Toward a Critical Evaluation of Robustness for Deep Learning Backdoor Countermeasures

Huming Qiu, Hua Ma, Zhi Zhang, Alsharif Abuadbba, Wei Kang, Anmin Fu, and Yansong Gao, Senior Member, IEEE

Abstract—Since Deep Learning (DL) backdoor attacks have been revealed as one of the most insidious adversarial attacks, a number of countermeasures have been developed with certain assumptions defined in their respective threat models. However, their robustness is currently inadvertently ignored, which can introduce severe consequences, e.g., a countermeasure can be misused and result in a false implication of backdoor detection. For the first time, we critically examine the robustness of existing backdoor countermeasures. As an initial study, we first identify five potential non-robust failure factors including binary classification, poison rate, model complexity, single-model justification, and hyperparameter sensitivity. As exhaustively examining defenses is infeasible, we instead focus on influential backdoor detection-based countermeasures consisting of model-inspection ones including Neural Cleanse (S&P’19), ABS (CCS’19), and MNTD (S&P’21), and data-inspection ones including SCAN (USENIX SECURITY’21) to examine their failure cases under one or more of these factors. Although these investigated countermeasures claim that they work well under their respective threat models, they have inherent unexplored non-robust cases, which are not even rooted from delicate adaptive attacks. We demonstrate how to trivially bypass them aligned with their respective threat models by simply varying the aforementioned factors. Particularly, for each defense, formal proofs or empirical studies are used to reveal its non-robust cases where it is not as robust as it claims or expects. This work highlights the necessity of thoroughly evaluating the robustness of backdoor countermeasures to avoid their misleading security implications in unknown non-robust cases.

Index Terms—Deep learning, robustness, backdoor countermeasure, failure factor.

I. INTRODUCTION

Since backdoor attacks first appeared in 2017 [1], [2], they have posed a serious challenge to Deep Learning (DL)-enabled security-sensitive applications [3] such as face recognition [4], self-driving [5], and medical diagnosis [6]. To mitigate them, significant efforts have been spent in proposing countermeasures, which can be divided into two categories [7] that are data inspection and model inspection, and each inspection can be done offline or online. A data-inspection approach audits every input sample either offline [8] or online [9], [10], [11] and decides whether a given sample is malicious (i.e., containing a trigger) or benign. For the offline data-inspection, the malicious samples are removed from the training set to prevent backdoor insertion. A model-inspection method is usually performed offline [12], [13], [14], justifying whether a given model is backdoored or not. If yes, a model user either rejects the model before its deployment or fixes it by removing its backdoor effect.

A. Limitations of State-of-The-Arts

Under predefined threat models, the backdoor defenses can be effective against targeted backdoor types or trigger types (refer to Section II-B). For example, some model-inspection countermeasures assume that a trigger for a backdoor attack is not large and thus they are effective against common source-agnostic backdoor attacks, where an input sample that is embedded with a small trigger (e.g., a fixed patch) can hijack an infected model. Yet, adaptive attacks (e.g., delicately crafted regularization is applied to backdoor a model) can defeat such countermeasures, which are regarded to be challenging to address in the arms-race of DL security [7], [8], [15], [16], [17], [18], [19].

However, we observe that robustness of existing backdoor defense approaches, especially those detection-based ones, has never been critically investigated, e.g., whether the effectiveness of a given defense is dependent on datasets, specific tasks, model architectures, model hyperparameters, etc.. When the defense is used in a non-robust case, it is likely to fail in detecting a backdoor attack even under its predefined threat model (e.g., assuming a source-agnostic backdoor with a small fixed trigger), which is not even caused by adaptive attacks.

To this end, we ask the following questions:

Is a given (influential) backdoor countermeasure (in particular, detection-based defenses) robust enough in different scenarios within its well-predefined threat model? If not, what factors make it non-robust and to which extent render its ineffective in backdoor detection?
The questions above are critical for a user as the user can misuse an influential countermeasure without fully understanding its robustness, resulting in a false security indication of the backdoor detection. That is, even if the countermeasure claims that a model is backdoor-free within its threat model, it does not mean that the model is not backdoored with the defense-targeted trigger or/and backdoor type.

B. Our Work and Contributions

This work is, for the first time, towards revealing the non-robust cases of existing backdoor countermeasures. As the methodology behind each backdoor detection can vary greatly and each defense can have different threat models, it is challenging to design a unified framework for evaluations. We, therefore, start by identifying potentially common non-robustness failure factors before examining a given defense. This work initially investigates five such failure factors: binary classification, poison rate, model complexity, single-model justification, and hyperparameter sensitivity (detailed and reasoned in Section III-B).

In addition, as there are many backdoor defenses since 2018 (backdoor attack was revealed in 2017 [1, 2]), it is infeasible to exhaustively evaluate them all against the aforementioned failure factors. Defenses chosen by our initial investigations mainly follow two principles: offline detection-based countermeasures; and high visibility in the community. Note we focus on model-based detection works, despite we also evaluate an influential data-based detection. The main reason is that these model-based detection defenses have attractive advantages of requiring no access to training data with only one-off offline cost—practical in major application scenarios such as model training outsourcing. For high visibility, we mainly chose defenses that appeared in big4 that are the flagship conferences with very high impact and visibility. In this context, we note that there are three such offline model-based detection countermeasures, which are Neural Cleanse (Oakland 19') [20], ABS (CCS 19') [13], and MNTD (Oakland 21') [14], and one offline data-based detection of SCAn (USENIX SECURITY '21) [8]. Note that offline data-based detection requires access to training data that consists of poisoned samples resulting in backdoor insertion, despite it is also done one-time during offline. Moreover, we have considered other defenses such as Trojan Signature [21] and ULP [22] for the purpose of validating the conjecture that detection works sharing some characteristics (i.e., methods or concepts) may fail to the same failure factor. These critically examined defenses are summarized in Table I. To this end, we throw unknown security caveats of these countermeasures, which indicate that their robustness must be thoroughly examined to avoid any misleading usage.

Specifically, we reevaluate the robustness of these backdoor detection countermeasures. For each defense, we have revealed one or more non-robust cases under one or more of the initially identified five failure factors through either formal proofs or empirical case studies. Notably, our reevaluation settings are well aligned with its predefined threat model. That is, the non-robust cases are not rooted in delicate adaptive attacks but in unexplored failure factors that are common in practice. This work highlights the importance and emergence of examining the robustness of existing backdoor countermeasures, particularly influential ones before deployment in the real-world.

The main contributions and results are summarized below—our evaluations are based on reproducing the source code of each original work.

- We are the first to investigate the robustness of backdoor countermeasures with an initial concentration on influential offline detection based ones, which are (inadvertently) neglected by the community but are imperative for backdoor detection.
- We, as an initial investigation, identify and reason five potentially common non-robustness failure factors consisting of binary classification, poison rate, model complexity, single-model justification, and hyperparameter sensitivity.
- We, through extensive experiments, reveal one or more non-robust cases of each detection countermeasure against one or more of the five identified failure factors, as summarized in Table I. The reasons behind these non-robust cases are further analyzed.
- For these examined defenses, the MNTD appears to be the least robust and most computing-intensive. The Neural Cleanse is the most robust in that it can work in common cases under its threat model. We also find that countermeasures (i.e., Trojan Signature and Neural Cleanse; ULP and MNTD) sharing some similarities (i.e., methods or concepts) may fail to the same non-robust factor.

The rest of paper is organized as follows. Section II provides related works on backdoor attacks and major differences between adaptive attacks and non-robust cases of the countermeasures. Section III describes the experimental setup, followed by identifying and reasoning five potentially common non-robust failure factors. Section IV introduces the methodology of each chosen countermeasure, and then reveals and analyzes one or more non-robust cases under one or more identified failure factors. We conclude this work in Section V with a call on evaluating the robustness of other influential DL backdoor countermeasures in the future.

