unknown effects on the model. Even though model accuracy after pruning does not decrease too much, many other important model properties, such as model bias (sometimes known as fairness) and model prediction performance, are not guaranteed to be the same. Using such models may potentially lead to severe consequences. Secondly, these mitigation techniques assume the access to the training process and clean inputs, which conflicts with poisoning-based attacks.

### Pre-deployment Model Inspections

Before the model is deployed, it is possible to check whether the model has been backdoored directly. This kind of strategy works without the running of the model, so it is also called static detection. For these types of techniques, some will require a large set of benign inputs to identify backdoors, such as Neural Cleanse (NC) [49], whereas others do not require much data (i.e., a limited number or even zero samples), such as ABS [28].

**Neural Cleanse.** Wang et al. [49] propose Neural Cleanse (NC), a pre-deployment technique to inspect DNNs, identify backdoors, and mitigate such attacks. Figure 3 illustrates the key observation that enables NC. The top figure shows a clean model with three output labels. If we want to perturb inputs belonging to C to A, more modification is needed to move samples across decision boundaries. The bottom
figure shows the infected model, where the backdoor changes decision boundaries leading to a small perturbation value for changing inputs belonging to B and C to A.

Based on this observation, NC proposes to first compute a universal perturbation, which is the minimized amount of change to make the model predict a given target label. If the perturbation is small enough (i.e., smaller than a given threshold), NC considers it as one trigger. It then verifies this by adding this trigger to a large number of benign inputs and tests if it is really a trigger and tries to optimize it based on prediction results. After identifying the trigger, it can mitigate the attack by either using a filter (i.e., to detect images with such a trigger pattern) or patching the DNN by removing the corresponding behaviors by pruning the neural network.

NC has a number of limitations. First, NC makes an incorrect assumption that if pixels in a small region have a strong influence on the output result, they are treated as backdoor triggers. This results in NC confusing triggers with strong benign features. In many tasks, there exist strong local features, where a region of pixels is important for one output label, for example, the antlers of deers in CIFAR-10. Secondly, NC assumes that the trigger has to be small and in the corner areas. These are heuristics, which do not hold for many attacks. For example, Salem et al. [40] proposes a dynamic attack, where triggers can be added to different places and can successfully bypass NC. Thirdly, NC requires a significant number of testing samples to determine if a backdoor exists in a model or not. In real-world scenarios, such a large number of benign inputs may not exist. Lastly, it is designed purely for input space attacks, and it does not work for feature space attacks, such as using Nashville and Gotham filters as triggers [28].

ABS is built on top of two key observations. The first is that successful attacks entail compromised neurons. In existing attacks, the backdoored model recognizes the trigger as a strong feature of the target label to achieve a high attack success rate. Such a feature is represented by a set of inner neurons, which are referred to as compromised neurons. The second observation is that compromised neurons represent a subspace for the target label that cuts across the whole space. This idea is shown in Figure 4. The feature space surface of a benign model (left figure in Figure 4) and that of a backdoored model are noticeably different. For a backdoored model, there

![Fig. 4. Overview of ABS observations [28]. The left figure shows the feature surface of a benign model. The middle figure shows the feature surface of a model with a backdoor. The right figure shows a slice of surface for the backdoored model. The red dot in the middle figure shows an state where the attack happens, and it corresponds to the dash line in the right figure.](image-url)
exists an cut of the surface that is significantly different from the benign model due to
the injected backdoor. As it works for all inputs, it will affect every prediction results
once it is activated. Thus, it will interact with the whole interface. The phenomenon
is demonstrated in the right figure of Figure 4. When a neuron value is assigned to
a special value, i.e., the trigger pixel value, the output will significantly deviate from
normal.

Based on these observations, Liu et al. propose Artificial Brain Stimulation
(ABS). For any given input, ABS first predicts its label using the neural network.
Then, it enumerates all neurons and performs a brain stimulation process. Namely,
for each neuron, it tries to change its activation value to all possible values and simultaneously observes the value changes in the output. If there is one neuron whose
behavior is similar to the right figure in Figure 4, ABS treats it as a backdoor. To
reconstruct the backdoor trigger, ABS then performs a reverse engineering process,
which will try to find an input pattern that can strongly activate these compromised
neurons and trigger the attack.

