Towards Inspecting and Eliminating Trojan Backdoors in Deep Neural Networks

Wenbo Guo[1], Lun Wang[2], Yan Xu[3], Xinyu Xing[1], Min Du[4], Dawn Song[2]
[1]The Pennsylvania State University, [2]University of California, Berkeley, [3]Peking University, [4]Palo Alto Networks, Inc.
{wzg13, xxing}@ist.psu.edu, {wanglun, dawnsong}@berkeley.edu, xu.yan@pku.edu.cn, min.du.email@gmail.com

***Abstract*—A trojan backdoor is a hidden pattern typically implanted in a deep neural network (DNN). It could be activated and thus forces that infected model to behave abnormally when an input sample with a particular trigger is fed to that model. As such, given a DNN and clean input samples, it is challenging to inspect and determine the existence of a trojan backdoor. Recently, researchers design and develop several pioneering solutions to address this problem. They demonstrate that the proposed techniques have great potential in trojan detection. However, we show that none of these existing techniques completely address the problem. On the one hand, they mostly work under an unrealistic assumption of assuming the availability of the contaminated training database. On the other hand, these techniques can neither accurately detect the existence of trojan backdoors, nor restore high-fidelity triggers, especially when infected models are trained with high-dimensional data, and the triggers pertaining to the trojan vary in size, shape, and position.**

In this work, we propose `TABOR`, a new trojan detection technique. Conceptually, it formalizes the detection of a trojan backdoor as solving an optimization objective function. Different from the existing technique which also models trojan detection as an optimization problem, `TABOR` first designs a new objective function that could guide optimization to identify a trojan backdoor more correctly and accurately. Second, `TABOR` borrows the idea of interpretable AI to further prune the restored triggers. Last, `TABOR` designs a new anomaly detection method, which could not only facilitate the identification of intentionally injected triggers but also filter out false alarms (*i.e.,* triggers detected from an uninfected model). We train 112 DNNs on five datasets and infect these models with two existing trojan attacks. We evaluate `TABOR` by using these infected models, and demonstrate that `TABOR` has much better performance in trigger restoration, trojan detection, and elimination than `Neural Cleanse`, the state-of-the-art trojan detection technique.

I. Introduction

We have witnessed that deep neural networks (DNNs) have delivered super-human accuracy in a variety of practical use cases, such as facial recognition [27] and object detection [25]. Along with the huge success of deep learning also comes many kinds of adversarial attacks [1], [14], among which trojan attack [13], [21] is a relatively novel one. Technically, this kind of attack inserts contaminated data samples into the training data of a DNN, seeking to trick the system into learning a trojan backdoor through which an adversary could mislead the system to misclassification for arbitrary inputs with a trigger present.

Recently, researchers have proposed several new techniques to inspect the existence of a trojan backdoor in a target learning model (*e.g.,* [30]). As we will specify in Section §III, these works are mostly designed under the assumption of having access to the training database. For the following reasons, such an assumption however is not quite practical. First, a user may not be involved in the training process of an AI system but acquire an AI system from vendors or open model repositories that are malicious, compromised or incompetent. Second, even if a user is engaged in the process of an AI system development, she may obtain a learning model by performing a transfer learning, which may take an existing, untrustworthy AI system as a base model.

To the best of our knowledge, `Neural Cleanse` [30] and DeepInspect [4] are the most recent – if not the only two – research works that can perform trojan backdoor inspection without the aforementioned assumption. Because `DeepInspect`'s objective function and anomaly detection are exactly the same as `Neural Cleanse`, we will only refer to `Neural Cleanse` in the rest of the paper for brevity. Technically speaking, `Neural Cleanse` defines an objective function and formalizes trojan detection as a non-convex optimization problem. With such a design, resolving the optimization can be viewed as searching special adversarial samples (*i.e.,* input samples with a trigger attached) in an adversarial subspace defined by that objective function. In [30], `Neural Cleanse` demonstrates decent performance in pointing out the existence of a trojan backdoor. However, as we will show in Section §V, `Neural Cleanse` becomes completely futile, especially when an infected model is trained to deal with high-dimensional data or it ingests a trigger with varying size/shape/location. We conjecture this is because both the high-dimensional and the trigger properties significantly vary the number of adversarial samples in the adversarial subspace, which forces the aforementioned adversarial sample search into encountering more adversarial samples that are not the true interest. In the design of `Neural Cleanse`, Wang *et al.* utilized a simple outlier detection algorithm to distinguish identified adversarial samples from the special ones. As we will show in Section §V, with more adversarial samples fed into this algorithm, it inevitably demonstrates the difficulty in distinguishing the special adversarial samples from other adversarial ones. In addition, we discover that `Neural Cleanse` has limited capability in restoring high-fidelity triggers. As we will demonstrate in Section §V, this jeopardizes our ability to patch a trojan-implanted model.

Inspired by the finding and the analysis above, we propose

The first two authors contributed equally.

TABOR, a new trojan detection approach. Similar to Neural Cleanse, it also formulates trojan detection as an optimization problem and thus views the detection as searching trigger-inserted inputs in an adversarial subspace. However, differently, TABOR tackles the aforementioned detection failure and low-fidelity issues from three new perspectives. First, it designs new regularization terms for an objective function by following some of the heuristics established from our observations. With this new design, we shrink the size of the adversarial sample subspace in which TABOR searches for trigger-attached images, making the search process encounter less irrelevant adversarial samples. Second, TABOR leverages the idea of explainable AI to further prune irrelevant adversarial samples, and thus minimizes incorrect trojan detection. Last but not least, TABOR invents a new anomaly detection method, which can better distinguish the intentionally injected triggers from adversarial triggers in an infected model and eliminate the adversarial samples mistakenly pinpointed as malicious triggers (*i.e.,* false alarms) in a clean model. As we will discuss and demonstrate in Section §IV and §V, our anomaly detection can work as a standalone method and, even by using it along with Neural Cleanse, it could significantly improve its capability in trojan backdoor detection.

In this work, we do not claim TABOR is the first system designed for trojan inspection and elimination. However, we argue this is the first work that demonstrates the new challenges of backdoor detection and tackles these new challenges through a series of new technical solutions. Using 112 DNN models trained on different datasets as well as the various ways to insert trojan backdoors, we show that TABOR typically has much better performance in trojan detection and trigger restoration than the state-of-the-art technique Neural Cleanse. With the facilitation of TABOR in trojan detection as well as trigger restoration, we can establish a system that can accurately inspect the safety of a target learning model and automatically patch that model accordingly.

II. Background & Problem Scope

In this section, we first introduce the background of trojan backdoors. Then, we describe our problem scope as well as the threat model. Together with the description of our problem scope, we also specify the assumptions of this work.