II. RELATED WORKS

A. Backdoor Attack

Backdoor attacks against DL were first highlighted in 2017 [1, 2]. Since then, it has received extreme attention from both academics and industry due to its potentially disastrous consequences and realistic attack scenarios [5]. The attack can stealthily result from data collection, model training outsourcing, and collaborative learning (e.g., federated

| Backdoor Countermeasure | Non-robust Factor(s) |
|-------------------------|----------------------|
| ABS                     | Poison Rate          |
| MNTD                    | Model Complexity, Hyperparameter Sensitivity |
| Neural Cleanse [20]     | Model Complexity, Binary Classification |
| SCAn [8]                | Poison Rate, Binary Classification |
| Trojan Signature [22]   | Binary Classification |
| ULP [21]                | Model Complexity, Hyperparameter Sensitivity |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
learning) [7], [23]. Considering the severe consequences of backdoor attacks, great efforts [24] have been devoted to detecting or eliminating backdoors both from either offline inspection [8], [20], [25], [26], [27] or online inspection [9], [13], [28]. There are two commonly used metrics to evaluate the performance of backdoored models: clean data accuracy (CDA) and attack success rate (ASR). The CDA refers to the percentage of clean test samples without triggers that are correctly predicted to be their ground truth labels. The ASR refers to the percentage of infected test samples containing the attacker-chosen trigger that are predicted to be the attacker-targeted label.

B. Backdoor and Trigger Types

1) Backdoor Type: Backdoor attacks have different backdoor types and most existing defenses focus on defending against source-agnostic backdoor attacks. In such an attack, any input regardless of its source class containing the trigger will fire the backdoor inserted in an infected model. There are other backdoor types and the representative one is source-specific backdoor attack. The backdoor is activated not only when the trigger is embedded within the input but also when input is selected from attacker-chosen source classes. If the input is from a non-source class, the backdoor does not exhibit even though the input is with the trigger. The source-specific backdoor attack is challenging [8], [29] to defeat as most countermeasures including [9], [13], [20] are ineffective against it.

2) Trigger Type: Most defenses have relatively strong assumptions on the trigger types, such as size and pattern, which may not be held in practice. For example, the trigger size could be overlaid with the entire image input with certain transparency, which trivially makes many defenses ineffective [20], [28]. Instead of targeting a label with a single trigger, multiple triggers can be used to target the same label, which is beyond the assumption of [13] that cannot be detected. In addition, Neural Cleanse is less effective when the trigger pattern becomes complicated [30] or network is deeper [31] or belongs to the feature-space based trigger attack [13]. Nonetheless, to explicitly demonstrate the non-robust cases of existing countermeasures, the backdoor type (i.e. source-agnostic backdoor) and trigger type (a small fixed patch trigger) used in this work are exactly within the threat model of each countermeasure. We do not consider any backdoor or trigger out the scope of the countermeasures in this work.

C. Adaptive Attack

In the area of AI security race such as adversarial example attack, model stealing attack, and this newly backdoor attack, it is unsurprising that the adaptive attack is extremely challenge to be defeated [7], [32]. The adaptive means that the attacker has knowledge of the defense strategy, thus developing more intelligent attacks to bypass it. In the adversarial example attack, it has been always shown that the new defenses are venerable to adaptive attacks [15], [16], similarly in backdoor defenses [18], [33]. For instance, Shokri et al. [18] devise an adaptive adversarial training algorithm to optimize the original loss function of the model, and also maximizes the indistinguishability of the latent representations of trigger input and benign input to bypass backdoor detections. For adaptive attacks, once it is developed, it’s effectiveness is independent of e.g., dataset, task when being incorporated to bypass the defenses.

This work focuses on the non-robust cases of the backdoor countermeasures, the non-robust cases raise mainly because of the countermeasure’s unknown sensitivity to some common factors e.g., poison rate and model complexity. It is not delicately designed adaptive attacks.

III. EXPERIMENTAL SETUP AND FAILURE FACTOR

We first describe the experimental setup. We then identify five non-robust failure factors against which backdoor countermeasures evaluate, detailed in Section IV.

A. Experimental Setup

Note that we follow the same datasets or/and models that were used by the evaluated backdoor countermeasures (e.g., Neural Cleanse, MNTD, ABS) whenever possible. This ensures that their failure cases are dominantly introduced by our initial focused five non-robust factors. We leave more thorough evaluations on diverse datasets and model architectures that these countermeasures might be non-robust in future work.

1) Dataset: MNIST: is a dataset consisting of 10 categories of handwritten digits provided by different people. It has 60,000 training images and 10,000 test images, both of which are 28 × 28 × 1 grayscale images. The MNIST was used in Neural Cleanse, MNTD and ULP.

CIFAR10: is a natural color image dataset for object recognition. It consists of 10 categories, and each category has 6,000 32 × 32 × 3 RGB images—60,000 in total. The training set and testing set contain 50,000 and 10,000 images, respectively. The CIFAR10 was used in ABS, MNTD and ULP.

GTSRB: is a dataset for German traffic sign recognition (GTSRB), which contains 43 types of traffic signs. In the data preprocessing stage, all traffic signs are cut out from the image according to the bounding box coordinates and aligned uniformly into a 32 × 32 × 3 RGB image—the original size of the image varies. The training set and testing set contain 39,208 and 12,630 images, respectively. The GTSRB was used in Neural Cleanse, ABS, SCAN, ULP, and Trojan Signature.

STL10: is an image recognition dataset with 10 classes. The training and testing sets contain 5,000 and 8,000 96 × 96 × 3 RGB images, respectively. All images in them are acquired from labeled examples of ImageNet to serve as a more complex dataset than the previous three datasets in our evaluations.

2) Trigger: The trigger of the CIFAR10 is set to be the white square at the bottom right of the image. The trigger of GTSRB is set to be the white triangle at the bottom left corner. The trigger of STL10 is set to be the white diamond at the top right corner. For three datasets, the first category is selected as the target category that means any input stamped with the trigger will be classified into the first category by the backdoored model—so that we consider the source-agnostic backdoor type that is under threat model of each countermeasure. It is worth noting that the size of the
three triggers accounts for merely 3.5% of the whole image, which is again within the effective detection range of the countermeasures. One or more of these three datasets along with the specified triggers are used to evaluate the Neural Cleanse, ABS, and SCAn.

As for the MNTD, we use CIFAR10 and MNIST for evaluation. The trigger pattern and target category are randomly generated from the general attack setting distribution modeled by jumbo learning according to the source code of MNTD [14]. In addition, for the ULP we evaluate it using CIFAR10, which uses multiple hand-crafted trigger patterns to train the backdoored model set.

3) Machine Configuration: Our test machine is MECHREVO with NVIDIA GeForce RTX 3070 GPU (8 GB video memory), Intel i7-11800H CPU (16 logical cores) and 16 GB DRAM memory. All experiments including model training and defense evaluation are done on this machine.

B. Non-Robust Failure Factor

Considering the fact that diverse assumptions (i.e., threat model) followed by each backdoor countermeasure and different methodologies adopted, it is challenging to develop a unified experimental framework when examining diverged backdoor countermeasures. As an initial study towards evaluating the robustness of these countermeasures, it is worth identifying potential factors that can result in non-robust failures. While exhaustively examining these factors appears to be infeasible, we empirically investigate five such potential common factors: binary classification, poison rate, single-model justification, hyperparameter sensitivity, and model complexity, which are elaborated below. More failure factors identification provides interesting future works. We conjecture that the if behind concepts or used methodologies share similarities for backdoor defenses, they may have similar non-robustness cases confronted with the same failure factor, which are affirmed to a large extent in Section IV.

1) Binary Classification: There are many security-sensitive applications (i.e., spamming email detection, malware detection, fraud detection) that are binary-classification tasks. Therefore, it is important that the defense is generalized to binary classification tasks. However, defenses requiring iterating all classes (i.e., Neural Cleanse, Trojan signature [22], B3D [34]) to identify backdoor behavior through anomaly detection methods may be inapplicable in the binary classification task. In these defenses, one value is generated per class by the defense method often to implement an anomaly detection strategy for backdoor detection. Since when the number of classes becomes small, especially with a size of two (i.e., only two classes), anomaly detection becomes hard or unstable.

