This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Attack To Defend: Exploiting Adversarial Attacks for Detecting Poisoned Models

Samar Fares   Karthik Nandakumar
Mohamed bin Zayed University of Artificial Intelligence, UAE (MBZUAI)
{samar.fares,karthik.nandakumar}@mbzuai.ac.ae

Abstract

Poisoning (trojan/backdoor) attacks enable an adversary to train and deploy a corrupted machine learning (ML) model, which typically works well and achieves good accuracy on clean input samples but behaves maliciously on poisoned samples containing specific trigger patterns. Using such poisoned ML models as the foundation to build real-world systems can compromise application safety. Hence, there is a critical need for algorithms that detect whether a given target model has been poisoned. This work proposes a novel approach for detecting poisoned models called Attack To Defend (A2D), which is based on the observation that poisoned models are more sensitive to adversarial perturbations compared to benign models. We propose a metric called sensitivity to adversarial perturbations (SAP) to measure the sensitivity of a ML model to adversarial attacks at a specific perturbation bound. We then generate strong adversarial attacks against an unrelated reference model and estimate the SAP value of the target model by transferring the generated attacks. The target model is deemed to be a `trojan` *if its SAP value exceeds a decision threshold. The A2D framework requires only black-box access to the target model and a small clean set, while being computationally efficient. The A2D approach has been evaluated on four standard image datasets and its effectiveness under various types of poisoning attacks has been demonstrated.*

1. Introduction

Deep neural networks (DNN) have played a key role in transforming applications such as autonomous vehicles [3], security [43], finance [16], and healthcare [2]. However, training of accurate DNN models requires a large amount of data and significant computing resources. To mitigate this problem, it is a common practice among machine learning (ML) practitioners to (a) leverage third-party datasets, (b) use pre-trained models obtained from third-party sources as a starting point for model development (because it requires less training resources compared to learning a model from scratch), or (c) completely outsource the model training to a third-party (e.g., cloud) service provider. The advent of foundation models [4] can be expected to further accelerate the trend of transfer learning from pre-trained models. However, the above approaches may introduce a vulnerability called *backdoor* (a.k.a. *poisoning* or *trojan*) attack, where an attacker manipulates the training dataset to control the prediction behavior of the trained/pre-trained model [17]. A poisoned model works normally for clean (unmodified) input samples, but behaves maliciously when activated by specific triggers in the input sample. This can compromise the safety of systems that make use of the poisoned model.

Techniques for defending ML models from poisoning attacks can be broadly classified into two categories [17]: (i) identifying poisoned samples in the training data (e.g., [46]) and (ii) detecting and purifying poisoned models (e.g., [49]). In this work, we assume that the model has already been trained by the adversary and the complete dataset used for training is no longer available to the defender. Hence, this work focuses only on the latter category of defense techniques, which can be further sub-divided into *detection-only* methods and *purification* (patching) methods. The goal of poisoned model detection is to determine whether a given target model is *benign* or *poisoned* [17, 28]. On the other hand, purification methods attempt to remove/mitigate the backdoor vulnerability from the target model. It is worth emphasizing that detection is often the first step in any purification process, and having an effective poisoned model detector is critical for two reasons. Firstly, as the triggers become increasingly stealthy [1, 34], purification becomes inherently hard (especially for unseen attacks). In such cases, having an accurate trigger-agnostic detector that can reject suspicious models is the only line of defense. Secondly, correct identification of benign models can avoid subjecting them to purification, thereby preserving their clean accuracy from any degradation introduced during purification.

Though many algorithms have been proposed for detecting poisoned models [17][7, 20, 24, 29, 49, 50, 52, 53, 56], most existing solutions have one or more of the following limitations: (i) *poor generalizability*: they are effective against only a specific type of attack (or a small class of attacks) and lack the ability to generalize to new/unseen types of attacks [50], (ii) *high computational complexity*: some methods require training of numerous shadow models for meta-learning of a poisoned model detector [24, 53], (iii) *data access*: some methods need access to a large set of clean samples [24, 49] or the entire training data (used to train the poisoned model) [6], and (iv) *knowledge of the target model*: they either require white-box access to the target model or assume that the target model architecture is known [6, 49]. The search for an effective poisoned model detector with good generalization capabilities, low computational overhead, as well as minimal access to the target model and clean set has remained elusive thus far.

In this work, we propose a simple yet effective method for detecting poisoned models by examining their behavior under adversarial attacks. Adversarial attacks [18] typically make small (often imperceptible) perturbations to the input sample during inference in order to fool a given ML model. While there are some recent studies connecting adversarial and backdoor attacks [32, 35, 55, 56], our approach is based on the idea that *poisoned models are more sensitive to adversarial samples compared to their benign counterparts*.

The three key contributions of this work are as follows:

1. We propose a novel approach called Attack To Defend (A2D) for detecting whether a given target model has been poisoned. The A2D framework generates strong adversarial attacks against a reference model and probes the target model using the generated attacks.
2. We propose a new metric called *sensitivity to adversarial perturbations* (SAP) to quantify the sensitivity of a ML model to adversarial attacks at a specific perturbation bound. The SAP metric enables normalized comparison of the adversarial sensitivity (robustness) of ML models. If the target model is poisoned, it exhibits higher SAP compared to a benign model, which can be exploited for poisoned model detection.
3. Experiments demonstrate that the A2D approach generalizes well to unseen attacks, while requiring only black-box access to the target model and a small clean set. We examine the adversarial pathways of poisoned models and show that the backdoor path present in poisoned models renders them more vulnerable to adversarial attacks.

2. Background and Related Work

2.1. Preliminaries

Let $M_\theta : \mathcal{X} \to \mathcal{Y}$ be a machine learning (ML) model that learns the mapping between an input space $\mathcal{X}$ and a label space $\mathcal{Y}$. In this work, we assume that $\mathcal{X} \subseteq [0,1]^d$ and $\mathcal{Y} = \{1, 2, \cdots, m\}$, where d is the dimension of the input sample and m is the number of classes.

