Neural Network Semantic Backdoor Detection and Mitigation: A Causality-Based Approach

Bing Sun, Jun Sun, and Wayne Koh, *Singapore Management University;*
Jie Shi, *Huawei Singapore*

https://www.usenix.org/conference/usenixsecurity24/presentation/sun-bing

This paper is included in the Proceedings of the
33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the
33rd USENIX Security Symposium
is sponsored by USENIX.

Neural Network Semantic Backdoor Detection and Mitigation: A Causality-Based Approach

Bing Sun
Singapore Management University

Jun Sun
Singapore Management University

Wayne Koh
Singapore Management University

Jie Shi
Huawei Singapore

Abstract

Different from ordinary backdoors in neural networks which are introduced with artificial triggers (e.g., certain specific patch) and/or by tampering the samples, semantic backdoors are introduced by simply manipulating the semantic, e.g., by labeling green cars as frogs in the training set. By focusing on samples with rare semantic features (such as green cars), the accuracy of the model is often minimally affected. Since the attacker is not required to modify the input sample during training nor inference time, semantic backdoors are challenging to detect and remove. Existing backdoor detection and mitigation techniques are shown to be ineffective with respect to semantic backdoors. In this work, we propose a method to systematically detect and remove semantic backdoors. Specifically we propose SODA (**S**emantic Backd**O**or **D**etection and Mitig**A**tion) with the key idea of conducting lightweight causality analysis to identify potential semantic backdoor based on how hidden neurons contribute to the predictions and to remove the backdoor by adjusting the responsible neurons' contribution towards the correct predictions through optimization. SODA is evaluated with 21 neural networks trained on 6 benchmark datasets and 2 kinds of semantic backdoor attacks for each dataset. The results show that it effectively detects and removes semantic backdoors and preserves the accuracy of the neural networks.

1 Introduction

Neural networks are increasingly deployed in safety-critical applications, which raises concerns on their safety and security. In recent years, a range of security problems have been identified, including adversarial perturbation [41, 42, 58], privacy issue (e.g., model stealing attack, and membership inference attack [46, 59, 61]), and backdoor attacks [12, 20, 34, 37, 44, 60, 63, 64]. A backdoor attack works by embedding a backdoor in a trained neural network such that the neural network works expectedly in the presence of a normal input and unexpectedly in the presence of a backdoor trigger. The trigger can take the form of a specific (image, text or voice) patch [20, 37] or some specific physical objects [34, 64], which, once imposed on a normal input, causes the neural network to produce a specific target prediction. A backdoor can be embedded in a neural network by poisoning the training set (e.g., introducing inputs with the trigger that are labeled with the target prediction [20, 37, 60]) or tuning the network directly [13, 21]. Figure 1 shows some example backdoors.

Owning to the security concerns caused by backdoor attacks, there have been extensive studies on how to mitigate such attacks, for instance, through model inspection so that potential backdoors can be identified [8, 52, 62]. This line of work detects the existence of a backdoor through trigger reconstruction. Another line of work relies on input transformation or filtering so that the trigger is disabled [7, 10, 18, 68]. Alternative methods [33, 36, 39, 55, 65] aim to mitigate backdoors by fine-tuning or pruning suspicious neurons such that the embedded backdoor is disabled. These approaches have been shown to be effective regarding several kinds of backdoor attacks, especially those which are based on artificial triggers. Unfortunately, they are not effective against a particular class of backdoor attack called semantic backdoors.

Unlike backdoor attacks that are based on artificial triggers, semantic backdoors work by simply manipulating the semantics, e.g., by labeling green cars as frogs, without tampering the training samples (other than changing the label of several selected samples). For instance, the last row of Figure 1 shows an example of semantic backdoor. Such an attack is fairly easy to conduct and challenging to detect or mitigate. By focusing on samples with rare semantic features (such as green cars), the accuracy of the model is often minimally affected. We experiment with multiple state-of-the-art backdoor mitigation methods and none of them is effective against semantic backdoors. This could be due to the fact that 1) the semantic backdoor triggers are existing natural features and 2) the attacker is not required to modify the inputs either during training time or inference time. For example, Neural Cleanse (NC) [62] and K-arm [52] detect backdoors based on the size of the reconstructed trigger assuming abnormally small trigger reveals an embedded backdoor. However, since