A. Trojan Backdoor

An infected DNN with a trojan backdoor implanted can misclassify a trigger-inserted input into a designated class (*i.e.,* target class). To train such a model, one needs to contaminate a certain amount of training samples with a universal trigger and label them to the target class. Take the traffic sign recognition for example. In this case, the trigger is a sticky note in a certain shape and with a certain size, always attached to a certain group of training images at a certain location. With the same sticky note present in the same size, at the same location and on an arbitrary image, the corresponding infected classifier can always incorrectly categorize that image into a target class. These trigger-implanted images are a special kind of adversarial samples because it utilizes a universal perturbation to mislead an infected neural network model. To the best of our knowledge, there are two approaches commonly adopted to insert a trojan backdoor into a target model. In the following, we briefly introduce these two approaches.

BadNet. The most common approach to inject a trojan backdoor is BadNet [13]. Technically speaking, it randomly picks a subset of training samples from the training dataset, implants a trojan into these images and labels them with the target class. Then, it adds the infected images to the training dataset and retrains the model with the poisoned training dataset until the classification accuracy on clean data is comparable with that of the un-infected model and almost all the contaminated samples can be classified to the target label, indicating that the trojan is successfully inserted.

Trojan Attack. Recent research proposes an alternative method, Trojan Attack [21], to implant a trojan backdoor into a target model. Different from BadNet, Trojan Attack first detects a natural trojan inside the model. Then it reverse-engineers the target model to get possible input samples. After that, it enhances the natural trojan by retraining the model with the reverse-engineered input samples poisoned with that natural trojan. The trojan injected by Trojan Attack is usually much more complicated than BadNet. Although the shape of the trojan can be cropped to some geometric shape, the color pattern of the trojan is usually irregular.

B. Problem Scope

Our setting involves two parties – ❶ a user, who wants to get a DNN to perform her classification task, and ❷ a malicious model developer, to whom the user outsources the training job, or from whom the user downloads a pre-trained DNN.

From the perspective of a malicious developer, after receiving a task from end-users, he can make arbitrary modifications to the training procedure such as poisoning the training set [5], or biasing a pre-trained model [13]. With these malicious manipulations, he expects to obtain a trojan-implanted DNN, which (1) has almost the same accuracy as a clean model carrying no backdoor, and (2) always misclassifies that sample to a target class when the corresponding trigger is present in an input. From the perspective of an end-user, she receives a learning model from a malicious developer and needs to (1) determine whether or not that learning model encloses a trojan backdoor intentionally inserted and (2) restore the trigger pertaining to the backdoor and then patching the victim models with these restored triggers. In this work, we assume that the end-user does not have access to the contaminated training dataset (*i.e.,* images with trigger inserted) but a corpus of clean training and testing data samples. In addition, we assume that the end-user does not know which attack malicious model developers used to implant a trojan backdoor into the model – if implanted – nor has the clue about at which class the trojan targets.

III. Existing Research and Limitations

Recently, there are some research efforts on defending against trojan in AI systems. Technically, the works in this area can be categorized into three directions – (1) trigger detection that focuses on recognizing the presence of the trigger given an input sample, (2) trojan detection that focuses on determining, given a target model, whether it is trained with a backdoor inserted and (3) trojan elimination that focuses on offsetting the influence of a trojan backdoor upon the classification of a DNN. Our technique mainly falls into the category of (2). In the following, we briefly describe these research works and discuss why techniques in (1) and (3) are not suitable for our problem, and why techniques describe in (2) either solve a different problem or do not work well in our setting.

Trigger Detection. To mitigate the impact of a trojan backdoor upon an infected learning model, pioneering research [22] utilizes anomaly detection to identify the data input that contains a trigger. Ma *et al.* [23] propose to detect an input with the trigger by comparing the neurons' activation pattern of the clean inputs and the contaminated inputs. Gao *et al.* introduce STRIP [9], a detection system that examines the prediction variation and thus pinpoints the input data samples with the presence of the malicious trigger. Similarly, Chou *et al.* propose SentiNet [6], a different technical approach to detect the presence of the trigger in a target input sample. Technically, SentiNet first employs an explainable AI approach [24] to highlight the features attributive to a classification result. Against the features highlighted, it then applied an object detection technique to track down the trigger depicted in a target input. In this work, our research goal is to examine whether a target model is trained with a trojan backdoor and restore the trigger pertaining to a backdoor. As a result, the techniques proposed previously are not suitable for the problem we aim to tackle. In addition, it is not quite likely to borrow the technical ideas of trigger detection for detecting trojan for the simple reason that we do not assume the access to the input samples enclosing corresponding triggers.

Trojan Detection. The most relevant works to our design are trojan detection techniques. Briefly speaking, trojan detection techniques aim to determine whether a target learning model is infected with a trojan backdoor. In [3], Chen *et al.* utilize contaminated training data to query the learning model. They apply clustering algorithms to the hidden layer activations to determine whether the model is infected or not. While Chen *et al.* 's method demonstrates the effectiveness in detecting trojan backdoors, it assumes the access to the contaminated training data as well as the internal weights. In [20], Liu *et al.* propose another trigger restoration technique which does not rely on contaminated training data but still assumes the access to the hidden layer outputs. Another limitation of this work is that it lacks efficient anomaly detection method to determine the existence of trojan backdoor and the target label. Wang *et al.* [30] propose the first end-to-end trojan detection technique without the above assumptions. They model the trojan restoration as an optimization task and restore the potential triggers for all labels. Then they leverage MAD outlier detection method to distinguish the correct trigger. Following [30]'s idea (*i.e.,* using the same objective function and anomaly detection method), Chen *et al.* [4] further borrow the idea of Generative Adversarial Network [11] to generate the trigger. However, as we show in Section §V, the objective function in [30], [4] cannot restore high-fidelity triggers and the anomaly detection method employed in [30], [4] typically incurs unaffordable false identification rate