Clean/Benign Model: Let $(\mathbf{x}, y)$ denote a *clean* labeled sample, where $\mathbf{x} \in \mathcal{X}$ denotes the input to the ML model, $y \in \mathcal{Y}$ is the corresponding ground-truth class label, and "clean" implies that both the input and the label have not been modified by any adversary. Let $\mathcal{G}_C : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ denote the joint probability distribution of the clean labeled samples. Let $\mathcal{D}_T = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_T}$ be a dataset containing N_T clean samples available for model training. The clean (a.k.a. benign) model is learned by minimizing the following empirical risk.

$$\min_\theta \frac{1}{N_T} \sum_{i=1}^{N_T} \mathcal{L}(M_\theta(\mathbf{x}_i), y_i), \tag{1}$$

where $\mathcal{L}$ denotes the loss function used to train the model. The clean accuracy ($\mathcal{A}_C$) of a ML model M_θ is defined as:

$$\mathcal{A}_C(M_\theta) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{G}_C} \, \mathbb{I}[M_\theta(\mathbf{x}) = y], \tag{2}$$

where $\mathbb{E}$ denotes the expectation operator and $\mathbb{I}[b]$ is an indicator random variable that takes value 1 when b is true and 0 otherwise. Note that $0 \leq \mathcal{A}_C \leq 1$.

Adversarial Attacks: Given a model M_θ, a clean sample $(\mathbf{x}, y) \sim \mathcal{G}_C$, the adversarial input $\ddot{\mathbf{x}} = \mathbf{x} + \mathbf{r}$ can be obtained by solving the following constrained optimization problem:

$$\max_{\mathbf{r}} \mathcal{L}(M_\theta(\mathbf{x} + \mathbf{r}), y), \text{such that } \|\mathbf{r}\|_p \leq \epsilon, \tag{3}$$

where $\mathbf{r}$ denotes the adversarial perturbation, ϵ is the *perturbation bound*, and $\|.\|_p$ denotes the ℓ_p norm. Unless otherwise stated, we use only ℓ_∞ norm-constrained adversarial attacks in this work. The robust accuracy ($\mathcal{A}_R$) of a model M_θ for a given perturbation bound ϵ is defined as:

$$\mathcal{A}_R(M_\theta, \epsilon) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{G}_C} \, \mathbb{I}[M_\theta(\mathbf{x} + \mathbf{r}) = y], \tag{4}$$

where $\mathbf{r}$ is obtained using eq. (3). For strong adversarial attacks, the robust accuracy $\mathcal{A}_R(M_\theta, \epsilon)$ can be expected to be small (closer to 0). It must be emphasized that adversarial attacks are prediction-time attacks and do not affect the model parameters. Some of the well-known adversarial attacks include FGSM [18], PGD [30], C-W [5], and AutoAttack [10]. These attacks primarily differ in how the optimization problem in eq. (3) is solved.

Poisoned Model: Given a clean sample $(\mathbf{x}, y) \sim \mathcal{G}_C$, a trigger pattern $\mathbf{t}$, and a target label $\tilde{y} \neq y$ for the poisoning attack, the poisoned sample is obtained as $(\tilde{\mathbf{x}}, \tilde{y})$, where the poisoned input is defined as:

$$\tilde{\mathbf{x}} = \mathcal{I}_\phi(\mathbf{x}, \mathbf{t}). \tag{5}$$

Here, $\mathcal{I}$ denotes the trigger insertion function parameterized by ϕ. A poisoned (a.k.a. trojaned or backdoored) model is typically learned by corrupting a fraction ρ (known as the *poisoning rate*) of the samples from the training set $\mathcal{D}_T$ based on the trigger $\mathbf{t}$ and minimizing the following objective:

$$\min_{\theta} \left(\frac{1}{N_T} \sum_{i=1}^{N_T} \mathcal{L}(M_\theta(\mathbf{x}_i), y_i) + \frac{1}{N_P} \sum_{i=1}^{N_p} \mathcal{L}(M_\theta(\tilde{\mathbf{x}}_i), \tilde{y}) \right), \tag{6}$$

where $N_P = \lfloor \rho N_T \rfloor$, $\lfloor \cdot \rfloor$ is the floor function, and $\tilde{\mathbf{x}}_i = \mathcal{I}_\phi(\mathbf{x}_i, \mathbf{t})$. Let θ_P be the model parameters obtained by solving eq. (6) and M_{θ_P} be the poisoned model. The attack success rate (ASR) of a poisoning attack is defined as:

$$\mathcal{A}_P(M_{\theta_P}, \mathbf{t}, \mathcal{I}_\phi) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{G}_C}\ \mathbb{I}[M_{\theta_P}(\mathcal{I}_\phi(\mathbf{x}, \mathbf{t})) = \tilde{y}]. \tag{7}$$

To avoid trivial detection, the clean accuracy of a poisoned model (i.e., $\mathcal{A}_C(M_{\theta_P})$) must also be high. Optimizing for objective (6) ensures that the poisoned model has both high clean accuracy and high attack success rate. In the above formulation, the trigger $\mathbf{t}$ is assumed to be same for all samples (*sample-agnostic*) and all poisoned samples have the same target label $\tilde{y}$ (*all-to-one* attack). The trigger can also be learned for individual samples (*sample-specific*) and the target labels be different for each sample or class (*all-to-all* attack). *Clean-label* attacks that poison without modifying the labels of the training samples are also possible.

2.2. Related Work

While hidden backdoors (trojans) can be inserted into a ML model in many ways including direct modification of model parameters [8, 37] and incorporating additional malicious modules [26, 44], this work focuses only on poisoning attacks, which incorporate a backdoor into the model during training that is subsequently exploited by inserting a trigger into the input at prediction time. Key poisoning attacks and countermeasures against them are summarized below[1].

Poisoning Attacks on ML Models: Many poisoning attacks have been proposed by varying the nature of the trigger and the trigger insertion function. These include: *Modification* (**Modify**) - inserting small, visible, and static trigger patterns without covering the original input during both training and prediction (e.g., BadNets [19]); *Input-Aware Dynamic Attacks* (**IAD**) - inserting visible, but sample-specific trigger patterns [33]; *Blending* (**Blend**) - stealthily blending a trigger of the same size as the input with the original sample [9]; *Invisible Sample-Specific Backdoor Attack* (**ISS**) - learning to add imperceptible sample-specific triggers [27]; *Label-Consistent Backdoor Attack* (**LC**) - injecting inputs (through adversarial perturbations or generative models) that are plausible but hard to classify, without modifying the training labels [47]; *Hidden/Concealed Trigger Backdoor Attacks* (HT) - concealing the trigger during training, but revealing them only during prediction [38, 41, 48]; *Warping-based Backdoor Attack* (**WaNet**) - injecting imperceptible warping-based triggers in lieu of noise perturbations [34]; and *Sinusoidal Backdoor Signal* (**SIG**) - injecting sinusoidal triggers that are imperceptible and label consistent [1].