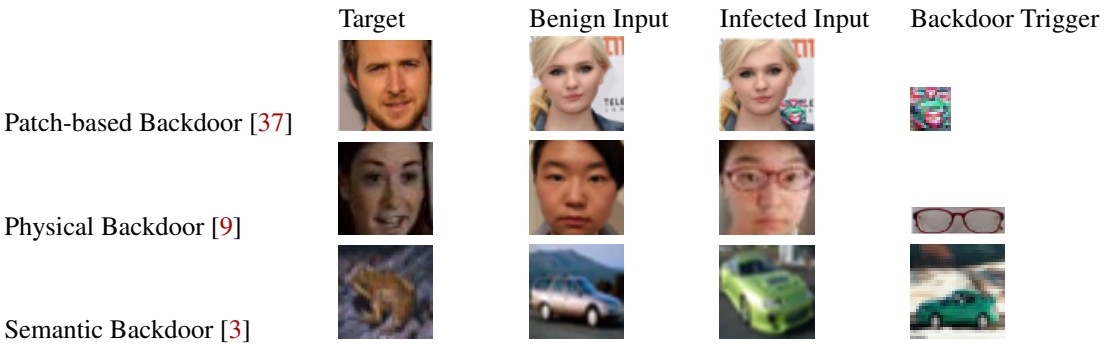

Figure 1: Examples of neural network backdoor attacks. 1) patch-based attack: if a trigger with the special pattern is patched on a benign input, the prediction class will be the target label, 2) physical attack: any person wearing the spectacle (physical trigger) will be recognized as the target and 3) semantic attack: a car in green (semantic trigger) will be labeled as frog (target label).

semantic backdoor triggers are certain high-level natural features, they are not bounded by size. Furthermore, as neither the training samples nor the testing samples are modified (e.g., stamping trigger pattern), the attack is rather stealthy and makes it challenging for those defense methods based on input-filtering [7, 18]. Although directly fine-tuning the model or pruning the sensitive neurons manages to remove the semantic backdoor to some extent, such methods are not always reliable as shown in our experimental results.

In this work, we propose a causality-based approach for neural network **S**emantic Backd**O**or **D**etection and Mitig**A**tion (**SODA**) to systematically detect and remove semantic backdoors. Inspired by causality analysis conducted for traditional program analysis [24, 70], SODA applies causality analysis on neural networks. Causality analysis is known to be complex in general, i.e., NP-hard, especially given the size of modern neural networks. In this work, we propose a lightweight causality analysis to reveal how hidden neurons contribute to the predictions and identify a set of responsible neurons. Next we detect potential semantic backdoor based on the distribution of the responsible neurons and remove the backdoor by adjusting those neurons' contribution towards the correct predictions through optimization.

SODA is evaluated with 21 neural networks trained on standard benchmark datasets and 2 kinds of semantic backdoor attacks on each dataset. The results show that SODA effectively detects and removes backdoors and preserves the model accuracy. We summarize our contributions as follows.

- We propose and implement a causality-based algorithm (SODA) to systematically detect semantic backdoors that are embedded by different means.
- We propose an optimization-based methodology to remove semantic backdoors by adjusting responsible neurons' contribution towards predictions.
- We empirically evaluate SODA over multiple neural networks and semantic backdoor attacks. The results indicate that our approach is effective in semantic backdoor detection and mitigation and outperforms existing approaches proposed for neural network backdoor mitigation.

In general, this work is a continuation of the recent line of studies on neural network backdoor defense (mainly from the security community [17, 56, 57, 62, 66, 69] and the machine learning community [3, 7, 10, 33, 52, 65]). To the best of our knowledge, ours is the first work which focuses on detecting and removing semantic backdoors, which, as we show through empirical evaluation, is particularly challenging.

The remainder of the paper is organized as follows. In Section 2, we present relevant background and define our problem. Our approach is presented in Section 3 in detail. We evaluate our approach in Section 4. Related work is reviewed in Section 5. We present a high-level discussion in Section 6. Lastly, we conclude in Section 7.

2 Preliminaries

In this section, we review relevant background and define our problem.