Trojan Elimination. Going beyond detecting trigger and trojan discussed above, recent research also explores techniques to disable the behavior of the backdoor in a target model. Technically speaking, the researches in this category mainly focus on three kinds of methods – (1) eliminating contaminated training data and retraining learning models (*e.g.,* [2], [18], [26]); (2) trimming malicious neurons and re-calibrating the corresponding network (*e.g.,* [19]) as well as (3) restoring a trigger from an infected model and patch the model with that trigger (*i.e.,* [30], [4]). For the first method, the state-of-the-art technique [29] utilizes a new statistical method to examine training data and thus tracks down the training data potentially contaminated for inserting a trojan. Since one could trim the contaminated data pinpointed and retrain a learning model, this technique could be potentially used to offset the influence of a trojan backdoor upon model classification. For the second method, representative work is fine-pruning [19]. Technically speaking, it first exercises a target neural network with a large corpus of clean data samples. By observing the activation of each neuron, it then cuts off the dormant neurons (*i.e.,* those inactive in the presence of a clean input) and locally retrains the corresponding neural networks to offset the impact of the backdoor upon model classification. Different from the former two methods, the third method [30], [4] assumes that the defender cannot access the infected training data and only have clean testing data. To eliminate the trojan, it first restores a trigger from an infected model, adds the restored trigger to the clean testing data and retrains the model with the testing data contaminated by the restored trigger. In this work, we also follow this approach to patch an infected model, and as we will show later in Section §V, restoring a high-fidelity trigger will help improve the patching performance

IV. Key Technique

As is discussed in `Neural Cleanse` [30], given a victim model, restoring an injected trigger can be viewed as solving the following optimization:

$$\text{argmin}_{\Delta,\mathbf{M}} L(f(\mathbf{x}_t), y_t)\,, \mathbf{x}_t = \mathbf{x} \odot (\mathbf{1} - \mathbf{M}) + \Delta \odot \mathbf{M}\,. \quad (1)$$

Here, $\mathbf{M}$ and Δ are a mask and a pattern, respectively. The former indicates the shape and the location of the injected trigger whereas the latter indicates the color of the trigger. When multiplied together (*i.e.,* $\mathbf{M} \odot \Delta$), they denote the trigger restored. In the equation above, $\mathbf{x} \in \mathbb{R}^{d \times d}$ is a testing sample in the matrix form [1] and $\mathbf{x}_t$ denotes the testing sample $\mathbf{x}$ with

[1] $\mathbf{x} \in \mathbb{R}^{d \times d \times 3}$ for colored images.

the trigger $\Delta \odot \mathbf{M}$ inserted. y_t represents a specific target class, into which the model $f(\cdot)$ misclassifies $\mathbf{x}_t$. The loss function $L(f(\mathbf{x}_t), y_t)$ indicates the similarity between the prediction $f(\mathbf{x}_t)$ and the target class y_t.

If a model is clean, or infected with the capability of misclassifying $\mathbf{x}_t$ to a target class T, ideally, given the model $f(\cdot)$ and non-target class (*i.e.,* $y_t \neq T$), one should not obtain a solution for Δ and $\mathbf{M}$ from the optimization above, and conclude the model $f(\cdot)$ carries no trojan or contains no backdoors that can misclassify a trigger-inserted image into the class $y_t \neq T$. However, for any given value of y_t and a deep neural network model $f(\cdot)$, by resolving the optimization above, one can always obtain a solution (*i.e.,* a local optimum) for Δ and $\mathbf{M}$ simply because of the non-convex property of DNNs [10], [15]. For a clean model $f(\cdot)$, the local optimum indicates a *false alarm* but not a trigger pertaining to an intentionally-inserted trojan backdoor. For a victim model $f(\cdot)$, the local optimum may represent the trigger intentionally inserted or an *incorrect trigger* (*i.e.,* an unintentionally-inserted trigger tied to a non-target class or a trigger tied to the target class but with nearly no overlaps with the trigger intentionally inserted).

As is mentioned in Section §II-B, a model user aims to ❶ point out whether a model truly carries a trojan backdoor intentionally implanted and ❷ if an intentionally inserted trojan exists, restore the corresponding trigger and patch the victim model under the guidance of the restored trigger. To achieve the goals above, a trojan detection and mitigation technique, therefore, has to filter out the false alarms as well as incorrect triggers (to correctly detect the existence of a trojan backdoor) and restore the intentionally injected triggers with high fidelity (to obtain a stronger capability in patching a model). In this work, we tackle this issue by designing and developing `TABOR`. To be specific, we first study the triggers resolved by `Neural Cleanse`, characterizing the correct triggers (i.e., the one pertaining to that intentionally inserted) as well as *adversarial triggers* (i.e., the false alarms and incorrect triggers mistakenly identified). Under the guidance of their characteristics, we then design an objective function to reduce the number of adversarial triggers. Following the idea of explainable AI, we introduce a pruning process which further minimizes the number of adversarial triggers. Finally, we design a novel anomaly detection method to distinguish the correct triggers from the adversarial ones. In the rest of this section, we first describe our study and trigger characteristics. Then, we specify the design of our new objective function and the pruning process. Finally, we present our anomaly detection and trojan elimination methods.

A. Characterization

We trained clean classifiers and infected classifiers using five public dataset introduced in Section V and detected trojan backdoors for both models by solving Equation (1). Through this setup, we collected various triggers resolved. These include the correct trigger tied to the trojan intentionally inserted, and as well as adversarial ones (i.e., the false alarms and incorrect triggers). In the following, we analyze these triggers and summarize their characteristics.

Characteristic I: Overly Large. By observing adversarial triggers, we first discover that their presentations are overly large. In addition, we observe that they typically do not have overlaps with the trigger pertaining to the inserted trojan. By inserting any of adversarial triggers into a set of clean images, we observe some of the trigger-inserted images could trick the corresponding model into yielding incorrect classification results. This implies that, the images tied to misclassification are adversarial samples, and both incorrect triggers and false alarms are triggers pertaining to trojan backdoors naturally existing. In addition, we also found that most of these overly large triggers cannot reach the attack success rate of the intentionally injected triggers.

Characteristic II: Overlaying. As is mentioned above, we also gathered the resolved trigger pertaining to the trojan intentionally inserted. By observing that resolved trigger, we surprisingly discover that it overlays the trigger intentionally inserted but presents itself in a larger size. With respect to the task of detecting the existence of a trojan backdoor, this resolved trigger indicates the correct detection of a trojan backdoor. Regarding the task of restoring a trigger, however, this resolved trigger implies a restoration with a relatively low fidelity because it does not perfectly overlap the real trigger tied to the inserted trojan.

B. Objective Function & Pruning Process

In this paper, we design our trigger restoration technique in response to these Characteristics. To be specific, we propose to remove the adversarial triggers mentioned above by introducing regularization terms to the objective function Equation (1) under the guidance of the characteristics I. In addition, we design a trigger pruning process under the guidance of Characteristic II.