[1]The methods highlighted in bold are used in our experiments.

Poisoned Model Detection and Purification: Detection of poisoned models is typically achieved by constructing many benign and poisoned shadow models, querying these shadow models to extract discriminative features, and learning a meta-classifier based on these features. While *meta-neural trojan detection* (**MNTD**) [53] dynamically optimizes a query set along with the meta-classifier to distinguish between benign and poisoned models, *universal litmus patterns* (**ULP**) are learned in [24] to achieve the same objective. In [23], poisoned models are detected by analyzing the critical paths corresponding to different classes in a DNN model. Most of the purification methods rely on reverse engineering the trigger pattern and unlearning these patterns. Examples of such reverse engineering defense (RED) methods include: *Neural Cleanse* (**NC**) [49] - searching for possible triggers that can push all inputs towards the target class and unlearning such triggers; *DeepInspect* (DI) - learning a conditional generative model to generate potential triggers [7]; *anti-backdoor scanning* (ABS) [29] - identifying potentially compromised neurons based on their activation behavior; *FeatureRE* (**FRE**)[50] - using feature space constraint to reverse engineer the triggers; and *expected transferability* (**ET**) [52] - measuring the transferability of trigger patterns between samples of the same class.

Intersection between Adversarial and Poisoning Attacks: Recent works have studied the relationship between adversarial and poisoning attacks from multiple perspectives. One of the directions that has been explored is using adversarial samples as either triggers for backdoor attacks [55] or probes for detecting poisoned training samples [54]. Another interesting line of research is the connection between adversarial training and backdoor attacks [13–15, 32, 35, 45, 51]. For example, it has been demonstrated that using adversarial samples with larger perturbation budgets and "spatial" adversarial training can provide better defense against poisoning attacks [51]. While these studies are illuminating, none of them directly address the question of whether adversarial attacks can be used to detect poisoned models. Our proposed approach is closest in theory to *Cassandra* (**CAS**) [56] because it is based on a similar hypothesis that poisoned models are more sensitive to adversarial attacks. In CAS [56], universal adversarial perturbations [31] are used to probe the target model and a heuristic attack difficulty measure is used to detect poisoned models. It requires a large dataset of benign and poisoned models to learn the meta-classifier, which does not generalize well to stealthy attacks that are not present during training. In this work, we overcome the above limitations of CAS and present a simple and sound approach for detecting poisoned models based on their sensitivity to standard adversarial attacks.

3. Proposed Methodology

3.1. Problem Statement

Threat Model: In this work, we consider an adversary who aims to *stealthily* deploy a poisoned model. Here, stealthiness implies that the deployed model has good clean accuracy and behaves maliciously only on poisoned inputs. We assume that the adversary has full access to the training set $\mathcal{D}_T$ and trains the poisoned model based on objective function 6 (or a close variant of it). The adversary can use any arbitrary trigger insertion function $\mathcal{I}_\phi$, trigger pattern $\mathbf{t}$ (which could also be sample-specific), poisoning rate ρ.
Defender Goal: Given only *black-box* (query) access to a target ML model, denoted as M_{θ_t}, the goal of the defender is to determine if the given target model is benign or poisoned. This requires designing a poisoned model detector $\mathcal{T} : \mathcal{M} \rightarrow \{benign, trojan\}$, where $\mathcal{M}$ represents the model space. In this work, we design a trojan detector $\tilde{\mathcal{T}} : \mathcal{M} \rightarrow [0, 1]$ and assign the target model to the $trojan$ class if $\tilde{\mathcal{T}}(M_{\theta_t}) \geq \tau$, where τ is the decision threshold parameter.
Defender Capabilities: We assume that the defender has access to a small clean set $\mathcal{D}_C = \{(\mathbf{x}_k, y_k)\}_{k=1}^{N_C}$, where $N_C << N_T$. Note that $\mathcal{D}_C$ could either be a subset of $\mathcal{D}_T$ or an independent dataset sampled from $\mathcal{G}_C$. It must be emphasized that the defender has *zero knowledge* of the (i) target model architecture or its parameters (black-box scenario) and (ii) attack characteristics (trigger size/shape/location, trigger insertion function, poisoning rate, optimization algorithm/hyperparameters, etc.) employed by the adversary.

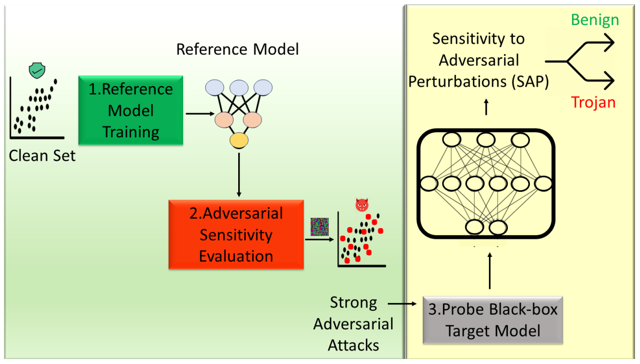
Reference Model
1.Reference Model Training
Clean Set
2.Adversarial Sensitivity Evaluation
Strong Adversarial Attacks
3.Probe Black-box Target Model
Sensitivity to Adversarial Perturbations (SAP)
Benign
Trojan

Figure 1. Overall workflow of Attack to Defend (A2D) framework.

3.2. Attack to Defend (A2D) Framework

The proposed attack-to-defend (A2D) framework consists of three main steps: (i) learning a reference model, (ii) generating strong adversarial samples that can circumvent the reference model, (iii) probing the target model based on the adversarial samples generated from the reference model to evaluate the adversarial sensitivity of the target model. These steps are depicted in Figure 1 and summarized in Alg. 1.