2) Poison Rate: The poison rate should be always considered. In practice, an attacker aims to ensure a high ASR of the backdoored model. However, this does not mean a high poison rate is mandatory. A small poison rate (i.e., 0.5%) can already achieve a nearly 100% ASR e.g., for the source-agnostic backdoor attacks [35]. For example, it [19] has shown that 30 poisoned samples out of 50,000 training samples can essentially lead to a close to 100% ASR for CIFAR10 dataset. While it is reasonable to assume that the attacker always prefers high ASR, it could be problematic to use a saturated high poison rate to gain high ASR to evaluate the devised backdoor defense. Because the attacker does not need to use a high poison rate as long as the ASR is already satisfactorily high. In this context, the poison rate factor should be always considered to be evaluated by backdoor defenses, especially for those relying on inspecting training dataset to identify poisonous samples. Since the attacker always tries to reduce the attack budget (i.e., using the smallest fraction of poisoned samples to be stealthier) as long as the ASR is already high.

3) Single-Model Justification: This is related to model-based detections. Although most such detection countermeasures are designed to examine whether there is a backdoor or not per model-under-test. We note that few defenses (in particular, AI-against-AI defenses such as MNTD and ULP) have only evaluated their performance using the AUC metric against a group of models consisting of both backdoored and benign ones that train on the same dataset. The AUC computation process is independent of a threshold that is used to decide whether a single model under test is backdoored or not. In the real world, the detection often has to be performed per target model, which often relies on a threshold to determine whether the given single model has a backdoor or not. For example, in the model outsourcing scenario, normally, the user receives one model per dataset from a third party. It is imperative to examine the applicability of the model detection countermeasures on justifying a single model’s backdoor behavior.

4) Hyperparameter Sensitivity: The hyperparameter sensitivity is closely related to defenses that rely on AI-against-AI approach, where a detection model or so-called meta-classifier is trained to judge whether a given model is backdoored or not. The training of the meta-classifier often requires meta-data (i.e., logits) that are extracted from shadow models trained on the same dataset (i.e., CIFAR10) on which the model-under-test is trained. The meta-data samples (i.e., logits) are inputs to the meta-classifier while their labels of backdoored or clean (as binary labels) are outputs during meta-classifier training. Once the meta-classifier is trained, it is used to judge whether a given model or the model-under-test trained on the same task (i.e., CIFAR10) is backdoored or not by feeding its extracted meta-data (i.e., logits) as input. However, training hyperparameters (i.e., training epochs, batch size, or even model architecture) of many shadow models are expected to be different from those of the model-under-test. Because the shadow models are usually trained without fully converging or on shallow models to retain the training overhead of the meta-classifier to be acceptable. For example, thousands of shadow models are required before training the meta-classifier in MNTD [14]. While for the mode-under-test, it is usually trained on a more complex model, higher number of epochs, and an optimized batch size to fully convergence for a high accuracy. In this context, hyperparameter sensitivity of the model-under-test has to be evaluated for these AI-against-AI backdoor defenses.

5) Model Complexity: Complicated models should be always considered because only in this case, can the model outsourcing (i.e., the main backdoor attack surface) be more preferred due to unavailable high computational resources and
deep learning expertise. In addition, as aforementioned, for the AI-against-AI defenses, the model complexity of the model-under-test is expected to be higher than that of the shadow models due to that the later has to be utilized to reduce the training overhead of the meta-classifier. Therefore, when a backdoor defense is evaluated, its robustness under complex models, in particular, deeper networks, should be taken into consideration.

IV. NON-ROBUST CASES REVELATION

We reveal and elaborate on one or more evasive cases serving as non-robust failure cases per countermeasure. Note that a methodology summary of the examined detection defense is introduced when revealing its first non-robust case. We reveal the non-robust cases according to the failure factors identified in Section III-B.

A. Binary Classification

The Neural Cleanse needs to iterate each class category to identify the backdoor infected label and reverse the trigger. So it might not be applicable to binary classification. We first introduce its methodology and then formally prove its inapplicability. We then show that other defenses sharing certain similarities (i.e., iterating each class) are likely to fail to this same non-robust factor.

1) Neural Cleanse: This detection stems from the key intuition: the backdoor trigger can cross the decision boundary of the model, akin to creating a “shortcut” from the ground-truth label to the target label in the latent space of the trigger-carrying sample [20]. Neural Cleanse estimates the minimum perturbation increment required to misclassify samples of any other categories into a certain category, thus generating the smallest “shortcut” as the potential reverse-engineered trigger for this category. The specific implementation processes of Neural Cleanse are divided into three steps.

- Step 1: Reverse-engineer a smallest trigger/perturbation for a given label (i.e., assuming this label is the targeted label) by using the optimization algorithm.
- Step 2: Perform Step 1 iteratively to generate a one-to-one corresponding reverse-engineered trigger per label.
- Step 3: Estimate whether there is a reverse-engineered trigger of a category is significantly smaller than other categories. The significant outlier reverse-engineered trigger means a real trigger, and the label that matches the trigger is the attacker targeted label.\footnote{Note if such an outlier is not found, the Neural Cleanse will deem the model as benign and will not output infected label.}

After producing all reverse-engineered triggers in Step 3, Neural Cleanse uses the Median Absolute Deviation (MAD) to estimate the anomaly index of the model. This method is more flexible in processing outliers than standard deviations, and can greatly reduce the impact of outliers on assessment robustness. More specifically, it calculates the $L_1$ norm of all reverse-engineered triggers, where the reverse-engineered trigger of the suspected infected label has a substantially small $L_1$ norm value, standing for the smallest perturbation. Then the median of absolute deviations between each norm and the median is calculated and recorded as MAD. The anomaly index is defined as the absolute deviation of the minimum norm divided by the MAD. Finally, a constant estimator (1.4826) is used to normalize the anomaly index. The anomaly index $AI$ is expressed as:

$$AI = \frac{\text{Min}(L_1) - \text{Med}(L_1)}{\text{Med}(L_1) - \text{Med}(L_1)}$$

where $L_1$ is the $L_1$ norm of all reverse-engineered triggers and $C$ is the constant estimator. The $\text{Med}(\cdot)$ and $\text{Min}(\cdot)$ represent the median function and the minimum function, respectively. Any model with an anomaly index greater than 2 is more than 95% likely to be a backdoored model.

**Non-robustness:** Neural Cleanse is not well suited for a classification task with only a few classes. In particular, for the binary classification, we prove that the anomaly index of the target model is always equal to 0.67, so the Neural Cleanse is completely non-applicable in this case.

Binary classification has only positive and negative classes. Neural Cleanse reverse-engineers a trigger for positive and negative samples, respectively. Assuming that the $L_1$ norms of reverse-engineered trigger 1 and reverse-engineered trigger 2 are $t_1$ and $t_2$, respectively, and $t_1 < t_2$. According to Equation 1, the anomaly index can be expressed as:

$$AI = \frac{\text{Min}([t_1, t_2]) - \text{Med}([t_1, t_2])}{\text{Med}([t_1, t_2]) - \text{Med}([t_1, t_2])}$$

and vice versa for $t_1 > t_2$. Among them, the $C$ is set to 1.482, so the anomaly index calculated by Neural Cleanse for the binary classification problem will be constantly equal to 0.67. This $AI$ value is much smaller than the threshold of 2. Therefore, Neural Cleanse will consider any 2-class model as a benign model regardless whether it is essentially backdoored or not.

**Observation 1:** The Neural Cleanse is non-applicable to binary classification tasks as the anomaly index is proven to be a constant.

In addition to Neural Cleanse, there are other defenses such as Trojan signature [22], SCAn [8], B3D [34] also requiring iterating each class to identify backdoor behavior. They can always be inapplicable in the binary classification task unless empirical rules are delicately considered. For example, B3D suggests judging whether there is a backdoor if the $L_1$ norm of the reversed trigger is smaller than $\frac{1}{2}$ of the median once the number of classes is small. However, this threshold may not always work because it is an empirical setting that can be dependent on e.g., the dataset—B3D however does not validate any backdoored binary classifier and does not release the source code. In this context, we formalize the Trojan signature [22] and SCAn [8] non-robustness against binary classifier.