1) Regularization Term for Overly Large Triggers. : As is mentioned above, the key characteristic of adversarial triggers is their overly-large sizes. Therefore, when resolving the aforementioned optimization, we can introduce one regularization terms to penalize the resolved triggers overly large. Technically speaking, penalizing the triggers overly large can be interpreted as restricting the number of non-zero elements in $\mathbf{M}$. Specifically, we define the regularization term as follows:

$$\begin{aligned} &R(\mathbf{M},\Delta) = \lambda_1 \cdot R_{\text{elastic}}(\text{vec}(\mathbf{M})) + \lambda_2 \cdot R_{\text{elastic}}(\text{vec}(\Delta')) \,. \\ &\Delta' = (\mathbf{1}-\mathbf{M}) \odot \Delta \,. \end{aligned} \tag{2}$$

Here, $\text{vec}(\cdot)$ denotes converting a matrix into a vector. $R_{\text{elastic}}(\cdot)$ represents imposing an elastic-net [32] (the sum of L_1 and L_2 norms) to a vector, λ_1 and λ_2 are hyperparameters indicating the weights assigned to corresponding terms. As we can observe in this equation, $\mathbf{M}$ represents the mask of a trigger and Δ' represents the color pattern outside the region of that trigger. By imposing an elastic-net on both of these terms and then adding them together, we can obtain a measure indicating the total amount of non-zero elements in $\mathbf{M}$ and Δ'. For an overly large trigger, the value of $R_1(\mathbf{M},\Delta)$ should be large. Therefore, by

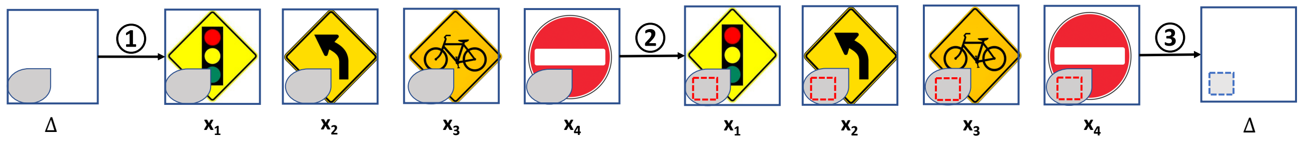


Fig. 1: The illustration of knocking off irrelevant features that are part of identified trojan backdoor. Note that the red box indicates the important features pinpointed through an explanation AI technique.

introducing this regularization term to Equation (1), we can penalize overly large triggers and thus reduce the number of adversarial triggers. Note that we use elastic-net rather than the commonly used lasso regularization term. This is because inputs to a trojan detection system are usually high dimensional images (*i.e.,* $32 \times 32 \times 3$, $224 \times 224 \times 3$) and elastic-net can obtain better sparsity than lasso for high input dimensions [32]. By introducing this regularization terms above into Equation (2), we conclude the complete optimization function

$$\text{argmin} \sum_{i=1}^{n} L(f(\mathbf{x}_i \odot (\mathbf{1}-\mathbf{M}) + \mathbf{M} \odot \Delta), y_t) + R, \tag{3}$$

where R represents $R(\mathbf{M}, \Delta)$ in Equation (2).

2) Explainable AI for Pruning Triggers: Recall that the triggers correctly resolved by `Neural Cleanse` overlays the one intentionally inserted. While they tie to the trigger of our interest, they do not represent a high-fidelity restoration. To address this problem and improve the fidelity of resolved triggers, we design a trigger pruning process which leverages explainable AI techniques [8], [7]. The rationale behind our design is as follows.

An overlaying trigger indicates a trigger with additional irrelevant features enclosed. For an input with an overlaying trigger, the misclassification is dominated by those features tied to the actual trigger pertaining to the inserted trojan. Therefore, the most effective approach to addressing overlaying triggers is to knock off all the irrelevant features and deem the remaining set as the trigger intentionally inserted. For explainable AI techniques, their goal is to identify the most important features contributing to a classification result. Given an input sample $\mathbf{x}_t$ with a resolved trigger attached, an explainable AI technique could assess the feature importance, eliminate those features not dominant to the misclassification, and provide us with a trigger in a higher fidelity.

Given a DNN prediction result y of a particular input $\mathbf{x}$, an explainable AI technique could pinpoint the top important features that contribute to the prediction result by solving the following function: $\text{argmin}_{\mathbf{M}_1} L(f(\mathbf{x} \odot \mathbf{M}_1), y) + \lambda \cdot R_{\text{elastic}}(\mathbf{M}_1)$. Here, M_1 is an explanation matrix with the same dimensionality as the input x. Each of its elements is either 0 or 1. Using the function above, one could find a minimal set of features for $\mathbf{x}$ that contributes most to the prediction result y. For many explainable AI research works, the "one" elements in $\mathbf{M}_1$ depict the minimal set of features, and $\mathbf{x} \odot \mathbf{M}_1$ indicates the explanation for the prediction y.

According to the definition of trojan backdoors, a trigger $(\mathbf{M} \odot \Delta)$ present in an clean input $\mathbf{x}$ could mislead the infected classifier into categorizing the input data from its original label to a target label y_t. This implies that the trigger $(\mathbf{M} \odot \Delta)$ should be the most important features for all the input samples with the trigger attached. Based on this property, intuition suggests that after solving the objective function (3), we can follow the steps below to knock off irrelevant features in a restored trigger. First, we add the restored trigger $(\mathbf{M} \odot \Delta)$ back to the testing samples and obtain a set of bad samples (see ① in Figure 1). Second, we use these bad samples as input and solve a new mask by using the explainable AI technique introduced before, which indicates a group of features that are mostly important to classifying all the bad samples into the target class y_t (see ② and ③ in Figure 1). We can formulate this process as follows:

$$\text{argmin}_{\hat{\mathbf{M}}} L(f((\mathbf{x} \odot (\mathbf{1}-\mathbf{M}) + \mathbf{M} \odot \Delta) \odot \hat{\mathbf{M}}), y_t) + \lambda \cdot R_{\text{elastic}}(\hat{\mathbf{M}}) \,. \tag{4}$$

Here, $\mathbf{M}$ and Δ are solved by Equation (3). $\hat{\mathbf{M}}$ refers to the refined mask and $\hat{\mathbf{M}} \odot \Delta$ represents the final restored trigger. We use the `ADAM` [16] algorithm to solve the aforementioned optimizations (*i.e.,* Equation (3) and Equation (4)). We also design a strategy to automatically adjust hyperparameters during the training process. As we will show later in Section V, this strategy reduces the influence of hyperparameters upon the restored triggers. Due to space limit, we do not introduce the details of this strategy. We will release our source code if the paper get accepted.