ABS also introduces a new type of backdoor attacks, which is the feature space
attack. Namely, the trigger is no longer an input pattern (i.e., a region with specific
pixel values), but feature space patterns represent high-level features (e.g., an image
filter). However, this attack also has its own limitations. Firstly, it assumes one back-
door for each class. This may not hold in practice, and backdoors have been shown
to be dynamic. Secondly, it currently enumerates neurons one by one, assuming
the presence of a strong correlation between one neuron and the backdoor behavior,
which may be hidden or overridden by more advanced attacks.

3.2 Post-deployment Techniques

In addition to static approaches functioning before models are deployed, there is also
work that monitors the model at runtime and determines if the model has a backdoor
and more importantly, if it has been triggered by an input or not. In this setting, the
defense or detection system can inspect individual inputs, offering a focused means
of directly reconstructing the trigger by inspecting the attack input.

Fig. 5. Overview of STRIP [13].
STRIP. Gao et al. [13] propose STRong Intentional Perturbation (STRIP), a run-time trojan attack detection system. The workflow of STRIP is shown in Figure 5. Firstly, STRIP will perturb each input by adding benign samples drawn from the test samples to obtain a list of perturbed inputs $X^{p_1}, X^{p_2}, ..., X^{p_N}$. These inputs are the overlap of an benign input and the given input. Next, it will feed all these inputs to the DNN model. Note that if the input contains a trigger, it is highly likely that a majority of the perturbed inputs will also yield predictions with the malicious output label result (due to the existence of the trigger), whereas for a benign input, the results are closer to random. As a result, STRIP only needs to examine every prediction result, and can then make a judgement on if the input will trigger the backdoor or not.

STRIP can effectively detect backdoor models and inputs that trigger the backdoor if the trigger lies in the corners of the image or at least does not overly overlap with the main contents. Such an example shown in Figure 5. However, if the trigger does overlap with the contents (e.g., overlap with digits in Figure 5), the detection will fail because the texture of the trigger will also be changed by the perturbations. Salem et al.’s [40] proposed dynamic backdoor attack uses triggers that can be in the middle of the image.

4 Applications of Backdoors

4.1 Watermarking

Digital Watermarking conceals information in a piece of media (e.g., sound, video, or images) to enable a party to verify the authenticity or the originality of the media. This watermark, however, must also be resilient to tampering and other actors seeking to subvert the legitimate piece of media.

Adi et al. [1] propose an IP protection method for DNNs by applying the backdoor to watermark DNNs. They present cryptographic modeling for both tasks of watermarking and backdooring DNNs, and show that the former can be constructed from the latter (through a cryptographic primitive known as commitment) in a blackbox manner. The definition of the backdoor attack that Adi et al. provided in a cryptographic framework is as follows: Given a trigger set $T$ and a labeling function $T_L$, the backdoor shall be termed as $b = (T, T_L)$. The backdooring algorithm $\text{Backdoor}(O^f, b, M)$ is a probabilistic polynomial time (PPT) algorithm that receives as input an oracle to $f$ (ground-truth labeling function $f: D \to L$, where $D$ is input space, $L$ is output space), the backdoor $b$ and a model $M$, and outputs $\hat{M}$. $\hat{M}$ is considered backdoored if

$$\Pr_{x\in\tilde{D}\setminus T}[f(x) \neq \text{Classify}(\hat{M}, x)] \leq \epsilon,$$

$$\Pr_{x\in T}[T_L(x) \neq \text{Classify}(\hat{M}, x)] \leq \epsilon,$$

(3)

where $\tilde{D}$ is the meaningful input, $\text{Classify}(M, x)$ is a deterministic polynomial-time algorithm that, for an input $x \in D$ outputs a value $M(x) \in L \setminus \{\bot\}$, and $\bot$ is an undefined output label. This definition presents two ways to embed a backdoor. The
first is that the backdoor is implanted into a pre-trained model. The second is the adversary can train a new model from scratch. A watermarking scheme can be split into three key components.

1. Generation of the secret “marking” key \( m_k \). This key will be embedded as the watermark. A public verification key \( v_k \) is also generated and will be used later to detect the watermark. In watermarking via backdoors, the backdoor is the marking key, while a commitment (the cryptographic primitive) used to generate the backdoor is the verification key.

2. Embedding the watermark (a backdoor \( b \)) into a DNN model. Through poisoned training data or retraining as previously described in Section 2.1, the watermark (backdoor) can be embedded.