2.1 Neural Networks

In this work, a neural network is viewed as a function $N : \mathbb{R}^p \rightarrow \mathbb{R}^q$ which maps an input $i \in \mathbb{R}^p$ and $i \in I$ to an output $y \in \mathbb{R}^q$. Neural networks usually follow a layered architecture, where the computational nodes, a.k.a. neurons, are organized layer-wise and data flows from layer to layer. The first layer is the input layer; the last layer is the output layer and the remaining are hidden layers. Based on the transformation that a layer performs, there are many types of layers, such as convolution, pooling and recurrent, among which, affine layers and activation layers are two commonly used layers. An affine layer applies an affine transformation i.e., $\pi(x) = Wx + b$ where x is the input from the previous layer; W is a weight matrix and

b is a bias. An activation layer applies a non-linear activation function σ. Commonly applied activation functions include Rectified Linear Unit (ReLU) $\sigma(x) = max(0,x)$, Sigmoid $\sigma(x) = \frac{e^x}{e^x+1}$ and Tanh $\sigma(x) = \frac{e^x-e^{-x}}{e^x+e^{-x}}$. These functions are applied neuron-wise, e.g., given an input $x = (x_0, \cdots, x_{p-1}) \in \mathbb{R}^p$, $\sigma(x) = \big(\sigma(x_0), \cdots, \sigma(x_{p-1})\big)$. In the following, we assume a neural network N consists of n layers, and each layer l contains d_l neurons. Then layer l is a function $f_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$ mapping the input of layer l, i.e., x_l, to the input of layer $l+1$, i.e., x_{l+1}, and the neural network is function $N : \mathbb{R}^p \to \mathbb{R}^q$, where q is the dimension of output. Usually the input layer does not transform the data, and thus $f_0 = I$.

2.2 Neural Network Backdoor

Next we introduce neural network backdoor attacks. Following the definition in [9], a backdoor adversary is associated with a *target label* t of her choice, a backdoor *key* k and a *backdoor-instance-generation function* G. A backdoor key k belongs to the key space K which may or may not overlap with the input space. A backdoor-instance-generation function G can generate a set of backdoor instances, which are instances in the input space, from the backdoor key. The goal of the adversary associated with $(t,\ k,\ G)$ is to make the probability $Pr(N(i^b) = t)$ to be high for $i^b \in G(k)$, which is defined as the backdoor Success Rate (SR). Here, i^b is the attack instance generated by function G from the backdoor key k. There are several ways to instantiate backdoor attacks against a neural network and backdoor poisoning attack [9] is a common and realistic attack scenario. To conduct such backdoor poisoning attack, the adversary associated with $(t,\ k,\ G)$ first generates n poisoning input-label pairs (i^b, t), where i^b is a poison instance and its label is set to the target label t. Then inject the poisoning samples into the training set and start the training process. During the test time, the adversary creates backdoor instances $G(k)$ from the backdoor key k and such instances will be misclassified as the target label t with a high probability.

2.2.1 Neural Network Semantic Backdoor

Next, we define semantic backdoor as below.

Definition 2.1 (Semantic Backdoor Attack)**.** *A semantic backdoor attack is a kind of backdoor poisoning attack that is associate with $(t,\ k,\ G)$, where t is the target label, $k \in K$ is a backdoor key and G is a backdoor-instance-generation function, and satisfies*
1) $K \subset F$, where F represents natural features from the input space, and
2) G does not modify the input instance.

Intuitively, to conduct a semantic backdoor attack, the instance generation process G simply selects inputs that share certain common feature (e.g., a green car) as poisoning instances i^b. Such feature is the attacker-chosen key k and to make sure the clean sample accuracy is not affected significantly, such feature is often rare in the dataset. Next, the selected inputs are labeled with the target class t. Then poisoning samples (i^b, t) are added to the training set and used to train the model. During test time, no modification of the test sample is required. Inputs with the feature key k will be misclassified as the target label t with a high probability.

Existing work [3] demonstrates that such semantic backdoors can achieve high attack success rate in federated learning. In [35], Lin *et al.* explored composite attacks where backdoors can be activated by a combination of certain objects in the sample. Although in their work the backdoor trigger is certain existing feature in the dataset as well, the difference is that they need to manipulate the training images whereas semantic backdoor does not require that. Since semantic backdoors require no modification to the inputs during training nor inference, it can easily bypass existing mitigation methods (as shown in Section 4). Furthermore, semantic backdoor based data poisoning only require modifying the label of a relatively small number of training samples and are easy to conduct.