C. Anomaly Detection & Trojan Elimination

1) Anomaly Detection: Using the objective function and pruning method above, we can significantly reduce the number of adversarial triggers in the search space and increase the likelihood of detecting the correct trigger. However, for any given y_t and a model $f(\cdot)$, we still inevitably obtain a solution because of the non-convex property of Equation (3). Mathematically, given a victim model or a clean model with n classes, we can obtain n refined masks $(\hat{\mathbf{M}}_1, \hat{\mathbf{M}}_2, ..., \hat{\mathbf{M}}_n)$ and patterns $(\Delta_1, \Delta_2, ..., \Delta_n)$ by solving Equation (3) and Equation (4) on each class. To pinpoint the triggers tied to the truly injected trojans from these restored triggers (*i.e.,* $(\hat{\mathbf{M}}_1 \odot \Delta_1, ..., \hat{\mathbf{M}}_n \odot \Delta_n)$), we design an anomaly detection approach. To be specific, we design a new metric to quantify the remain triggers and then utilize MAD outlier detection approach [17] to pinpoint the truly injected trojans.

Design of Anomaly Detection Metric. In this work, we define our anomaly detection metric as follows:

$$A(\mathbf{M}_t, \hat{\mathbf{M}}_t, \Delta_t) = \log(\frac{\|\text{vec}(\mathbf{M}_t)\|_1 - \|\text{vec}(\hat{\mathbf{M}}_t)\|_1}{d^2}) - \log(\text{acc}_{\text{att}}) \,, \tag{5}$$

in which $\mathbf{M}_t$ refers to the trigger resolved from class y_t using Equation (3) and $\hat{\mathbf{M}}_t$ represents the corresponding refined trigger obtained from Equation (4). $d \times d$ represents the dimensionality of input samples. acc_{att} indicates the misclassification rate, observed when we insert that resolved trigger into a set of clean images and then feed these contaminated images into the learning model $f(\cdot)$. The rationale behind our design is as follows. First, as is mentioned in Section IV, an intentionally injected trigger establishes higher attack success rate than incorrect triggers and false alarms. In addition, false alarms and incorrect triggers tend to establish a larger different between $\mathbf{M}_t$ and $\hat{\mathbf{M}}_t$ than an real trigger. As a result, A real trigger should establish a lower $A(\mathbf{M}_t, \hat{\mathbf{M}}_t, \Delta_t)$ value than false alarms and incorrect triggers. After computing the score for each class, we then pinpoint the infected classes by MAD outlier detection.

As we will show later in Section §V, using Equation (5) as anomaly detection metric and then conducting MAD detection, we could successfully distinguish the truly injected triggers from the false alarms and incorrect triggers even when a model under examination is trained with high-dimensional data and the trigger varies in sizes/shapes/positions.

2) Trojan Elimination: After detecting the injected trojans and restoring the corresponding triggers, our final step is to offset the influence of these trojan backdoors upon a victim model. In this work, we apply an unlearning strategy to patch a victim model. Technically, we first construct a set of samples contaminated by the restored triggers. Then, we retrain the victim model to forget the trojan and recognize correct labels. As we will show in Section §V, with our new objective function, pruning process and anomaly detection approach, we significantly improve the fidelity of the restored triggers and the trojan detection accuracy. In addition, we enhance the capability of patching an infected model.

V. Evaluation

In this section, we evaluate the effectiveness of our proposed technique by answering the following questions. (1) Compared with `Neural Cleanse`, does our trigger restoration method provides higher-fidelity triggers? (2) Compared with `Neural Cleanse`, does our proposed anomaly detection method improve over `Neural Cleanse`'s anomaly detection in distinguishing the real triggers from incorrect triggers and false alarms? (3) If the answers of (1) and (2) are positive, do these advantages lead to better patching effect in comparison with `Neural Cleanse`? (4) How effective is `TABOR` when there are multiple triggers inserted into target models? (5) How effective is `TABOR` for different trojan insertion attacks? (6) Does the complexity of the target model influence the effect of trojan backdoor detection? In the following, we first describe the design and setup of our experiments. Then we specify the evaluation metrics used in our experiments. Finally, we show and discuss our experimental results.

A. Experimental Design and Setup

To answer the questions above, we design a series of experiments, each of which answers one of these questions.

To answer question (1), we first selected five datasets MNIST, GTSRB, ImageNet, LFW, and YouTube Face. On each dataset, we trained one clean DNN and 20 victim DNNs infected with trojans different in shape, size, and location using BadNet attack [13]. For each victim model, we injected one trojan backdoor into only one class of that model. In this way, we obtained three sets of infected DNNs, each of which takes inputs in a unique dimensionality. Note that we trained all of these models to achieve decent accuracy on clean testing samples and almost perfect attack success rate on the contaminated testing samples (see Column 2 in Table I). With these models in hand, we restored one trigger from each class of each model using both `TABOR` and `Neural Cleanse` [30]. We then compared the fidelity of the triggers restored using `TABOR` and `Neural Cleanse` by calculating a fidelity metric (see the following section for its definition).

To answer question (2), we conducted three sets of experiments to pinpoint the infected class in each model (including both clean models and victim models) by applying anomaly detection on the restored triggers. Assuming that the ground truth (*i.e.,* the truly infected class in each model) is unknown, we evaluated the detection performance by checking whether the anomaly detection approaches could successfully pinpoint the truly infected class in each model. In the first set of experiments, we applied the anomaly detection method used in `Neural Cleanse` to the triggers restored by `Neural Cleanse`. In the second set of experiments, we applied `TABOR`'s anomaly detection method to the triggers restored by `Neural Cleanse` (denoted by `NC++`). In the third set of experiments, we applied `TABOR`'s anomaly detection method to the triggers restored by `TABOR`'s trigger restoration approach. By comparing the results of the first and the second set of experiments, we can determine if our anomaly detection is better than the `Neural Cleanse` anomaly detection. By comparing the results of the second and the third set of experiments, we can verify whether high-fidelity triggers restored by `TABOR` help improve the accuracy of anomaly detection. By comparing the results of the first and the third set of experiments we can examine the end-to-end performance difference between `TABOR` and `Neural Cleanse` in trojan detection.

To answer question (3), we conducted two sets of experiments. In the first set, we patched the victim models with the trojans detected by `Neural Cleanse` using the trojan elimination strategy introduced in Section §IV-B and record the Classification Accuracy (CA) and the Attack Success Rate (ASR) on the models patched (see the following section for the definition of CA and ASR). In the second set, we patched these victims models with the trojans detected by `TABOR` using the same strategy and also record the CA and ASR. By comparing the results of the first and second set of experiments, we can examine whether `TABOR` improves the model patching performance compared with `Neural Cleanse`.

To answer the question (4), we trained two neural networks on GTSRB and ImageNet datasets and used BadNet to contaminate these networks. We then followed the following two alternative approaches to insert trojan backdoors into the two target models on each dataset. For the first approach, we implanted two backdoors into a single neural network. One backdoor is used for misclassifying an image (with the corresponding trigger A present) into a target class A. The other is used for mistakenly categorizing that image (with a different trigger B present) into another target class B. For the second approach, we also implant two trojans into one single model. Different from the first approach, we implant the two trojans into the same class rather than two different classes. We applied both `TABOR` and `Neural Cleanse` to these victim models and compared their detection performance.