2) Trojan Signature: The Trojan signature relies on the observation that many of the underlying features of the original class will still be present in a trigger-carrying sample. So that the trigger must overtake the original class confidence to force the trigger-carrying sample to be predicted into the target class.
This enforcement results in the correlation weights of the target class having a significantly larger inner product with the non-negative feature representations of each class. Note the non-negative feature refers to the feature that is activated by ReLU function. In other words, the correlation weights of the target class are significantly different from those of the non-infected classes.

The analysis of the Trojan signature focuses on the final linear layer of the network where the weights $W \in \mathbb{R}^{c \times d}$, c is the number of task classes and d is the dimension of the input features. The Trojan signature first calculates the average weight $w_j = \frac{1}{J} \sum_{i=1}^{J} w_{i,j}$ ($W \in \mathbb{R}^{c \times d}$ is a matrix and $i/j$ the subscript of the rows/columns) per class and sorts them so that $w_{i_1} \leq \cdots \leq w_{i_{c-1}} \leq w_{i_c}$. The Q-value is then introduced to decide whether the model-under-test has a backdoor:

$$Q = \frac{|w_{i_1} - w_{i_{c-1}}|}{w_{i_c} - w_{i_1}}. \quad (3)$$

If Q is greater than a predetermined threshold then the model-under-test is backdoored.

**Non-robustness**: However, the Trojan signature, similar to the Neural Cleanse, is not suitable for the binary classification task. Specifically, a binary classification model with linear layer weights $W \in \mathbb{R}^{2 \times d}$, assuming the average weight of the two classes follow $w_{i_1} \leq w_{i_2}$, then it is easy to infer:

$$Q = \frac{|w_{i_2} - w_{i_{c-1}}|}{w_{i_c} - w_{i_1}} = \frac{|w_{i_2} - w_{i_1}|}{w_{i_2} - w_{i_1}} = 1. \quad (4)$$

Both the Q-value and threshold range from 0 to 1, while the Q-value of the binary classifier is constantly equal to 1. This means that Trojan signature will always recognize the 2-classes model as a backdoored model by default.

**Observation 2**: The Trojan Signature is non-applicable to binary classification tasks as its Q-value is proven to be a constant of 1.

3) SCAn: SCAn is built on a crucial observation that there are fundamental differences in the distribution of representations between images of the target class and infected images from other classes. To identify these disparities, SCAn employs statistical methods to estimate the optimal parameters for the decomposition and untangling models. These estimated parameters are then compared to the null hypothesis using a likelihood ratio test. If the null hypothesis is rejected, the corresponding labels are identified as infected. In this way, SCAn effectively detects and labels the infected categories. In this way, SCAn is unable to effectively identify and detect backdoored binary classification models.

**Observation 3**: SCAn is non-applicable to binary classification tasks as the test statistic has been demonstrated to be a constant.

B. Poison Rate

We reveal that ABS and SCAn are non-robust against decreased poison rate, despite the ASR of the backdoored model being retained to be equally high.

1) ABS: Artificial Brain Stimulation (ABS) is inspired by electrical brain stimulation (EBS), which uses a controllable way to change the activation value of an individual artificial neuron but fix the rest to study whether it is damaged [13]. The ABS is based on two key observations: one is that there are damaged neurons in the backdoored model, and the backdoor effect is likely to be dominated by one or a group of neurons in the compromised model. Another observation is that the subspace of the target label in the decision space can cross the decision boundary of all labels, and the activation of damaged neurons can cause the model to classify the trigger sample into the target label.

**Neuron Stimulation Function (NSF) of each Neuron**: In the process of identifying whether the model has a backdoor or not, ABS explores the influence of internal neurons’ activation on the output activation of each label. Specifically, ABS observes the forward inference process of the model on benign samples and fixes the activation values of other neurons belonging to the same layer as one neuron $\alpha$ to be tested. Immediately afterwards, the neuron $\alpha$ is stimulated to change its activation value to calculate its effect on the activation of subsequent layers. Then it stimulates the neuron $\alpha$ to change its activation in order to calculate its effect on the activation of the subsequent layers. Finally, the relationship between the activation the neuron $\alpha$ and the activation of the output layer is determined, which is called neuron stimulation function (NSF).

**Damaged Neurons Identification**: After the above process, the NSF of each neuron in certain chosen layers is calculated.

where $\mathcal{L}$ is the entire set of class labels and the constant $C=1.4826$ represents the normalization constant for the standard normal distribution. When $J^* > 7.3891 = \text{exp}(2)$ the null hypothesis can be confidently rejected with a confidence level exceeding (1-1e-9). Therefore, the class t is identified as contamination.

**Non-robustness**: Nevertheless, SCAn is not suitable for binary classification tasks because it relies on the MAD approach, similar to Neural Cleanse, for outlier detection. Let’s assume the class statistics for a binary classification task are denoted as $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Based on Equation 5, we can deduce the following:

$$J_1^* = J_2^* = \frac{1}{C} = 0.67. \quad (6)$$

It is evident that the values of $\mathcal{T}_1$ and $\mathcal{T}_2$ are significantly smaller than the threshold $\text{exp}(2)$. Consequently, SCAn will classify any binary classification model as benign, irrespective of whether it is inherently compromised or not. In other words, SCAn is unable to effectively identify and detect backdoored binary classification models.
(testing all neurons in all layers could be costly). If the activation of a certain neuron within a certain range significantly increases the output activation of a specific label, the neuron is marked as a damaged neuron candidate. ABS can mark the 10 most likely damaged candidate neurons through the NSF of the neuron.

**Backdoored Model Determination:** After filtering out the damaged candidate neurons in the model, an optimized method is leveraged to reverse engineer a trigger per candidate neuron. If the reversed trigger is effective for other benign samples, then the neuron is affirmed to be a damaged neuron and the trigger is the final reversed trigger. Following the settings in the ABS released code, it can be determined that the model has a backdoor when the reversed trigger based attack success rate (REASR) is greater than 88%.

**Non-robustness:** In [13], the ASR of the backdoored model under evaluation and the poison rate is not clearly given when reporting the REASR score. In our reproduction experiments, we noticed that the ABS always fails to detect the backdoor even it exhibits nearly 100% ASR—same as reported in [13] but the backdoor is implanted with a not very high poison rate (i.e. 11%).

Following the experimental settings [13], the model is backdoored through data poisoning [1]. Specifically, given the training dataset, a subset is randomly selected and the trigger is stamped on each sample, then the sample label is modified to be the targeted label. The fraction of the subset to the entire set is the poison rate. We use a set of different poisoning rates: 50%, 11%, 1.5% and 0% (clean model). Four backdoored models are trained based on each setting, where the model architectures are VGG16 and ResNet18: each trains on CIFAR10, GTSRB and STL10. The backdoor attack type is the class-agnostic attack that ABS aims to detect. The trigger follows the settings in Section III-A.2.

The ASR and REASR of backdoored models are detailed in Figure 1. Although the backdoored model still has a very high attack success rate (nearly 100%), ABS does not consider it a backdoored model even if the ASR is nearly 100% when the poisoning rate for implanting the backdoor is not very high (in fact, the failure case of 11% poisoning rate is already high). This is because the REASR score of the ABS is below than the threshold of 0.88. One possible reason is that the training set with a higher rate of poisoning samples (i.e. 50%) will affect the behavior of neurons more drastically during the process of injecting the backdoor, causing the backdoor effect to be expressed by only one or a few of neurons. On the contrary, a training set with a not very high poisoning rate will trivially cause less damage to a given specific neuron, and its malicious behavior is scattered throughout the neural network, requiring a large number of neurons to be cooperated for the sake of an effective backdoor.

It is notable that the data poisoning based attack usually relies on a small poisoning rate, e.g., several percent or even below 1% [19]. As in Figure 1, we can observe that the REASRs of 11.1%, 1.5% and 0% poison rates are quite similar. Note the 0% poison rate corresponds to the clean model, this therefore means that the REASRs of 11.1%, 1.5% poison rates are almost indistinguishable from the clean model. Therefore, the ABS appears to be quite cumbersome in the reality. It is also noted that the REASR score of the ResNet18 is much smaller than that of the VGG16 for both CIFAR10 and GTSRB datasets. This implies that the ABS is sensitive to model architecture. Moreover, according to Figure 1, the REASR is also dependent on the dataset: GTSRB and STL10 have a lower value than CIFAR10.