Algorithm 1: Attack To Defend (A2D) Framework

```
Input: Target model $M_{\theta_t}$, clean set $\mathcal{D}_C$, parameters:
  sample size $N_A$, margin $\omega$, threshold $\tau$
Output: Predicted label of $M_{\theta_t}$: $benign$ or $trojan$
1: Choose reference model architecture $\widehat{M}$
2: Train parameters $\theta_r$ of $\widehat{M}$ based on $\mathcal{D}_C$ using eq. (8)
3: Select $\mathcal{D}_A \subseteq \mathcal{D}_C$, such that $|\mathcal{D}_A| = N_A$
4: Estimate SAP $\hat{\mathcal{S}}(\widehat{M}_{\theta_r}, \epsilon)$ for different $\epsilon$ based on $\mathcal{D}_A$
5: $\epsilon_{min} \leftarrow \arg\min_{\epsilon} \hat{\mathcal{S}}(\widehat{M}_{\theta_r}, \epsilon) \geq (1 - \omega)$
6: Obtain attack set $\ddot{\mathcal{D}}_A$ with $\epsilon_{min}$ as perturbation bound
7: Evaluate $\hat{\mathcal{S}}^*(M_{\theta_t}, \epsilon_{min})$ based on $\mathcal{D}_A$ and $\ddot{\mathcal{D}}_A$
8: return $trojan$ if $\hat{\mathcal{S}}^*(M_{\theta_t}, \epsilon_{min}) \geq \tau$, else $benign$
```

1. Reference Model Learning: The defender first picks a reference model architecture $\widehat{M}$ (which need not be the same as the target model) and learns the reference model parameters (θ_r) by minimizing the following empirical risk based on the available clean set $\mathcal{D}_C$.

$$\theta_r = \arg\min_{\theta} \frac{1}{N_C} \sum_{k=1}^{N_C} \mathcal{L}(\widehat{M}_\theta(\mathbf{x}_k), y_k). \tag{8}$$

2. Adversarial Sensitivity Evaluation of Reference Model: In this step, the goal is to generate strong adversarial samples that are highly likely to fool the reference model $\widehat{M}_{\theta_r}$. The strength of an adversarial attack is primarily determined by the robust accuracy $\mathcal{A}_R(\cdot, \epsilon)$ of a model, which in turn depends on the perturbation bound ϵ. Strong adversarial samples should lead to low robust accuracy. However, robust accuracy alone is not sufficient to characterize the adversarial attack strength because it is easier to achieve a specific robust accuracy, if the initial clean accuracy of the model is low. Hence, we introduce a new metric to characterize the adversarial attack strength.

Sensitivity to Adversarial Perturbations (SAP) $\mathcal{S}$ of a ML model M_θ is defined as the relative reduction in accuracy caused by adversarial samples generated using a specific perturbation bound ϵ. Then,

$$\mathcal{S}(M_\theta, \epsilon) = \frac{\mathcal{A}_C(M_\theta) - \mathcal{A}_R(M_\theta, \epsilon)}{\mathcal{A}_C(M_\theta)}. \tag{9}$$

Intuitively, small values of ϵ will correspond to weaker adversarial samples that fail to mislead the model, resulting in SAP values closer to zero. In contrast, as $\epsilon \rightarrow 1$, the adversarial samples almost certainly fool the model leading to SAP values closer to one.

To generate strong adversarial samples against the reference model, we first choose a subset $\mathcal{D}_A$ of the clean set $\mathcal{D}_C$ containing N_A samples ($\mathcal{D}_A \subseteq \mathcal{D}_C$ and $N_A \leq N_C$) and estimate the clean accuracy of the reference model $\hat{\mathcal{A}}_C(\widehat{M}_{\theta_r})$.

Here, $\hat{\mathcal{A}}_C$ is an estimate of true clean accuracy $\mathcal{A}_C$ because the expectation in eq. (2) is replaced with the sample average computed over $\mathcal{D}_A$. For each input $\mathbf{x}_k \in \mathcal{D}_A$, we find the corresponding adversarial sample $\ddot{\mathbf{x}}_k = \mathbf{x}_k + \mathbf{r}_k$, where $\mathbf{r}_k$ is found using eq. (3) based on the reference model $\widehat{M}_{\theta_r}$ and perturbation bound ϵ. This process is repeated over a range of ϵ values, and for each ϵ, the corresponding robust accuracy $\hat{\mathcal{A}}_R(\widehat{M}_{\theta_r}, \epsilon)$ is estimated. Based on $\hat{\mathcal{A}}_C(\widehat{M}_{\theta_r})$ and $\hat{\mathcal{A}}_R(\widehat{M}_{\theta_r}, \epsilon)$, the $\hat{\mathcal{S}}(\widehat{M}_{\theta_r}, \epsilon)$ values are estimated using eq. (9). Next, we find the smallest value of ϵ for which the $\hat{\mathcal{S}}(\widehat{M}_{\theta_r}, \epsilon)$ value is very close to 1. Formally,

$$\epsilon_{min} = \arg\min_{\epsilon} \hat{\mathcal{S}}(\widehat{M}_{\theta_r}, \epsilon) \geq (1 - \omega), \tag{10}$$

where $\omega \to 0$ is a margin parameter. After finding ϵ_{min}, let $\ddot{\mathcal{D}}_A = \{(\ddot{\mathbf{x}}_k, y_k)\}_{k=1}^{N_A}$ denote the set of strong adversarial samples generated against the reference model $\widehat{M}_{\theta_r}$ with perturbation bound ϵ_{min}. Thus, at the end of this step, we have two sets $\mathcal{D}_A$ and $\ddot{\mathcal{D}}_A$, with the former consisting of clean samples and the latter containing adversarial samples that succeed in fooling the reference model. Note that the sample labels in both sets remain untouched. **3. Adversarial Probing of Target Model**: The given target model M_{θ_t} is probed (queried) using input samples from the clean ($\mathcal{D}_A$) and adversarial ($\ddot{\mathcal{D}}_A$) sets obtained from the previous step and the clean and robust accuracy values of the target model are estimated as follows:

$$\hat{\mathcal{A}}_C(M_{\theta_t}) = \frac{1}{N_A}\sum_{k=1}^{N_A} \mathbb{I}[M_{\theta_t}(\mathbf{x}_k) = y_k], \tag{11}$$

$$\hat{\mathcal{A}}^*_R(M_{\theta_t}, \epsilon_{min}) = \frac{1}{N_A}\sum_{k=1}^{N_A} \mathbb{I}[M_{\theta_t}(\ddot{\mathbf{x}}_k) = y_k]. \tag{12}$$

Here, the notation $*$ indicates that the robust accuracy of model M_{θ_t} has been estimated based on adversarial samples *transferred* from a different model $\widehat{M}_{\theta_r}$. Based on $\hat{\mathcal{A}}_C(M_{\theta_t})$ and $\hat{\mathcal{A}}^*_R(M_{\theta_t}, \epsilon_{min})$, the $\hat{\mathcal{S}}^*(M_{\theta_t}, \epsilon_{min})$ value of the target model is estimated using eq. (9). Since poisoned models are expected to have higher adversarial sensitivity than benign models, the estimated SAP value can be directly used as a measure for detecting poisoned models, i.e., $\tilde{\mathcal{T}}(M_{\theta_t}, \tau) = \hat{\mathcal{S}}^*(M_{\theta_t}, \epsilon_{min})$. The target model is detected as a $trojan$ if $\tilde{\mathcal{T}}(M_{\theta_t})$ is greater than the decision threshold τ. It is worth highlighting that the poisoned model detector $\tilde{\mathcal{T}}$ in the proposed A2D framework requires only black-box (query) access to the target model M_{θ_t} and a small clean set $\mathcal{D}_C$. More importantly, it does not require any knowledge of the target model or the attack characteristics. It is worth noting that the A2D framework can also be extended to the white-box setting as described in Appendix (A).