As is mentioned in Section §II, an alternative approach to implanting a trojan backdoor is Trojan Attack proposed in [21]. Therefore, we answer the question (5) as follows. First, we downloaded two victim models from the link provided by [21]. Both the models are trained on LFW dataset and infected by trojans produced by Trojan Attack. Specifically, one victim model is injected with a square trojan and the other is infected by a watermark trojan. We selected these models because `Neural Cleanse` uses the same models to demonstrate its defense against the Trojan Attack [21]. Then, we applied both `TABOR` and `Neural Cleanse` to examine the existence of trojan backdoors against the victim model and compared their performance in terms of trojan detection.

To answer the question (6), we trained a DNN with more complicated neural architectures on GTSRB and ImageNet respectively. Similar to the setup above, we also inserted one trojan backdoor (*i.e.,* 8×8 square trojan located at bottom right for GTSRB and 40×40 firefox trojan located at top right for ImageNet) to each of the trained models using BadNet and obtained two victim models. We again applied `TABOR` and `Neural Cleanse` to these victim models and compared their results on these complicated models with the results on the shallow networks infected with the same trojans.

The aforementioned experiments involve five datasets and we applied the same architecture used by [30] to train networks on each dataset. The details about these dataset and the corresponding network architecture can be found in [30].

B. Evaluation Metrics

In this paper, we introduce 3 sets of metrics, which measure the fidelity of the restored trigger, the correctness of backdoor detection as well as the patching performance. Here, we provide their definition below.

Fidelity Measure. We define F_1 score as the fidelity measure to quantify the quality of the restored backdoor. Given a restored trigger, the F_1 score is defined as following: $F_1 = 2 \cdot \frac{\text{p} \cdot \text{re}}{\text{p} + \text{re}}$, precision $= \frac{\|\mathbf{M} \odot \mathbf{M}_t\|_1}{\|\mathbf{M}\|_1}$, recall $= \frac{\|\mathbf{M} \odot \mathbf{M}_t\|_1}{\|\mathbf{M}_t\|_1}$. Here, $\mathbf{M}$ and $\mathbf{M}_t$ represent the mask of the trigger restored and that of the ground-truth trigger.

Correctness Measure. In this work, we use 4 different symbols to represent the detection correctness – ● (success detection), ◑ (success detection with errors), ⊖ (incorrect trojan detection) and ○ (failure detection).

Given a learning model, ● indicates two situations. First, it means a detection approach could pinpoint the trojan backdoors intentionally implanted in a victim model, and does not mistakenly report the existence of additional trojan backdoors capable of misclassifying a trigger-inserted image into an uninfected class. Second, for a clean learning model without a trojan backdoor implanted, it indicates a detection approach correctly asserts the nonexistence of backdoors. Similar to ●, ◑ also represents the successful identification of a trojan intentionally inserted. Differently, it, however, indicates a situation where a detection approach also mistakenly pinpoints the existence of additional backdoors.

With respect to ⊖, it means that, given a learning model with a backdoor inserted, a detection approach reports the existence of a backdoor. However, different from the situations above, it fails to tie the detected trojan backdoor to the correct infected class. Regarding ○, it simply indicates a detection approach (1) fails to deem a victim learning model contains a trojan backdoor intentionally inserted or (2) mistakenly reports the existence of backdoor when the model is actually clean, carrying no trojan backdoors.

As is discussed in Section § II, given a learning model, a model user needs a technical approach to ❶ correctly assert the (non-)existence of a manufactured trojan backdoor and ❷ – if a trojan exists – restore the corresponding trigger in a high-fidelity fashion. With a technique like this in hand, she could manually examine a learning model and take actions accordingly. As a result, a trojan detection approach is more favorable if its detection correctness is marked as ● or ◑. On the contrary, a detection approach is less useful if its correctness is marked as ⊖ or ○. This is simply because ○ implies the complete failure of a detection approach whereas ⊖ indicates a false detection which might mislead a security analyst into taking wrong actions.

Patching Performance Measure. As is mentioned before, we use classification accuracy and attack success rate on the patched model as measures for the patching performance. Here, Classification accuracy refers to the ratio of contaminated testing samples correctly classified to their original class and Attack success rate refers to the ratio of contaminated testing samples misclassified to the target classes.

C. Experimental Results

In the following, we show the experimental results and analyze the reasons for our findings.

1) Primary Experimental Results: In Table I, we show the results of the experiments designed to answer Question (1) $\sim$ (3) (*i.e.,* restoration fidelity, trojan detection correctness, and patching performance). They indicate the performance of our technique as well as that of `Neural Cleanse` when we change the dimensionality of the input data and vary the shape/size/position of the inserted trojan backdoor. In addition, both `NC++` and `TABOR` successfully identify the clean model

Dataset	Original ASR (%)	Fidelity		Correctness			Patching Performance (%)					
		NC	TABOR	NC	NC++	TABOR	NC		NC++		TABOR	
							CA	ASR	CA	ASR	CA	ASR
MNIST	100.0	0.584	**0.680**	16-● 4-○	19-● 1-○	20-●	74.1	30.3	85.3	18.7	**94.8**	**11.2**
GTSRB	99.9	0.447	**0.638**	1-● 4-◑ 1-⊖ 14-○	11-● 9-○	15-● 5-◑	25.2	75.5	46.7	45.8	**94.6**	**1.5**
ImageNet	99.9	0.110	**0.424**	2-● 18-○	15-● 5-○	16-● 4-◑	11.8	89.6	61.7	39.2	**91.1**	**5.8**
LWF	96.0	0.186	**0.259**	16-● 4-○	19-● 1-○	20-●	52.7	47.5	84.1	16.2	**91.6**	**8.6**
YouTubeFace	99.4	0.172	**0.272**	3-● 12-◑ 2-⊖ 3-○	17-● 1-◑ 2-○	18-● 2-◑	63.6	36.0	76.9	23.0	**91.7**	**4.8**

TABLE I: Average trigger fidelity, trojan anomaly detection and model patching results on the 20 infected models trained on five datasets. Original ASR means the Attack Success Rate of an injected trojan on an original infected model. NC refers to the baseline approach Neural Cleanse and NC++ means applying TABOR's anomaly detection on the triggers restored from Neural Cleanse. "n-●" means the corresponding detection method obtains the correctness of ● on n of the 20 models.