**Observation 4:** The ABS is unable to detect backdoored model even if the ASR is nearly 100% when the poisoning rate for implanting the backdoor is not very high (in fact, the failure case of 11% poisoning rate is already high). In addition, the ABS appears to be sensitive to model architecture and dataset. The main failure cause is due to the ABS’s strong assumption that the backdoor attack only compromises a single or few neurons, which appears to be not always held in practice.

2) **SCAN:** The SCAN is an offline training data inspection defense. SCAN is devised to mainly defend against source-specific attack [29], which is more difficult to detect compared to source-agnostic attacks. It performs statistical analysis of global information and decomposes features of samples into identity features (class-specific features such as face features) and variation features (class-independent features such as expression features) using the expectation–maximization (EM) algorithm. However, in the case of a backdoor attack, poisoned samples change the distribution of identity features and variation features of the target class, thus distinguishing it from the distribution of features of other non-infected classes.

SCAN is based on the core assumption that the variation features among clean samples follow the same distribution and that the variation features of a sample are independent of its class. The target class samples under attack consist of

---

2 This threshold of 88% is following the setting in the source code.

3 We note that in Table II of [13], it does consider three poison rate of 1%, 9% and 50%. However, only a single REASR score is reported that is always 100% or nearly 100%, it is unclear whether this score refers to the best case or other case—only the best case matches our reevaluations.

4 The source code is at https://github.com/naiyeleo/ABS, which we reproduce.

5 We have shown that the ABS is ineffective to the backdoored model crafted through weight perturbation, which is somehow alike an adaptive attack. The main failure reason is that the ABS has a strong assumption that the backdoor attack compromises a single neuron or few neurons. Interested readers can find more details at https://arxiv.org/abs/2204.06273
two disjoint subsets of clean and poisoned samples, which means that the identity features of the target class consist of a mixture of samples from multiple classes. Therefore, SCAn transforms the backdoor detection problem into a hypothesis testing problem, i.e., to determine whether the identity features of a certain class of samples obey the mixture distribution. In other words, a class will be identified as contaminated if identity features of the samples in that class conforms to the mixture distribution.

**Non-robustness:** We trained a total of 12 source-specific backdoored models to evaluate the sensitivity of SCAn to the poisoning rate by considering two model architectures, VGG16 and ResNet18, and two datasets, CIFAR10 and GTSRB (i.e., both datasets were used in SCAn [8]), for three different poisoning rates {1%, 2%, 3%}. Table II reports the performance of all backdoored models, indicated by their CDA and ASR values. For each poisoning rate, the ratio of poisoned samples (i.e., trigger samples with changed label) and covered samples (i.e., trigger samples but with intact label) is the same. The target class is set to class 0 and the source class is set to class 1, which means that samples from class 1 are misclassified to class 0 by the model when triggers are added. But trigger samples from non-source classes have no misclassification effect. On the one hand, for all settings the backdoored models have more than 95% ASRs (see Figure 2). On the other hand, for the backdoored models with low poisoning rate, the metric Ln(J*) of SCAn on the target class shows a decreasing trend. More specifically, for the model with 1% poisoning rate although the ASR is kept to be 95%, but Ln(J*) is less than the threshold 2, which fails to be detected by SCAn. Therefore, we can see that the SCAn is indeed sensitive to poison rate to a large extent, despite its sensitivity being much less compared to the ABS.

**Observation 5:** The SCAn is sensitive to poison rate to a large extent, despite its sensitivity being much less compared to the ABS.

**C. Single-Model Justification**

1) **MNTD:** Meta Neural Trojan Detection (MNTD) aims to train a meta-classifier that can distinguish backdoored models from clean models given a specific dataset [14]. The meta-classifier takes meta-data extracted from the model-under-test, or namely the target model, as input and performs binary classification to determine whether it has a backdoor or not. We note that the meta-data is logits from the target model when it is queried with a number of querying samples (i.e., MNIST images in the MNIST dataset).

The MNTD has three steps: shadow model generation; meta-training; and target model detection. The first two steps are performed only once to gain a trained meta-classifier that will be used for target model detection. The MNTD assumes the defender has a small clean dataset of the given dataset (i.e., MNIST) on which the target model trains.

**Shadow Model Generation:** A large number of shadow models (i.e., several thousand) on the given dataset on which the target model trains are trained on a small dataset accessible to the defender. The shadow models consist of clean models and backdoored models. Both are trained with different initialized parameters. For the backdoored models, each model is trained with a differing backdoor setting through the typical poison attacks to insert backdoor [1]. The MNTD considers two typical attack settings: modification attacks [1] and blending attacks [2]. A modification attack directly modifies some part of the sample as a trigger and sets the label of the poisoned sample as the target label. In contrast, a blending attack blends the trigger pattern into the original sample (e.g., blending a specific pattern into the image). The former is usually a small-size trigger while the latter is normally a large-size trigger but transparent, as exemplified in Figure 3.
shadow model makes outputs (in particular, logits according to its source code implementation\textsuperscript{6}). These outputs (logits) of differing queries are concatenated to serve as the input of the meta-classifier, while benign or backdoored labels are the ground-truth labels of the meta-classifier. To be efficient, these queries are not simply randomly selected images from a dataset (i.e., CIFAR10), the content (i.e., pixel values) of these queries are iteratively updated/optimized along with the training processing of the meta-classifier.

**Target Model Detection:** The optimized queries are used as the input of the target model to obtain the logits. The meta-classifier will receive the logits as input and output a score. Note this score is the logits of the meta-classifier. The score is used to measure the malicious degree of the target model, and the expected score of the clean target model is smaller than that of the backdoored target model. The meta-classifier takes the median of the scores of all shadow models in the training/validation set as the threshold. If the score of the target model is greater than the threshold, it indicates that it is injected into the backdoor.

We note that the meta-classifier performance is in fact not evaluated per target model according to the above Target Model Detection step\textsuperscript{7}.

**Non-robustness:** The MNTD uses Area Under Curve (AUC) as the primary metric for evaluating the meta-classifier performance. In this context, the AUC, however, needs to be computed on a set of benign and backdoored target models trained on the same dataset over which the targeted model trains. The AUC computation process is independent of a threshold that is used to decide whether a target model is backdoored or not (as described in the above step Target Model Detection). In the real-world, the detection has to be performed per target model, which often relies on a threshold to determine whether the given target model has a backdoor or not. However, it is unclear how to choose such a threshold in [14], thus determining whether a given model is backdoored or not.

According to the settings in the source code,\textsuperscript{8} the median of all shadow model scores in the testing shadow model set is chosen as the threshold. However, this threshold determination appears to be unreasonable. In general, the model trainer should be assumed to have no threshold knowledge of the testing set unless it is already tested. Consequently, we choose the median of all shadow model scores in the training shadow model set as the threshold, rather than the testing shadow model set (that are target models).

Even in this context, we find that the correct threshold chosen is still challenging. Because there is a significant difference between the thresholds in the training shadow model set and the thresholds in the testing shadow model set. In other words, the usage of the training shadow model set determined threshold is non-applicable to separating positive and negative samples (clean and backdoored models) in the testing shadow

\textsuperscript{6}In the description of [14], it refers the output as representation vector, but does not explicitly mention what it is.

\textsuperscript{7}In Section II-B of [14] when describing the workflow of meta neural analysis, it does mention the inference is done on a given target model, but there are no evaluations on a single model in its experiments.