4. Experimental Results

We first summarize the implementation details before presenting the results of our evaluation.

4.1. Implementation Details

Datasets: We evaluate the performance of the A2D framework using three well-known image datasets: MNIST [12], CIFAR10 [25], and GTSRB (German Traffic Sign Recognition Benchmark) [42]. We also use a real-world chest X-ray dataset [36] for additional evaluation. More detailed information about the datasets is presented in Appendix (B.1). See Appendix (C.7) for evaluation on ImageNet [11].
Evaluation Metrics: Given an evaluation set of n target ML models $\{M_{\theta_j}, z_j\}_{i=1}^{n}$, where $z_j \in \{benign, trojan\}$ is the ground-truth label of the target models, the accuracy (ACC) of the poisoned model detector $\mathcal{T}$ is defined as:

$$\text{ACC}(\mathcal{T}, \tau) = \frac{1}{n}\sum_{j=1}^{n} \mathbb{I}[\mathcal{T}(M_{\theta_j}) = z_j], \tag{13}$$

where $\mathcal{T}$ assigns the target model to the $trojan$ class if $\tilde{\mathcal{T}}(M_{\theta_j}) \geq \tau$ and to the $benign$ class otherwise. The false detection rate (FDR) and true detection rate (TDR) of the detector can be computed as:

$$\begin{aligned}\text{FDR}(\mathcal{T}, \tau) &= \frac{1}{(n - n_p)} \sum_{\substack{j=1 \\ z_j = benign}}^{n} \mathbb{I}[\mathcal{T}(M_{\theta_j}) = trojan], \\ \text{TDR}(\mathcal{T}, \tau) &= \frac{1}{n_p} \sum_{\substack{j=1 \\ z_j = trojan}}^{n} \mathbb{I}[\mathcal{T}(M_{\theta_j}) = trojan], \end{aligned} \tag{14}$$

where n_p and $(n - n_p)$ are the numbers of poisoned and benign target models in the evaluation set, respectively. Finally, the receiver operating characteristic (ROC) curve of the detector $\mathcal{T}$ can be obtained by varying the threshold parameter τ and the area under the ROC curve (AUC) can be computed. For most experiments, five runs with different random seeds are used and the average metrics are reported. For ACC computation, we select the value of τ that minimizes the FDR. Typically, we observed that a value of τ between 0.6 and 0.7 is effective for all datasets when $\omega \leq 0.01$.
Poisoning Attacks: Most of our results are based on the following seven types of poisoning attacks: (i) Modify [19], (ii) Blend [9], (iii) WaNet [34], (iv) IAD [33], (v) LC [47], (vi) SIG [1], and (vii) ISS [27]. More details about the implementation of these attacks are included in Appendix (B.3). Note that the above seven poisoning methods cover a wide range of attack scenarios including sample-agnostic [1, 9, 19, 34, 47] vs. sample-specific [27, 33], label modification [9, 19, 27, 33, 34] vs. clean label [1, 47], and visible [9, 19] vs. imperceptible triggers [1, 27, 33, 34, 47]. Also, it must be emphasized that *none of these attacks* are known

to the defender apriori and are used while constructing the poisoned model detector.

Baseline Defenses: The core defense methods used for benchmarking the proposed detector are: (i) MNTD [53], (ii) ULP [24], (iii) CAS [56], (iv) NC [49], and (v) FRE [50]. Since FRE [50] is a very recent work that is clearly superior to four other REDs (namely, ABS [29], DI [7], TABOR [20], and K-arm [39]), we do not compare A2D against these four approaches (see Appendix (C.7) for comparison against ABS [29]). For the chest X-ray binary classification task, we also benchmark against ET [52]. While MNTD [53], ULP [24], and CAS [56] are detection-only methods, NC [49] and FRE [50] are purification methods. We evaluate only their detection components (see Appendix (B.4)).

Reference Model Details: Since the defender only has black-box access and has no knowledge of the target model architecture, we use ResNet-18 [21] as the default architecture for the reference model. For each dataset, the defender is assumed to possess 2% of the samples randomly selected from the original training dataset as the clean set $\mathcal{D}_C$, which is used to train the reference model from scratch. Details of the training hyperparameters and clean accuracy of the reference models are summarized in Appendix (B.2).

Target Model Architecture: The target models for MNIST use a 2CONV+2FC architecture (following [53]) for experiments with Modify and Blend attacks and a 3CONV+2FC architecture (following [34]) for WaNet and IAD attacks. For CIFAR10 and GTSRB datasets, target models are based on the PREACTRESNET18 architecture [22].

Training Target Models: 50 benign target models are trained for each dataset using the full training set based on eq. (1). It is assumed that the adversary has access to the full training set and trains poisoned models by corrupting a proportion ρ of the training samples, where ρ is uniformly sampled from the range $[0.05, 0.5]$. The adversary can employ any of the seven poisoning attacks listed earlier to create the poisoned models. For evaluation, we use 50 models each of Modify and Blend attacks, and 20 models each of WaNet, IAD, LC, SIG, and ISS attacks per dataset. Details of the training hyperparameters and clean accuracy of the target models are summarized in Appendix (B.2).

Adversarial Attacks: We employ Projected Gradient Descent (PGD) [30] as our main adversarial attack to evaluate the adversarial sensitivity of the reference model and generate the attack set $\ddot{\mathcal{D}}_A$. Results on the Fast Gradient Sign Method (FGSM) [18] are also reported in Appendix (C.1).