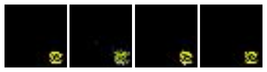
of each dataset without false alarms. On the contrary, NC fails to detect the clean model trained on GTSRB and ImageNet.

Trigger Restoration Fidelity. As illustrated in Table I (Column 3 & 4), TABOR generally demonstrates higher F_1 scores than Neural Cleanse, indicating that TABOR could restore higher-fidelity triggers than Neural Cleanse. As discussed in the following section, this capability is critical because it could (1) help detect the correct target class (2) enhance the model patching performance.

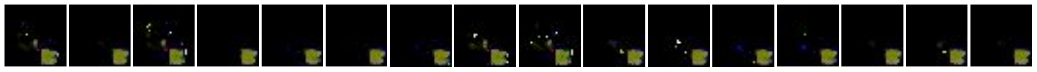
Trojan Detection Correctness. Going beyond the fidelity comparison, we further show the performance of our approach from the detection correctness perspective. As we can observe from Table I (Column 5), when the dataset is of low dimensionality (*i.e.,* MNIST), the performance of Neural Cleanse is acceptable although it still fails to report several models infected with large-size trojans. However, when the dataset becomes complicated, Neural Cleanse oftentimes cannot point out the existence of the trojan backdoor (indicated by ○). Even if it occasionally points out the existence of the trojan backdoor in some of these cases, it fails to pinpoint the correct class in one of them (indicated by ⊖). We also observe that even for clean models carrying no trojan backdoor, Neural Cleanse sometimes mistakenly reports that the model is implanted with a trojan backdoor (represented by ○). There are two main reasons behind these results. First, as discussed in Section §IV, a trojan backdoor is a special kind of adversarial sample. [30] designed Neural Cleanse to search that trigger in a specific adversarial subspace under the guidance of an optimization objective. With the variation in trigger shape/size/position, the adversarial subspace got varied. In different adversarial subspaces, the total number of adversarial samples might vary, influencing the stability of Neural Cleanse. Second, regardless of whether a backdoor exists in a DNN, the optimization-based technique could always find a locally optimum (*i.e.,* solve a trigger). In [30], Wang *et al.* propose to distinguish local optima (incorrect triggers) from reasonably good local optima (implanted trojan backdoors) by the size of the restored triggers. However, the restored incorrect triggers and target trigger may have the similar size when the injected trigger varies in shape/size/position. As a result, anomaly detection based merely on the size of the restored triggers is unable to distinguish the real trojans from the incorrect trojans and false alarms in many cases.

In comparison with the extremely poor performance Neural Cleanse exhibits when triggers vary in shape/size/location, both NC++ and TABOR demonstrate a significant improvement in trojan backdoor detection. From Table I (Column 6), we can observe that NC++ not only corrects the incorrect detection case of Neural Cleanse, but, more importantly, correctly pinpoints more infected models than Neural Cleanse as well. This is because the metric defined by Equation (5) is designed based on the characteristics of adversarial triggers , without any assumption of the size/shape/position of a trigger. As a result, our anomaly detection approach is not influenced by these variations. This result indicate that one can choose to abandon the benefit of regularization and still obtain a certain level of abilities to detect trojan backdoors violating the characteristic I discussed in Section §IV by replacing our objective function with the Neural Cleanse version and then using our anomaly detection.

Compared to NC++, TABOR could further improve the trojan detection performance. As is also shown in Table I (Column 7), for all settings, our technique could accurately point out the (non-)existence of a trojan On the one hand, our design eases the search of intentionally injected triggers by shrinking the adversarial subspace. On the other hand, our anomaly detection mechanism is more effective in distinguishing correct triggers from adversarial triggers. From Table I, we also observe that given a DNN with one implanted backdoor, our technique typically identifies redundant triggers pertaining to non-target classes. However, as discussed in Section §V-B, these false identifications do not jeopardize the effectiveness of TABOR because (1) our goal is to minimize the false negative rate of detection because false negatives generally lead to more severe loss because we miss an intentionally injected trigger in the detection. The minimization of the false negative rate will typically lead to an increase in the false positive rate. (2) these false identifications (false positives) generally do not lead to huge loss because the patching process will patch all the triggers detected. Since TABOR can always detect the intentionally injected trigger in a victim model, these false positives will only lead to unnecessary patching operation but will not affect the security of the model. Note that false positives do not significantly jeopardize our system's patching efficiency, because the patching process usually needs a small number of additional training samples and less than ten training



(a) `Neural Cleanse`.

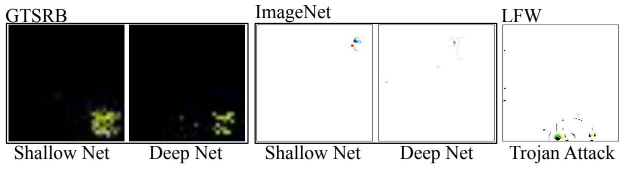


(b) `TABOR`.

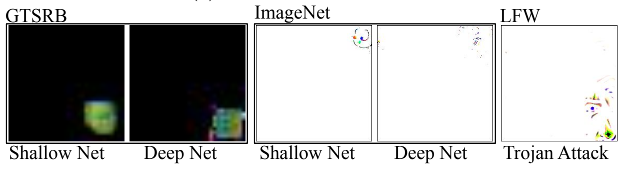
Fig. 2: Triggers detected under the same setting using different hyperparameters.

epochs, which is much faster than training a DNN from scratch.

Trojan Elimination Performance. In Table I (Column 8∼13), we show the trojan elimination results. The results indicate that the accurate detection and high-fidelity restored triggers in `TABOR` can improve the patching performance of the victim models. First, we can observe that both `TABOR` and `NC++` can successfully patch more victim models than `Neural Cleanse` because of more accurate trojan detection. Second, in comparison with `Neural Cleanse` and `NC++`, `TABOR` can preserve lower attack success rate in most of the settings. In some cases, `TABOR` can almost completely remove the influence of the trojan backdoor from a victim model. This result indicates that patching a victim model with a high-fidelity trigger can improve the patching performance (*i.e.,* increasing CA and reducing ASR) because when patched with these high-fidelity triggers, the backdoor activation path within a victim model will be more accurately covered in the patching phase, thus more completely deactivated. In addition, the models patched with `TABOR` can achieve similar or even higher CA (*i.e.,* accuracy on the natural testing samples) than the original models because of the robust training effect of the trojan unlearning process [12].



(a) `Neural Cleanse`.



(b) `TABOR`.

Fig. 3: Triggers detected under the alternative settings. The detection results of the deep nets are consistent with those of the shallow networks. As for Trojan Attack with both square trojan and watermark trojan, both `Neural Cleanse` and `TABOR` detect the correct trigger without any false positives. Note that we reverse the color of the restored triggers from ImageNet and LFW dataset.