\textsuperscript{8}https://github.com/AI-secure/Meta-Neural-Trojan-Detection

| dataset | No. | Threshold of training set | Threshold & Accuracy(%) |
|---------|-----|---------------------------|-------------------------|
| MNIST   | 1   | -0.49 & -7.59             | 15.70 & (50:00; 50:00)  |
|         | 2   | -0.80 & -2.89             | 4.95 & (50:20; 50:00)   |
|         | 3   | -0.94 & -1.24             | 8.35 & (50:00; 50:00)   |
|         | 4   | -3.46 & -2.35             | 5.60 & (50:00; 50:00)   |
|         | 5   | -3.49 & -3.90             | 1.69 & (50:00; 50:00)   |
| CIFAR10 | 1   | -2.64 & -3.21             | 2.94 & (50:00; 50:00)   |
|         | 2   | -0.24 & -1.96             | 3.14 & (50:00; 50:00)   |
|         | 3   | 0.03 & -3.51              | 0.09 & (52:34; 50:00)   |
|         | 4   | -2.90 & -5.17             | 8.57 & (50:00; 50:00)   |
|         | 5   | -3.52 & -6.07             | 3.76 & (50:00; 50:00)   |

\textsuperscript{-M} means the backdoored model is attacked by a modification attack, while \textsuperscript{-B} means a blending attack.\textsuperscript{9}

| observation 6: | The MNTD is challenging to inspect target model individually as the AUC metric cannot do so, while the median score of meta-classifier as threshold exhibits an almost guessing performance. So it is unclear how to apply the MNTD to inspect an individual target model per dataset that is more likely demanded in practice. |

\textsuperscript{9}The source code has trained five meta-classifiers that we follow.

\textbf{D. Hyperparameter Sensitivity}

1) MNTD: As shown in Table III, there is a significant variance between the thresholds of the meta-classifier determined on the training shadow model set and on the testing shadow model set. This may be caused by the difference in hyperparameters used to train the training shadow model set and the testing shadow model. In this case, we further explore the effect of hyperparameters on the MNTD detection performance and confirm that the meta-classifier is highly sensitive to hyperparameters. Generally, once the logits extracted from the target model differ from the logits extracted from the training shadow models that are used to train the meta-classifier—the extracted logits are the input model set that are target models, which indicates that it is challenging to adopt the MNTD in practice.

In Table III, determined thresholds from five trained meta-classifiers\textsuperscript{9} are detailed. In addition, for each meta-classifier, we use the threshold of the training shadow model set to evaluate the target models in the testing shadow model set (the testing set consists of 256 clean and backdoored models, respectively, the same as the source code setting), which accuracy results are also detailed. The accuracy is the ratio that the number of target models in the testing set correctly judged by the meta-classifier—whether the targeted model is benign or backdoored—to the number of all models under evaluation. The targeted model has backdoored if the score is higher than the threshold, otherwise, benign. We can see that the accuracy is always 50% akin to guessing. In addition, we can see that there exists a large difference among the training shadow model set determined thresholds even the meta-classifiers are trained for the same dataset (i.e., MNIST). In addition, the thresholds between the training and testing shadow model sets determined by the same meta-classifier are also diverse.
We note that the convergence of the testing shadow models is CIFAR10 case even on the (4 conv + models training (though 12 seconds per shadow model) in the extremely costly, e.g., 14 hours to complete the 4096 shadow without fully converging—even such partial training is already training shadow models but also the testing shadow models.

To reduce computational overhead, the MNTD trains not only producing the MNTD according to the same setting in Table IV. hyperparameter on the MNTD. For the detection performance, we use the hyperparameter. For the epochs, batch size, and proportion of the training set, respectively. (middle) batch size, (bottom) proportion of training images. Fig. 4. The influence of hyperparameters on AUC, (top) the number of epochs, (middle) batch size, (bottom) proportion of training images.

Table V summarizes two main performances that are CDA and ASR of the shadow benign/backdoored models by reproducing the MNTD according to the same setting in Table IV. To reduce computational overhead, the MNTD trains not only the training shadow models but also the testing shadow models without fully converging—even such partial training is already extremely costly, e.g., 14 hours to complete the 4096 shadow models training (though 12 seconds per shadow model) in the CIFAR10 case even on the (4 conv + 3 full) shallow network. We note that the convergence of the testing shadow models is indeed better than that of the training shadow models (reflected by higher CDA) mainly attributed to the higher proportion of training images used (see Table IV). As a comparison, training a well-converged model takes only about e.g., one hour for ResNet50—much shorter than 14 hours to train the MNTD’s meta-classifier for a given task. This somehow means that the MNTD is inapplicable to the backdoor induced in the common model training outsourcing scenario. As it is more reasonable for the user to train the model by herself/himself given that the MNTD does require heavy computation and ML expertise—costing more than training from scratch.

The AUCs are detailed in Figure 4 for the specific change of epochs, batch size, and proportion of training set, respectively. Before diving into detailed analyses, we provide necessary knowledge about how the MNTD meta-classifier scores the benign and backdoored models to ease the following understandings. According to the source code, the backdoored models are generally scored higher by the meta-classifier than benign models under the same hyperparameter settings. Consequentially, the evaluated AUC exhibits a higher value (i.e., close to 100%) by virtue of the larger difference between the two scores of backdoored and clean target models. In other words, if the score difference between the clean and backdoored model measured by the meta-classifier decreases, then the AUC will degrade (i.e. close to 50%).

Epochs: In the first case, the number of epochs when training the clean testing models is varied from 4 to 8 (the source code uses 4)—the batch size and proportion of training set are fixed to be 100 and 0.5, respectively. As shown in Figure 4 (top row), the AUC decreases as the epoch number

Table V

| Clean or backdoored dataset | Training shadow model dataset | Testing shadow model dataset |
|-----------------------------|-------------------------------|-------------------------------|
| epochs | batch size | proportion of training images |
| MNIST | 94.19 | 86.96 | 99.72 |
| MNIST-M | 94.19 | 86.96 | 99.72 |
| CIFAR10 | 61.60 | 81.22 | 99.72 |
| CIFAR10-B | 59.18 | 81.22 | 99.72 |

Table VI

| Model | Epochs | Batch size | Proportion of training images | CDA(%) & ASR(%) | Median of scores | AUC(%) |
|-------|--------|------------|-------------------------------|----------------|-----------------|--------|
| Benign | 50 | 100 | 1 | 80.65 & - | 65.19 | 43.20 |
| Backdoored | 50 | 100 | 1 | 79.64 & 99.99 | 45.18 | 43.20 |
| Benign | 100 | 100 | 1 | 80.65 & - | 82.88 | 43.20 |
| Backdoored | 50 | 100 | 1 | 79.64 & 99.99 | 65.19 | 13.30 |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
goes up. For the CIFAR10-M, the AUC almost degrades to 50% for epoch number of 8. In contrast, in the second case, when the epoch number of training testing backdoored models decreases from 4 to 1 (the source code uses 4), the AUC also degrades e.g., to 65% in the CIFAR10-B.

In the former case, it appears that the closer the clean target model converges (more epochs are applied), the higher score for it is measured by the meta-classifier. As the score of backdoored models is fixed, the score difference between the benign and backdoored models is narrowed, resulting in a lower AUC. In the latter case, when the backdoored model less converges, the lower its score is given by the meta-classifier. This also results in a smaller score difference between the benign and backdoored models, thus a smaller AUC.

**Batch Size:** In this setting, the batch size when training the testing clean and backdoored models varies while keeping the rest two constant (4 and 0.5 for epochs and proportion of training images, respectively). As can be seen from Figure 4 (middle row), the AUC of benign models notably decreases as the batch size is reduced from 100 to 32. The AUC decreases when the batch size of the backdoored model increases from 100 to 400. Note a smaller batch size means there are more steps in a given epoch, thus the model will be often converged faster. As the meta-classifier scores higher for the more converged models, we can observe a smaller score difference between benign models and backdoored models, because the benign models converge better to gain a higher score by the meta-classifier but the backdoored models’ higher score fixed. Similarly, when the benign model’s lower score is fixed and the backdoored models are less converged resulting in correspondingly lower score, the score difference between the benign and backdoored models is also decreasing, thus rendering a degraded AUC.

**Proportion of Training Images:** In this setting, the images used for training the testing benign and backdoored models are changed while the rest two hyperparameters being fixed (4 and 100 for epochs and batch size, respectively). The AUC results are shown in Figure 4 (bottom row). When testing benign models are trained with more images, it converges better, thus scored higher by the meta-classifier. This consequently leads to smaller score differences between the benign models and backdoored models—the backdoored models are fixed with higher scores. So that a lower AUC is exhibited. Similarly, fewer training images for backdoored models result in lower scores by meta-classifier, exhibiting smaller score differences, and consequently degraded AUC.