3. Verifying the presence of the watermark. Provided \( m_k, v_k \), for a backdoor test \( b = (T, T_L) \). If \( \forall i \in T : T_{L(i)} \neq f(t(i)) \), proceed to the next step, otherwise the verification fails. Despite the detection of the watermark, one must verify the integrity of the commitment, i.e., if it was tampered or not. In the final step, the accuracy of the algorithm is verified. For all \( i \in 1, \ldots, n \), if more than \( \varepsilon |T| \) elements from \( T \) does not satisfy \( \text{Classify}(t(i), M) = T_{L(i)} \), then the verification fails, otherwise the commitment has been successfully verified.

Adi et al. [1] prove their method upholds the properties of:

- **Functionality-preserving**: the prediction accuracy of the model should not be negatively influenced by the presence of the watermark.
- **Unremovability**: an adversary with full knowledge of the watermark generation process should not be able to remove the watermark from the model.
- **Unforgeability**: an adversary with only the verification key should not be able to demonstrate ownership of the marking key.
- **Non-trivial ownership**: with knowledge of the watermark generation algorithm, a third party should not be able to generate marker and verification key pairs, and claim models for future models.

Li et al. [23], however, observe that the watermarking system proposed by Adi et al. [1] makes the assumptions that only one backdoor (watermark) may be inserted into the model. For example, Salem et al. [40]’s Dynamic Backdoors contain multiple backdoors. The existence of multiple backdoors would result in multiple valid watermarks, and thus void the Unforgeability claim. The insertion of multiple backdoors would also impact the Unremovability of the original backdoor, otherwise termed as the persistence of the watermark. In response, Li et al. leverage two data preprocessing techniques that use out-of-bound values and null-embedding to improve the persistence of the watermark against other attackers and limit the effects of retraining in the event that another backdoor is to be injected on top of the existing backdoor. Li et al. also introduce wonder filters, a primitive to enable the embedding of bit-sequences (from the marker key) into the model.

The largest hurdle to overcome in the application of the backdoor attack as a means to watermark DNNs, is that neural networks are fundamentally designed to
Deep Learning Backdoors

be tuned and trained incrementally. Li et al. propose a model piracy attack setting whereby an adversary wants to stake its own ownership claims on the model, or destroy the original owner’s claims. To defend against this attack, Li et al. design a DNN watermarking system based on wonder filters that strongly authenticates owners by embedding (into the DNN) a filter described by the owner’s private key. Where Li et al’s work differs to Adi et al. is in the wonder filter $W$, which is a two-dimensional digital filter that can be applied to any input image. This filter will have 3 possible permutations for each pixel, transparent, positive change, or negative change, with a majority of filter pixels being transparent. Thus, $W$ is defined by the position, size, and values of a 0/1 bit pattern block.

When Li et al. apply out-of-bound values, they translate the 0/1 bit pattern of $W$ as out-of-bound values in the input images. A set of training data is processed with the filter. They then flip the values of the wonder filter to create an inverted wonder filter $W^-$. The inverted filter $W^-$ is then applied to the same set of training data processed by $W$. The set of images filtered by $W$ are labeled as the target class label, while the $W^-$ filtered data is labeled as the original class label before the data is used to train the backdoored model. As for the normal and null embeddings approach, the normal and null embeddings serve complementary objectives. The normal embedding injects the desired marker into the model, while the null embedding “locks down” the model, so no additional watermarks may be added.

Li et al.’s process of watermarking the image is similar to Adi et al.’s process, with the same three key processes of: generating the secret “marking”, or in this instance, the wonder filter $W$, embedding the watermark (and/or additionally locking down the model), and finally, the process of verifying the watermark, by using the image to compute $W$ and an associated label. After applying $W$ to a random set of images, it is expected that an authentic watermark should yield a majority of the target class label, instead of a random assortment of classes as expected from a random set of images, without $W$.

Li et al. also provide a security analysis to prove that their approach can uphold the requirements of reliability, no false positives, unforgeability, and persistence, whereby Reliability describes that for given an input $x$, a poisoned input ($x \oplus W$, or $x \oplus W^-$), the backdoored DNNs will produce the predefined output in a deterministic manner. No False Positives denotes that a verifier should not be capable of judging a clean model as the watermarked model. Unforgeability ensures that the watermark injected on a DNN has a strong association with its owner, and Persistence guarantees that the watermark embedded cannot be corrupted or removed by an adversary.

4.2 Adversarial Example Detection

In Gotta Catch [43], Shan et al. observe that the backdoor attack will alter the decision boundary of the DNN models. Following the injection of a backdoor, the decision boundary of the original clean model will mutate. The mutation will result in triggers establishing shortcuts in the decision boundary of the backdoored model.