4.2. Benchmarking of Detection Accuracy

Performance comparison of the A2D approach against other state-of-the-art (SOTA) poisoned model detectors is summarized in Table 1. The results clearly show that the proposed method outperforms the SOTA by a significant margin on almost all the attacks across the three datasets. Some key insights from Table 1 are as follows:

(**1**) While many methods achieve good detection accuracy for the simple Modify and Blend attacks, their performance drops substantially when encountering more sophisticated attacks such as WaNet, IAD, LC, SIG, and ISS, highlighting their poor generalizability to new/unseen attacks. The proposed approach generalizes well across attacks, though it has marginally low detection accuracy than SOTA for Modify and Blend attacks in GTSRB and CIFAR-10, respectively.

(**2**) It must be emphasized that the low detection accuracy (50%) of the baseline methods for some attacks (e.g., NC on LC and ISS attacks in the CIFAR-10 dataset, FRE on LC and SIG attacks on GTSRB, etc.) is primarily due to their inability to detect poisoned models (all the target models in the evaluation set are categorized as $benign$, leading to low TDR). This demonstrates the difficulty in detecting invisible and clean label attacks and overcoming this challenge is the main strength of the proposed A2D framework. Furthermore, since NC and FRE are purification methods, failure to detect a poisoning attack will result in the model evading purification and remaining poisoned. This is worse than incorrectly flagging a few benign models as $trojan$ (degrading their clean accuracy) because it may give a false sense of security.

(**3**) The performance metrics of many baseline methods vary widely across datasets - for example, while NC and FRE methods achieve high accuracy against ISS attacks on the GTSRB dataset, they have very poor detection accuracy for the same attack on the CIFAR-10 dataset. This may be attributed to their high sensitivity to hyperparameters. Since we attempt to retain the same hyperparameters across datasets, settings that are optimal for one dataset may not work for a different dataset. The consistent performance of the A2D framework across datasets is a testimony to its lower sensitivity to hyperparameter values.

(**4**) The good generalizability of the A2D method in comparison to CAS [56] (which is also based on the same core hypothesis) can be attributed to the fact that there is no meta-classifier training phase involved and no specific poisoning attacks are used anywhere in the proposed detection process.

To assess the robustness of our method on real-world datasets, we conducted additional experiments on the chest X-ray dataset involving a binary classification task (tuberculosis prediction). Since many of the REDs cannot be applied for binary classifiers, we compare A2D with ET [52] on this dataset. Table 2 provides a summary of the detection accuracy of our method compared to ET on Blend, WaNet, and IAD attacks. It is evident that our method outperforms ET, especially in detecting stealthy attacks where the features of the image and the trigger are intricately mixed.

4.3. Why A2D Framework Works?

Recall that the core hypothesis underlying the A2D framework is that poisoned models are more sensitive to adversar-

Table 1. Benchmarking of different state-of-the-art poisoned model detection methods. The metrics in this Table are averaged over 5 runs.

Metric (%)	**Method**	**MNIST**				**CIFAR10**							**GTSRB**						
		Modify	Blend	WaNet	IAD	Modify	Blend	WaNet	IAD	LC	SIG	ISS	Modify	Blend	WaNet	IAD	LC	SIG	ISS
ACC(↑)	**MNTD [53]**	56.6	54.4	57	59.3	68.9	68.7	75	64.3	57.2	62.5	71	60	55.5	<50	50.2	50	62.5	55
	ULP [24]	63	77	<50	82.5	62.3	59.2	52.5	<50	69.4	50	53	54	58	<50	<50	60	50	50
	NC [49]	80	60	57.5	65	74	66	70	70	50	90	50	93	95	75	67.5	50	55	95
	FRE [50]	-	-	-	-	88	**90**	75	80	50	55	60	90	90	86	65	50	50	**100**
	CAS [56]	94	74	87.5	70	91	85	57.5	62	85	60	55	**99**	97	92.5	50	50	50	50
	A2D (Ours)	**94.6**	**99.6**	**100**	**100**	**93**	86.6	**100**	**95**	**95**	**100**	**100**	96.8	**97.4**	**100**	**100**	**95**	**95.5**	**100**
TDR(↑)	A2D (Ours)	89.6	99.6	100	100	92.8	80	100	90	90	100	100	98.8	100	100	100	90	91	100
FDR(↓)	A2D (Ours)	0.4	0.4	0	0	6.8	6.8	0	0	0	0	0	5.2	5.2	0	0	0	0	0
AUC(↑)	A2D (Ours)	95.7	100	100	96.3	89.6	87.7	96.9	86.2	100	100	100	96.4	96	99.9	100	97.9	95.1	100

ial attacks compared to benign models. Independent works in [35, 56] have also hinted at the same possibility, without offering any theoretical proofs. It has been argued that poisoning can alter the model's decision boundary, making it easier for an adversary to find points that are close to the boundary but still classified incorrectly by the model [56]. To verify this hypothesis, we conducted the following experiment. We observe the behavior of poisoned models with modification attack when fed with a clean sample $(\mathbf{x})$ and its corresponding poisoned counterpart $(\tilde{\mathbf{x}} = \mathcal{I}_\phi(\mathbf{x}, \mathbf{t}))$. Let $a(\mathbf{x})$ and $a(\tilde{\mathbf{x}})$ denote the neuron activations based on the clean and poisoned samples, respectively. The set of neurons corresponding to the top-K activation differences between $a(\mathbf{x})$ and $a(\tilde{\mathbf{x}})$ (we set $K = 10000$), denoted as $bp(\mathbf{x}, \tilde{\mathbf{x}})$, can be considered as the backdoor pathway created via poisoning attack.

Next, we adversarially attack the poisoned model based on the same input $\mathbf{x}$ to create the adversarial sample $\ddot{\mathbf{x}}$. Let $a(\ddot{\mathbf{x}})$ be the neuron activations for the adversarial sample and $ap(\mathbf{x}, \ddot{\mathbf{x}})$ be the top-K activation differences between $a(\mathbf{x})$ and $a(\ddot{\mathbf{x}})$, which can be considered as the pathway used by the adversarial attack. We then compute the *intersection-over-union* (IOU) metric between the two sets of neurons in $bp(\mathbf{x}, \tilde{\mathbf{x}})$ and $ap(\mathbf{x}, \ddot{\mathbf{x}})$. A higher IOU value is an indication that the adversarial attack is leveraging the same pathway (neurons) inserted into the model by the poisoning attack. We repeat the same analysis based on benign models and compute the corresponding IOU values. Figure 2 shows the density estimate of the IOU values computed using benign and poisoned models. Clearly, the poisoned models tend to have higher IOU values than benign models, which demonstrates that the backdoor inserted by the poisoning attack indeed creates an *easy* pathway through the network that can be exploited by the adversarial attack to find adversarial samples. While this trend is consistent across different types of model architectures and poisoning attacks, the degree of separation between the benign and trojan IOU distributions varies depending on the specific architecture and poisoning attack employed. This experiment validates our hypothesis about the higher adversarial vulnerability of poisoned models and explains the good performance of the A2D framework.