Based on the above analyses, we can see that either a higher epoch number or smaller batch size, or more training images will result in the benign model converging better, at which point the meta-classifier will output a higher score for it. Conversely, the less adequately trained the backdoored model is, the lower the score will be for the backdoored models. Both cases narrow the difference between the score of the benign model and the backdoored model, resulting in a lower AUC. Therefore, the tendency for backdoored models to score higher than benign models is likely to be caused by the fact that the backdoor sub-task in the backdoored model always converges (i.e. high ASR) faster than the main task, validated by higher ASR than the CDA in Table V.

In summary, the meta-classifier is extremely sensitive to hyperparameters that are used to train its shadow models and the targeted models, where a slight change (i.e. epochs variation) is sufficient to render the meta-classifier fails to identify the backdoored model. Because the characteristics of the meta-classifier input that is the logits extracted from the target model will not follow a similar extracted logits distribution of those shadow models used to train the meta-classifier. For instance, the AUC decreases from 96% to 50% when the epoch of the testing benign models increases from 4 to 8 in the CIFAR10-M test set—50% is simply guessing. In other words, the meta-classifier seems to prefer to give higher scores for the well-trained models (i.e., those model converged well) rather than for the backdoored models.

**Observation 7:** The MNTD is found to be (highly) sensitive to hyperparameters used to train the model that is under inspection, which renders its usage difficulty even when the AUC as a detection metric is used. Because the AUC is sensitive when the hyperparameter (i.e., epochs, batch size) varies. Note that the MNTD is (at least inexplicitly claimed) generic to different backdoor types (i.e. source-agnostic backdoor) and triggers. It however easily fails for the simplest backdoor attack and easiest-to-detect triggers.

2) ULP: Inspired by Universal Adversarial Perturbations, Kolouri et al. [21] propose the Universal Litmus Patterns (ULPs) which is a set of optimized input images. Given a suspicious model $M$, they first take the ULPs as $M$ input and outputs a prediction vector (i.e., probability after softmax) for each ULP. All these vectors are concatenated, which is then fed into a linear layer with 2 neurons to gain the confidence that indicates whether $M$ is benign or backdoored.

Both [21] and MNTD are meta-classifier-based detection methods and consist of two optimizeable components: ULPs (functionality is same to the optimizeable queries in MNTD) and a linear layer that acts as the meta-classifier. We train a total of 800 models on CIFAR10 with 400 benign and 400 backdoored models with different trigger settings as the training model set, referring to the default settings in the source code of [21]. In addition, we train two sets of testing models as the testing set of the meta-classifier, each containing 20 benign and 20 backdoored models, respectively. Specifically, one testing model set follows the same distribution as the training model set (i.e., the model training hyperparameters are the same as the training model set), and the other testing set is different (they are the testing models that have been trained in Table VII) for validating the sensitivity of the hyperparameters, and the average performance of all testing models is reported in Table VII in terms of CDA and ASR.

We trained three meta-classifiers, each with a different number of its corresponding queries/ULPs $m \in \{1, 5, 10\}$. We have also used the baseline queries/ULPs that are not optimized but merely created with random perturbations or noise, referred as noise in Figure 5—the optimized queries are referred as ULP. As shown in Figure 5, the left depicts the ROC curves of the meta-classifiers on 20 testing models with same hyperparameter setting of the training models, which achieve an AUC close to 1 for both $m = 5$ and 10. However, the right shows...
that the ROC curves of the meta-classifiers on 20 testing models trained with different hyperparameter settings exhibit a significant decrease in AUC. More specifically, for the \(m = 10\) meta-classifier, it classifies all 40 testing models as backdoored models. This means that the meta-classifier is only to detect backdoored models that are trained with the same or similar hyperparameter settings as the training model set to train the meta-classifier. We argue this is impractical. Because the hyperparameter setting can be arbitrary and is unknown to the defender. The design of AI-Against-AI backdoor countermeasures relying upon meta-models such as the MNTD and UPLs should demonstrate robustness against varying hyperparameters used to gain the target/testing model. Inadvertently, both MNTD and UPL require hundreds or thousands training shadow models trained on the target task (i.e., CIFAR10) to train a meta-classifier. Training these shadow models are very expensive, which overhead is unbearable unless they are trained without convergence (i.e., as followed by the MNTD) or with a shallow model. However, a target model under examination has to be trained well to meet the normal behavior for clean data, thus it already has a different hyperparameter setting (i.e., a more complex model architecture, more epochs, smaller learning rate, smaller batch size).

**Observation 8:** The ULP as a AI-against-AI method, similar to MNTD, is found to be (highly) sensitive to hyperparameters as well.

## E. Model Complexity

1) **Neural Cleanse:** We consider four models with much higher number of parameters that are suitable for recognition tasks\(^{10}\), VGG19 [36], ResNet101 [37], Inceptionv4 [38], and ResNeXt152 [39]. The trigger follows the settings in Section III-A.2. A clean model and a backdoored model are trained for each model architecture on each dataset of CIFAR10, GTSRB, and STL10. Note that VGG19 acts as a comparative role. Although it has more parameters, VGG19 model is far shallower than other models. The number of parameters and the number of layers of each of the four models is detailed in Table VIII. This Table also shows that the clean model and its backdoored model counterpart have almost the same CDA while the backdoored model exhibits nearly 100% ASR.

![Fig. 5. Meta-classifier performance measured with ROC. The dataset is CIFAR10 following [21].](image1)

![Fig. 6. The trigger reverse-engineered by Neural Cleanse on the target label. Notably, the intentionally used small and simple trigger here should be easily and correctly reverse-engineered by the Neural Cleanse in principle.](image2)

![Fig. 7. Neural Cleanse performance on four complicated model architectures. Each model architecture evaluates three datasets: CIFAR10, GTSRB, and STL10.](image3)
reverse-engineered trigger tends to present the characteristics of the targeted category samples rather than that of the trigger. The reverse-engineered trigger deviates more from the original trigger as the model is more complex. As $AI$ greater than the threshold of 2 means that the backdoored model is identified by the Neural Cleanse and vice versa, the backdoored model with deeper network cannot be detected—always regarded as benign (shown in Table VIII).

**Observation 9:** The Neural Cleanse almost always fails for deeper complicated networks. In addition, the behaviors (i.e., $AI$) detected by Neural Cleanse between the clean model and the backdoored becomes less distinguishable when the model goes deeper.

2) **MNTD:** Experiments in subsubsection IV-D.1 have followed the same setting of the MNTD code with the same model architecture when training shadow models and testing shadow models. The trained shadow models are usually not fully converged—doing so would incur further substantial training overhead for the MNTD. While for the target models under inspection, it is more reasonable that the target models are converged well and use a different more complex model architecture. In this context, we followed the training procedure for the shadow models but trained 40 VGG11 models (a different model architecture used in the original MNTD source code) to fully convergence on the CIFAR10 dataset as the testing set of the meta-classifier—half benign models and half backdoored models. The training parameters and performance on the meta-classifier for the VGG11-based target models for testing are shown in Table VI. As both the benign and backdoored models are trained to convergence, the meta-classifier gives high scores for both. When the training parameters are the same regardless of the presence of a backdoor in the model, the scores for the converged model are similar to that depicted in Figure 8, with an average AUC of close to 50% for the MNTD evaluation (2nd and 3rd rows in Table VI). And when the training epoch for the benign model is increased to 100 on top of this, the average AUC decreases further (4th and 5th rows in Table VI). Notably, when we examine the score median for the backdoored model and benign models, the score of the benign models is now surprisingly overtaken by the backdoored models. This contradicts the empirical observation of the MNTD that the score of the clean model should be smaller than the score of the backdoored model.

**Observation 10:** The MNTD is also found to be (highly) sensitive to model architectures that are different from the ones used to train the shadow models, which is mainly rooted by its sensitivity to the extraction of logits that are inputs of the meta-classifier.