On the contrary, there is a common approach of adversarial attacks to find adversarial examples; for example, universal adversarial attacks [35, 41], will try to
iteratively search the whole dataset to find similar shortcuts to use as their universal adversarial examples. Based on this observation, the shortcut created by a backdoor can act as a trapdoor to capture the adversarial attacker’s optimization process, detect, and/or recover from the adversarial attack [42]. The trapdoor implementation uses techniques similar to those found in BadNets backdoor attacks [15]. The authors define the trapdoor perturbation (the trigger) from multiple dimensions, e.g., mask ratio, size, pixel intensities, and relative locations.

References

1. Adi, Y., Baum, C., Cissé, M., Pinkas, B., and Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018. (2018), pp. 1615–1631.
2. Bagdasaryan, E., and Shmatikov, V. Blind backdoors in deep learning models. arXiv preprint arXiv:2005.03823 (2020).
3. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. In International Conference on Artificial Intelligence and Statistics (2020), pp. 2938–2948.
4. Barni, M., Kallas, K., and Tondi, B. A new backdoor attack in cnns by training set corruption without label poisoning. CoRR abs/1902.11237 (2019).
5. Barni, M., Kallas, K., and Tondi, B. A new backdoor attack in cnns by training set corruption without label poisoning. In 2019 IEEE International Conference on Image Processing (ICIP) (2019), IEEE, pp. 101–105.
6. Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. Vggface2: A dataset for recognising faces across pose and age. In 13th IEEE International Conference on Automatic Face & Gesture Recognition, FG 2018, Xi’an, China, May 15-19, 2018 (2018), pp. 67–74.
7. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., and Srivastava, B. Detecting backdoor attacks on deep neural networks by activation clustering. In Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019. (2019).
8. Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017).
9. Costales, R., Mao, C., Norwitz, R., Kim, B., and Yang, J. Live trojan attacks on deep neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (2020), pp. 796–797.
10. Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA (2009), pp. 248–255.
11. Désidéri, J.-A. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. Comptes Rendus Mathematique 350, 5-6 (2012), 313–318.
12. Dumford, J., and Scheirer, W. J. Backdooring convolutional neural networks via targeted weight perturbations. CoRR abs/1812.03128 (2018).
13. GAO, Y., XU, C., WANG, D., CHEN, S., RANASINGHE, D. C., AND NEPAL, S. Strip: A defence against trojan attacks on deep neural networks. In Proceedings of the 35th Annual Computer Security Applications Conference (2019), pp. 113–125.
14. GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial nets. In Advances in neural information processing systems (2014), pp. 2672–2680.
15. GU, T., DOLAN-GAVITT, B., AND GARG, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017).
16. GUO, W., WANG, L., XING, X., DU, M., AND SONG, D. Tabor: A highly accurate approach to inspecting and restoring Trojan backdoors in AI systems. arXiv preprint arXiv:1908.01763 (2019).
17. HEFFNER, C. Binwalk: Firmware analysis tool. URL: https://code.google.com/p/binwalk/visited on 03/03/2013 (2010).
18. HOUBEN, S., STALLKAMP, J., SALMEN, J., SCHLIPSING, M., AND IGEL, C. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013 (2013), pp. 1–8.
19. KOLOURI, S., SAHA, A., PIRSIAVASH, H., AND HOFFMANN, H. Universal litmus patterns: Revealing backdoor attacks in cnns. CoRR abs/1906.10842 (2019).
20. KONG, Y., AND ZHANG, J. Adversarial audio: A new information hiding method and backdoor for dnn-based speech recognition models. CoRR abs/1904.03829 (2019).
21. KURITA, K., MICHEL, P., AND NEUBIG, G. Weight poisoning attacks on pretrained models. In Annual Conference of the Association for Computational Linguistics (ACL) (July 2020).
22. LANGNER, R. Stuxnet: Dissecting a cyberwarfare weapon. IEEE Security & Privacy 9, 3 (2011), 49–51.
23. LI, H., WILLSON, E., ZHENG, H., AND ZHAO, B. Y. Persistent and unforgeable watermarks for deep neural networks. arXiv preprint arXiv:1910.01226 (2019).
24. LI, S., XUE, M., ZHAO, B. Z. H., ZHU, H., AND ZHANG, X. Invisible backdoor attacks on deep neural networks via steganography and regularization. arXiv preprint arXiv:1909.02742 (2020).
25. LIAO, C., ZHONG, H., SQUICCIANI, A., ZHU, S., AND MILLER, D. Backdoor embedding in convolutional neural network models via invisible perturbation. arXiv preprint arXiv:1808.10307 (2018).
26. LIGH, M. H., CASE, A., LEVY, J., AND WALTERS, A. The art of memory forensics: detecting malware and threats in windows, linux, and Mac memory. John Wiley & Sons, 2014.
27. LIU, K., DOLAN-GAVITT, B., AND GARG, S. Fine-pruning: Defending against backdoor attacks on deep neural networks. In International Symposium on Research in Attacks, Intrusions, and Defenses (2018), Springer, pp. 273–294.
28. LIU, Y., LEE, W.-C., TAO, G., MA, S., AAFER, Y., AND ZHANG, X. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019), pp. 1265–1282.
29. LIU, Y., MA, S., AAFER, Y., LEE, W.-C., ZHAI, J., WANG, W., AND ZHANG, X. Trojaning attack on neural networks. The Network and Distributed System Security Symposium (NDSS) (2017).
30. LIU, Y., MA, S., AAFER, Y., LEE, W.-C., ZHAI, J., WANG, W., AND ZHANG, X. Trojaning attack on neural networks. In 25nd Annual Network and Distributed System
31. Liu, Y., Mondal, A., Chakraborty, A., Zuzak, M., Jacobsen, N., Xing, D., and Srivastava, A. A survey on neural trojans. *IACR Cryptol. ePrint Arch.* 2020 (2020), 201.