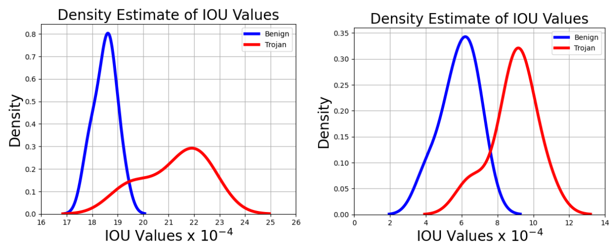

Figure 2. The density estimate of the IOU values computed using benign and poisoned models on CIFAR10(left) and GTSRB(right).

4.4. Sensitivity Analysis

The key choices that can potentially impact the effectiveness of the A2D framework are: (i) reference model architecture $\widehat{M}$, (ii) adversarial attack type, (iii) size and quality of the clean set $\mathcal{D}_C$ and the number of samples N_A used for adversarial sensitivity evaluation, and (iv) trigger properties.

Impact of Reference Model Architecture: To evaluate the impact of reference model architecture, we consider three architectures - ResNet18 [21], PREACTRESNET18 [22] and VGG16 [40], and use all possible combinations of these architectures for target and reference models. Interestingly, our findings in Table 3 indicate that the architecture of the reference model itself does not significantly affect the detection accuracy. While the detection accuracy is marginally higher in cases where the reference model architecture is the same as the target model architecture (gray-box setting), good detection accuracy can also be achieved in the black-box scenario when the two architectures are different. This implies that the reference model architecture plays a minor role and the data distribution is the main factor determining the adversarial sensitivity of models. While these experiments indicate good generalizability across a family of convolutional neural network models, more experiments with other models such as vision transformer are required to confirm the black-box capabilities of the A2D approach.

Metric (%)	Method	Chest X-ray		
		Blend	WaNet	IAD
ACC(↑)	ET [52]	75	50	50
	A2D (Ours)	80	87.5	100

Table 2. Benchmarking against ET [52] on chest X-ray dataset.

Table 3. Average detection accuracy (ACC %) comparison for different model architectures.

Dataset	Reference Model	Target Model		
		PreActResNet	ResNet18	VGG16
CIFAR10	PreActResNet	**95.66**	92.79	96.07
	ResNet18	95.06	**92.86**	96.07
	VGG16	95.06	97.36	**97.93**
GTSRB	PreActResNet	**98.53**	98.86	98.14
	ResNet18	97.91	**99.71**	98.43
	VGG16	98.29	**100.00**	98.21

Impact of Sample Sizes: We investigated the impact of various sample size parameters on the A2D method and the detailed results are presented in Appendix (C.2). We observed that changing the size of the clean set from 2% to 5% of the training set had little impact on the accuracy and ϵ_{min} values. Using a clean set drawn from the same distribution $\mathcal{G}_C$ but is independent of the training set had only a marginal negative impact on the detection accuracy for both datasets - detailed results are in Appendix (C.4). However, the number of samples used for the adversarial attack did have some impact on the detection accuracy. Specifically, a minimum of 100 attack samples is required for reliable poisoned model detection. Furthermore, the margin parameter ω should be less than 0.1 (i.e., $\hat{\mathcal{S}}(\widehat{M}_{\theta_r}, \epsilon) \geq 0.9$) for achieving high accuracy. This is because only strong attacks against the reference model transfer well to the target model, as shown in Appendix (C.1). **Impact of Trigger Properties**: We evaluated the effect of *trigger transparency* (visibility) and *spatial size of the trigger* on detection accuracy. The results in Appendix (C.3) show that the poisoned models are detectable irrespective of the trigger size and visibility.

4.5. Robustness and Utility of A2D Framework

Adaptive Attack: If the attacker knows that a specific defense mechanism is employed, can an additional objective be added to the poisoning attack to achieve high ACC and ASR as well as fool the defense mechanism? In this experiment, we investigated whether the adversary can generate a poisoned model that is also adversarially robust. This can be realized by either starting with an adversarially trained model (see Appendix (C.5) for results) or by incorporating an additional adversarial loss term during training (to reduce the adversarial sensitivity of the model) in addition to minimizing the loss of clean and poisoned samples. The results of this strong adaptive attack are depicted in Figure 3. This shows the infeasibility of increasing vulnerability to poisoning attacks while maintaining a low adversarial sensitivity. Attempting to minimize the three losses (clean, poison, and adversarial) together leads to a substantial drop in ASR. This is due to the conflicting objectives of poisoning attack and adversarial training. While poisoning aims to push the poisoned sample towards the target class, adversarial training aims to push the perturbed sample back to the source class.

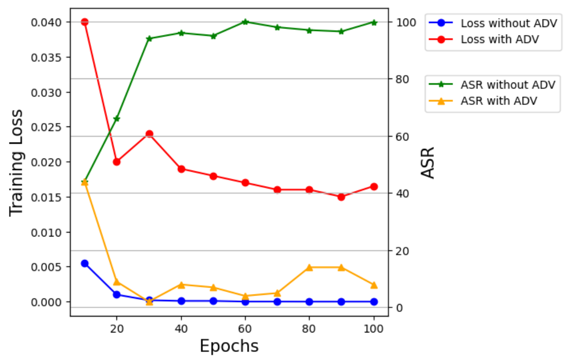

Figure 3. Training loss (left y-axis) and attack success rate (right y-axis) for an adaptive attack.

Using A2D in conjunction with NC: It must be emphasized that the A2D approach only detects poisoned models. However, it can be easily integrated with other purification methods such as NC or FRE, which often have a low TDR. For example, in NC [49], the detection step based on anomaly index can be replaced using the A2D method. If a model is flagged as $trojan$ by our method, we can utilize the reversed trigger in NC with the highest anomaly index and apply the NC patching (unlearning) method. To validate this approach, we identified 9 poisoned models on the CIFAR10 dataset, employing PREACTResNet-18, ResNet, and VGG architectures along with SIG, LC, and ISS attacks. Originally, NC classified these models as benign and failed to purify them, while our A2D method correctly identified them as $trojan$. Results in Appendix (C.6) illustrate the average clean accuracy and ASR of these models before and after cleansing. Although the "cleansed" models experienced $\approx 10\%$ drop in clean accuracy, the ASR dropped from over 90% to less than 10%. This proves the utility of our approach, especially when the option to reject a model is not feasible.