V. Conclusion

This work is the first to critically examine non-robust cases of DL backdoor countermeasures, which is inadvertently neglected to date. The root cause is its inherent sensitivity to non-robust failure factors, which are unexplored and not elucidated. We have initially identified five such common failure factors, then extensively examined and revealed one or more non-robust cases of each of these chosen influential detection-based backdoor countermeasures under one or more failure factors. Significantly, we have shown that countermeasures sharing some similar concepts or methodologies are likely to fail against the same failure factor. We hope more efforts will be devoted to this research area in the future, or the emerging devised countermeasures should critically examine and clarify their non-robust cases more thoroughly.

**References**

[1] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, “BadNets: Evaluating backdooring attacks on deep neural networks,” IEEE Access, vol. 7, pp. 47230–47244, 2019.

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

[3] S. Hu et al., “BadHash: Invisible backdoor attacks against deep hashing with clean label,” in Proc. 30th ACM Int. Conf. Multimedia, Oct. 2022, pp. 678–686.

[4] E. Wenger, J. Passananti, A. N. Bhagoji, Y. Yao, H. Zheng, and B. Y. Zhao, “Backdoor attacks against deep learning systems in the physical world,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 6202–6211.

[5] H. Ma et al., “Dangerous cloaking: Natural trigger based backdoor attacks on object detectors in the physical world,” 2022, arXiv:2201.08619.

[6] M. Niwadike, T. Miyawaki, E. Sarkar, M. Maniatakos, and F. Shamon, “Explainability matters: Backdoor attacks on medical imaging,” 2020, arXiv:2101.00008.

[7] Y. Gao et al., “Backdoor attacks and countermeasures on deep learning: A comprehensive review,” 2020, arXiv:2007.10760.

[8] D. Tang, X. Wang, H. Tang, and K. Zhang, “Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection,” Proc. USENIX Secur., 2021.

[9] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, “STRIP: A defence against trojan attacks on deep neural networks,” in Proc. 35th Ann. Comput. Secur. Appl. Conf., Dec. 2019, pp. 113–125.

[10] Y. Li et al., “NTD: Non-transferability enabled backdoor detection,” 2021, arXiv:2111.11157.

[11] Z. Chen, S. Wang, A. Fu, Y. Gao, S. Yu, and R. H. Deng, “LinkBreaker: Breaking the backdoor-trigger link in DNNs via neurons consistency check,” IEEE Trans. Inf. Forensics Security, vol. 17, pp. 2000–2014, 2022.

[12] Y. Liu, Y. Xie, and A. Srivastava, “Neural trojans,” in Proc. IEEE Int. Conf. Comput. Design (ICCD), Nov. 2017, pp. 45–48.

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

[14] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, “Detecting AI trojans using meta neural analysis,” in Proc. IEEE Symp. Secur. Privacy (SP), May 2021, pp. 103–120.

[15] N. Carlini et al., “On evaluating adversarial robustness,” 2019, arXiv:1902.06705.

[16] N. Carlini and D. Wagner, “Adversarial examples are not easily detected: Bypassing ten detection methods,” in Proc. 10th ACM Workshop Artif. Intell. Secur., Nov. 2017, pp. 3–14.

[17] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, “NIC: Detecting adversarial samples with neural network invariant checking,” in Proc. Netw. Distrib. Syst. Secur. Symp., 2019.

[18] R. Shokri, “Bypassing backdoor detection algorithms in deep learning,” in Proc. IEEE Eur. Symp. Secur. Privacy, Sep. 2020, pp. 175–183.
Yao Gao et al., “Design and evaluation of a multi-domain Trojan detection method on deep neural networks,” IEEE Trans. Dependable Secure Comput., vol. 19, no. 4, pp. 2349–2364, Jul. 2022.

Bo Wang et al., “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks,” in Proc. IEEE Symp. Secur. Privacy (SP), May 2019, pp. 707–723.

S. Kolouri, A. Saha, H. Pirsivash, and H. Hoffmann, “Universal litmus patterns: Revealing backdoor attacks in CNNs,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 298–307.

G. Fields, M. Samragh, M. Javaheripi, F. Koushanfar, and T. Javidi, “Trojan signatures in DNN weights,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW), Oct. 2021, pp. 12–20.

V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, “Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning,” in Proc. IEEE Symp. Secur. Privacy (SP), May 2022, pp. 1354–1371.

M. Goldblum et al., “Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 2, pp. 1563–1580, Feb. 2023.

H. Chen, C. Fu, J. Zhao, and F. Koushanfar, “DeepInspect: A black-box Trojan detection and mitigation framework for deep neural networks,” in Proc. 28th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 4658–4664.

B. Tran, J. Li, and A. Madry, “Spectral signatures in backdoor attacks,” in Proc. IEEE 2018, pp. 8000–8010.

B. Chen et al., “Detecting backdoor attacks on deep neural networks by activation clustering,” 2018, arXiv:1811.03728.

B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, “Februus: Input purification defense against trojan attacks on deep neural network systems,” in Proc. Ann. Comput. Secur. Appl. Conf., Dec. 2020, pp. 897–912.

S. Wang et al., “CASSOCK: Viable backdoor attacks against DNN in the wall of source-specific backdoor defenses,” in Proc. ACM Asia Conf. Comput. Commun. Secur., Jul. 2023, pp. 938–950.

W. Guo, L. Wang, Y. Xu, X. Xing, M. Du, and D. Song, “Towards inspecting and eliminating trojan backdoors in deep neural networks,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2020, pp. 162–171.

H. Ma et al., “Quantization backdoors to deep learning commercial frameworks,” 2021, arXiv:2108.09187.

Y. Hu et al., “Artificial intelligence security: Threats and countermeasures,” ACM Comput. Surv., vol. 55, no. 1, pp. 1–36, Jan. 2023.

T. A. Nguyen and A. Tran, “Input-aware dynamic backdoor attack,” in Proc. NIPS, vol. 33, 2020, pp. 3454–3464.

Y. Dong et al., “Black-box detection of backdoor attacks with limited information and data,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 16462–16471.

Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, “Anti-backdoor learning: Training clean models on poisoned data,” in Proc. NIPS, vol. 34, 2021, pp. 14900–14912.

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

K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-ResNet and the impact of residual connections on learning,” in Proc. AAAI, 2017.

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 5987–5995.

Huning Qiu received the bachelor’s degree from Southeast University in 2020. He is currently pursuing the master’s degree with the Nanjing University of Science and Technology. His research interests include AI security and privacy.

Hua Ma is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, The University of Adelaide. Her research interests include machine learning security and optimization.

Zhi Zhang received the bachelor’s degree from Sichuan University, the master’s degree from Peking University, China, and the Ph.D. degree in computer science from The University of New South Wales, Australia. He is currently a Lecturer with The University of Western Australia. Prior to that, he was a Research Scientist with Data61, CSIRO. His research interests include system security, rowhammer, and adversarial artificial intelligence.

Alsharif Abuadbba received the Ph.D. degree in computer security from RMIT University, Australia, 2017. He is currently a Senior Research Scientist with Data61, CSIRO, and a Cybersecurity CRC Fellow. He has also several years of experience working as a senior research engineer with Californian-based technology companies. He has contributions to a few U.S. IP filed patents in cybersecurity. His research interests include AI and cybersecurity, the IoT-cloud security, system security, and watermarking.

Wei Kang received the Ph.D. degree in computer science from the National University of Singapore in 2015. He is currently a Research Scientist with Data61, CSIRO. His research interests include data mining and machine learning, predictive analytics, social media analytics, cyber security, access control and management, text summarization, and visualization.

Anmin Fu received the Ph.D. degree in information security from Xidian University in 2011. From 2017 to 2018, he was a Visiting Research Fellow with the University of Wollongong, Australia. He is currently a Professor with the Nanjing University of Science and Technology, China. His research interests include the IoT security, cloud computing security, and privacy preserving. He has published more than 80 technical papers, including international journals and conferences, such as IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Services Computing, IEEE Internet of Things Journal, Computers and Security, IEEE ICC, IEEE GLOBECOM, and ACISP.

Yansong Gao (Senior Member, IEEE) received the M.Sc. degree from the University of Electronic Science and Technology of China in 2013 and the Ph.D. degree from The University of Adelaide, Australia, in 2017. He is currently a Research Scientist with Data61, CSIRO. His current research interests include AI security and privacy, hardware security, and system security.