32. Lovisotto, G., Eberz, S., and Martinovic, I. Biometric backdoors: A poisoning attack against unsupervised template updating. *CoRR abs/1905.09162* (2019).

33. Mirza, M., and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

34. Møgelose, A., Liu, D., and Trivedi, M. M. Traffic sign detection for us roads: Remaining challenges and a case for tracking. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (2014).

35. Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (2017), pp. 86–94.

36. Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1765–1773.

37. Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).

38. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

39. Saha, A., Subramanya, A., and Pirsiavash, H. Hidden trigger backdoor attacks. *arXiv preprint arXiv:1910.00033* (2019).

40. Salem, A., Wen, R., Backes, M., Ma, S., and Zhang, Y. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675* (2020).

41. Shafahi, A., Najibi, M., Xu, Z., Dickerson, J., Davis, L. S., and Goldstein, T. Universal adversarial training. *arXiv preprint arXiv:1811.11304* (2018).

42. Shan, S., Wenger, E., Wang, B., Li, B., Zheng, H., and Zhao, B. Y. Gotta catch ’em all: Using honeypots to detect adversarial attacks on neural networks. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS* (2020).

43. Shan, S., Willson, E., Wang, B., Li, B., Zheng, H., and Zhao, B. Y. Gotta catch’em all: Using concealed trapdoors to detect adversarial attacks on neural networks. *arXiv preprint arXiv:1904.08554* (2019).

44. Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy* (2017), IEEE, pp. 3–18.

45. Tan, T. J. L., and Shokri, R. Bypassing backdoor detection algorithms in deep learning. *CoRR abs/1905.13409* (2019).

46. Tang, D., Wang, X., Tang, H., and Zhang, K. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. *CoRR abs/1908.00686* (2019).

47. Tran, B., Li, J., and Madry, A. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems* (2018), pp. 8000–8010.

48. Turner, A., Tsipras, D., and Madry, A. Clean-label backdoor attacks.
49. **Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y.** Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 707–723.

50. **Wolf, L., Hassner, T., and Maoz, I.** Face recognition in unconstrained videos with matched background similarity. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011* (2011), pp. 529–534.

51. **Xiang, Z., Miller, D. J., and Kesidis, G.** Revealing backdoors, post-training, in DNN classifiers via novel inference on optimized perturbations inducing group misclassification. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020).

52. **Xie, C., Huang, K., Chen, P.-Y., and Li, B.** Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations* (2019).

53. **Yang, Z., Iyer, N., Reimann, J., and Virani, N.** Design of intentional backdoors in sequential models. *CoRR abs/1902.09972* (2019).

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

55. **Yao, Y., Li, H., Zheng, H., and Zhao, B. Y.** Regula sub-rosa: Latent backdoor attacks on deep neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019* (2019).

56. **Zhao, B., and Lao, Y.** Resilience of pruned neural network against poisoning attack. In *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)* (2018), IEEE, pp. 78–83.