5. Conclusions

We have presented the (A2D) approach for detecting poisoned DNN models, which leverages the increased vulnerability of poisoned models to adversarial attacks. We have demonstrated the generalizability of our method across datasets, attacks, and models, especially in the black-box scenario. **Limitations**: As with majority of existing methods, our approach also needs a limited set of clean samples.

References

[1] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE, 2019.

[2] Chandradeep Bhatt, Indrajeet Kumar, V Vijayakumar, Kamred Udham Singh, and Abhishek Kumar. The state of the art of deep learning models in medical science and their challenges. *Multimedia Systems*, 27(4):599–613, 2021.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[4] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 39–57. IEEE, 2017.

[6] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *Workshop on Artificial Intelligence Safety*. CEUR-WS, 2019.

[7] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, page 8, 2019.

[8] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Proflip: Targeted trojan attack with progressive bit flips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7718–7727, 2021.

[9] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[10] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, pages 2206–2216. PMLR, 2020.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 248–255. IEEE, 2009.

[12] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[13] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojciech Czaja, and Tom Goldstein. Adversarial examples make strong poisons. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:30339–30351, 2021.

[14] Yinghua Gao, Dongxian Wu, Jingfeng Zhang, Guanhao Gan, Shu-Tao Xia, Gang Niu, and Masashi Sugiyama. On the effectiveness of adversarial training against backdoor attacks. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2023.

[15] Jonas Geiping, Liam Fowl, Gowthami Somepalli, Micah Goldblum, Michael Moeller, and Tom Goldstein. What doesn't kill you makes you robust (er): How to adversarially train against data poisoning. *arXiv preprint arXiv:2102.13624*, 2021.

[16] Hamed Ghoddusi, Germán G Creamer, and Nima Rafizadeh. Machine learning in energy economics and finance: A review. *Energy Economics*, 81:709–727, 2019.

[17] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[19] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[20] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR)*, pages 770–778, 2016.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645. Springer, 2016.

[23] Wei Jiang, Xiangyu Wen, Jinyu Zhan, Xupeng Wang, Ziwei Song, and Chen Bian. Critical path-based backdoor detection for deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2022.

[24] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *IEEE /CVF Computer Vision and Pattern Recognition Conference (CVPR)*, pages 301–310, 2020.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[26] Yuanchun Li, Jiayi Hua, Haoyu Wang, Chunyang Chen, and Yunxin Liu. Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 263–274. IEEE, 2021.

[27] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *In IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16463–16472, 2021.

[28] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2022.

[29] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural

networks for back-doors by artificial brain stimulation. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019.

[30] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *IEEE /CVF Computer Vision and Pattern Recognition Conference (CVPR)*, pages 1765–1773, 2017.

[32] Bingxu Mu, Zhenxing Niu, Le Wang, Xue Wang, Qiguang Miao, Rong Jin, and Gang Hua. Progressive backdoor erasing via connecting backdoor and adversarial attacks. In *IEEE /CVF Computer Vision and Pattern Recognition Conference (CVPR)*, pages 20495–20503, 2023.

[33] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:3454–3464, 2020.

[34] Tuan Anh Nguyen and Anh Tuan Tran. Wanet - imperceptible warping-based backdoor attack. In *International Conference on Learning Representations (ICLR)*, 2021.

[35] Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. A tale of evil twins: Adversarial inputs versus poisoned models. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 85–99, 2020.

[36] Tawsifur Rahman, Amith Khandakar, Muhammad Abdul Kadir, Khandaker Rejaul Islam, Khandakar F Islam, Rashid Mazhar, Tahir Hamid, Mohammad Tariqul Islam, Saad Kashem, Zaid Bin Mahbub, et al. Reliable tuberculosis detection using chest x-ray with deep learning, segmentation and visualization. *IEEE Access*, 8:191586–191601, 2020.

[37] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, pages 13198–13207, 2020.

[38] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *AAAI Conference on Artificial Intelligence*, pages 11957–11965, 2020.

[39] Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization. In *International Conference on Machine Learning (ICML)*, pages 9525–9536. PMLR, 2021.

[40] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[41] Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:19165–19178, 2022.

[42] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1453–1460. IEEE, 2011.

[43] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE /CVF Computer Vision and Pattern Recognition Conference (CVPR)*, pages 1701–1708, 2014.

[44] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. An embarrassingly simple approach for trojan attack in deep neural networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 218–228, 2020.

[45] Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Better safe than sorry: Preventing delusive adversaries with adversarial training. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:16209–16225, 2021.

[46] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[47] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[48] Eric Wallace, Tony Z Zhao, Shi Feng, and Sameer Singh. Customizing triggers with concealed data poisoning. *arXiv preprint arXiv:2010.12563*, 2020.

[49] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 707–723. IEEE, 2019.

[50] Zhenting Wang, Kai Mei, Hailun Ding, Juan Zhai, and Shiqing Ma. Rethinking the reverse-engineering of trojan triggers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[51] Cheng-Hsin Weng, Yan-Ting Lee, and Shan-Hung Brandon Wu. On the trade-off between adversarial and backdoor robustness. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:11973–11983, 2020.

[52] Zhen Xiang, David Miller, and George Kesidis. Post-training detection of backdoor attacks for two-class and multi-attack scenarios. In *International Conference on Learning Representations (ICLR)*, 2022.

[53] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *IEEE Symposium on Security and Privacy (S&P)*, pages 103–120. IEEE, 2021.

[54] Mingfu Xue, Yinghao Wu, Zhiyu Wu, Yushu Zhang, Jian Wang, and Weiqiang Liu. Detecting backdoor in deep neural networks via intentional adversarial perturbations. *Information Sciences*, 634:564–577, 2023.

[55] Quan Zhang, Yifeng Ding, Yongqiang Tian, Jianmin Guo, Min Yuan, and Yu Jiang. Advdoor: adversarial backdoor attack of deep learning system. In *ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 127–138, 2021.

[56] Xiaoyu Zhang, Rohit Gupta, Ajmal Mian, Nazanin Rahnavard, and Mubarak Shah. Cassandra: Detecting trojaned networks from adversarial perturbations. *IEEE Access*, 9: 135856–135867, 2021.