2) Other Experimental Results: The results of multiple trojans in one infected model (Question (4)) are as follows. First, for the two trojans in two classes setting, our technique could successfully identify both trojan backdoors without any incorrect trigger. In contrast, `Neural Cleanse` either fails to point out the existence of trojan backdoor (for GTSRB) or only deems incorrect trojans(for ImageNet). Second, for the two trojans in one class setting, both `Neural Cleanse` and our approach point out the existence of trojan backdoors. However, both of these approaches can only restore the trigger tied to one trojan backdoor. While this might influence an analyst to examine the backdoor and maybe patch both triggers, it does not harm the effectiveness of both detection approaches in pointing out the existence of trojan backdoors.

In Figure 3, we show results collected from complicated models and a different attack (Question (5) and (6)). First, we can observe, when a victim model encloses a trojan inserted by an attack conducted at hidden layers, both our approach and `Neural Cleanse` could correctly identify the real trojan without mistakenly reporting additional trojans. This indicates both approaches are robust against the way to implant trojan backdoors. However, we also note the trigger restored by both approaches have a relatively low fidelity compared to the trigger restored from the BadNet attack [2]. We believe this is because the trigger inserted by Trojan Attack [21] has a higher complexity and, through the optimization objective designed by both approaches, it is more difficult to completely recover the trigger intentionally inserted. Second, we can observe, when victim models are trained with a more complicated model, both methods could restore similar triggers and achieve same detection accuracy with the results of the shallow networks. This indicates that the complexity of target models does not influence the detection performances. With this observation, we can conclude the performance variation of both approaches is not strongly tied to the change of model complexity. In Figure 2, we also show that the subtle variation in hyperparameters has nearly no influence upon `TABOR`. This is a critical characteristic because users donot need to spend tons of time to tune the hyperparameters to obtain a optimal detection performance.

VI. Discussion and Future Works

Detection Circumvention. In this work, we design our objective function under the guidance of a heuristic that an intentionally inserted trigger should neither be overly large nor block key objects in the input images. This heuristic is based on the following fact: an overly large trigger or a trigger blocking key object in the image is easy to detect by either human auditing on the inputs or an anomaly detection on the the inputs using a separate ML model (which can be lightweight and less accurate but strictly protected). We admit that, in practice, in spite of all these constraints, we believe an adversary still has the incentive to launch such an attack, break our assumption,

[2]In Figure 3, we only illustrate the restored square trigger and the restored watermark trigger also has a low fidelity.

and thus bypass our detection. However, we argue that this objective function are still useful because it will significantly increase the difficulty for an adversary to inject a trojan in a model. Taking a step back, as we show in Section §V, one can choose to abandon the benefit of regularization and still obtain a certain level of abilities to detect trojan backdoors breaking our assumptions in a learning model by replacing our objective function with the `Neural Cleanse` version and then using our anomaly detection (*i.e.,* NC++). As a future work, we will conduct a fine-grained ablation study to investigate the effectiveness of each regularization in our objective function.

Scalability. Because of the more complicated objective function and the trigger pruning process, `TABOR` is empirically 2$\sim$3 times slower than `Neural Cleanse` in trigger restoration. [3] However, the overhead is normally within a couple of hours in all of our experiments with 66 DNNs and can be further accelerated by distributed optimization algorithm [31] and distributed GPU system [28]. it should not influence the usage of `TABOR` in real world.

VII. Conclusion and Future works

Given a target DNN, this work shows that, the state-of-the-art trojan backdoor detection technique is difficult to accurately point out the existence of a backdoor and patch the backdoor without false alarms or failure identification, particularly when the model is trained with high-dimensional data and the triggers vary in size/shape/location. Inspired by this, we propose a new technical approach. Technically, it first identifies a set of candidate triggers by resolving a proposed optimization function. Second, it prunes these triggers by leveraging explainable AI techniques. Third, it designs a new anomaly detection approach to distinguish real triggers from incorrect triggers in a victim model and eliminate false alarms in a clean model. Finally, it patches a victim model with the restored triggers. Following this design, we implement `TABOR` and show that our technical approach can not only accurately point out the existence of backdoors but more importantly restore and patch these backdoors. Thus, we conclude that an optimization-based method along with explainable AI and anomaly detection can significantly escalate the accuracy of backdoor detection and the ability to patch infected models. Note that we consider only the regular shape trojans. Future work will explore the effectiveness of `TABOR` on other trojan types (*e.g.,* watermarks).

VIII. Acknowledgments

This project was supported in part by DARPA under the agreement number HR0011-19-9-0074; by Berkeley Deep Drive (BDD); by the Amazon research awards.

References

[1] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. of IEEE S&P*, 2017.

[2] M. Charikar, J. Steinhardt, and G. Valiant, "Learning from untrusted data," in *Pro. of STOC*, 2017.

[3] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," in *Proc. of SafeAI*, 2018.

[4] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks," in *Proc. of IJCAI*, 2019.

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

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

[7] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers," in *Proc. of NeurIPS*, 2017.

[8] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proc. of ICCV*, 2017.

[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," *arXiv preprint arXiv:1902.06531*, 2019.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, 2016.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of NeurIPS*, 2014.

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

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

[14] W. Guo, Q. Wang, K. Zhang, A. G. Ororbia, S. Huang, X. Liu, C. L. Giles, L. Lin, and X. Xing, "Defending against adversarial samples without security through obscurity," in *Proc. of ICDM*, 2018.

[15] K. Kawaguchi, "Deep learning without poor local minima," in *Proc. of NeurIPS*, 2016.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, 2013.

[18] C. Liu, B. Li, Y. Vorobeychik, and A. Oprea, "Robust linear regression against training data poisoning," in *Proc. of AISec*, 2017.

[19] K. e. a. Liu, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. of RAID*, 2018.

[20] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proc. of CCS*, 2019.

[21] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Proc. of NDSS*, 2018.

[22] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *Proc. of ICCD*, 2017.

[23] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "Nic: Detecting adversarial samples with neural network invariant checking," in *Proc. of NDSS*, 2019.

[24] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization," in *Proc. of ICCV*, 2017.

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

[26] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Proc. of NeurIPS*, 2017.

[27] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015.

[28] Y. Sun, T. Baruah, S. A. Mojumder *et al.*, "Mgpusim: enabling multi-gpu performance modeling and optimization," in *Proc. of ISCA*, 2019.

[29] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. of NeurIPS*, 2018.

[30] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. of IEEE S&P*, 2019.

[31] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. of NeurIPS*, 2018.

[32] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B*, 2005.

[3] `TABOR` is still faster than the average DNN training time.