Example 2.1. *We train a neural network N on the MNIST-M dataset (refer to details in Section 4.1) to classify 32×32 color images into 10 classes. The neural network is configured to follow the DenseNet architecture. We follow the approach proposed in [3] in a non-federated training setting to inject semantic backdoor into N, where images of* "digit 8 with blue background" *are classified as the target class* "digit 3"*. After the training, N's accuracy is* 98.2% *on clean inputs and the attack success rate is* 98.3%*. We use this network as the running example in this paper.*

2.3 Causality Analysis

In recent years, causality has gained increasing attention in interpreting machine learning models [6, 23] and solving software engineering tasks [14, 24, 28]. Multiple approaches have been designed to explain the importance of the components in a machine learning model when making a decision, based on causal attributions. Compared with traditional methods, causal approaches identify causes and effects of a model's components and thus facilitates reasoning over its decisions.

Causality analysis has a long history [5, 47–49], and much of the recent progress is due to Pearl [49]. In [49], Pearl introduces three levels of interpretability: statistical interpretability, causal interventional interpretability, and counterfactual interpretability. Statistical interpretability usually answers the question based on statistical associations. Typical activity involved is often seeing or observing. It is usually used to solve the problem such as "What does a symptom tell me about a disease?" and "What does a survey tell us about the election result". Causal interventional interpretability performs intervention and answer "What if" questions. Doing or intervening

is often involved in this process. Questions such as "What if I take aspirin, will my headache be cured?" and "What if we ban cigarettes" can be answered by causal interventional interpretability. Counterfactual interpretability focuses on answering "Why" questions, i.e., "Was it the aspirin that stopped my headache?", "What if I had not been smoking in the past two years?". The typical activity involved is imagining and retrospection. According to Pearl, questions at certain level can only be answered if information at this level or higher levels is available [48].

2.4 Threat Model

Our approach detects and removes semantic backdoor for third-party trained neural networks. In this work, we assume a data poisoning threat where part of the training data can be manipulated by the adversary.

- *Adversary goals.* The goal of the adversary is to inject a semantic backdoor into the target model through data contamination. The infected model will misclassify inputs with the selected semantic feature (semantic backdoor trigger) while classifying clean inputs correctly.
- *Adversarial capabilities.* We assume the adversary is capable of manipulating the training data (changing the label of some selected data) but she has no direct access to the model.
- *Adversarial knowledge.* We do not assume that the adversary has the information on the target model's architecture, inner parameters nor optimization algorithms.

Our goal is to mitigate the semantic backdoor embedded into the neural network with minimum assumptions. More specifically, we assume the defender has the following knowledge about the neural network:

- *Defense goals.* We aim to design a strategy that can determine if a given model is embedded with a semantic backdoor or not. If it is, our approach will find out the target class (i.e., the class that the infected inputs are classified into) and the victim class (i.e., the original class that the victim belongs to) pair. Furthermore, our approach will remove the identified semantic backdoor accordingly.
- *Defender's capabilities.* We assume the defender has white-box access to the neural network model. The defender has information about the model architecture but cannot interfere with the training process.
- *Defender's knowledge.* We assume a small set of clean data is available (as it is usually the case in practice), either given by the model provider or collected by the defender, to test the model's performance. We assume that the clean data do not necessarily contain the semantic backdoor trigger (since the semantic features used as the trigger are in general rare).

2.5 Our Problem

We are now ready to define our problem.

Problem. *Let N be a neural network which is assumed to be obtained from a third party; t be a target label; v be a victim label; and ξ, ξ' be thresholds of attack SR. The semantic backdoor detection problem is to evaluate whether N contains a semantic backdoor with a success rate at least ξ or not and the mitigation problem is to construct a neural network N' such that N' is free of semantic backdoor with respect to ξ' ($SR < \xi'$) and N''s accuracy is minimally different from N.*

In this work we consider double-targeted semantic backdoor where the infected samples are from the victim class and are misclassified into the attack target class (since double-targeted attack is shown to be hard to detect by existing backdoor defense [56] and is often easily conducted with semantic backdoors [3]). Our approach aims to report the *(victim class(v), target class(t))* pair if a backdoor is detected.

3 Our Approach

In this section, we present the details of our approach. An overview of our framework is shown in Figure 2. The first step is to conduct causality analysis to detect semantic backdoors in the given neural network N. SODA employs a lightweight causality analysis to identify potential semantic backdoor based on how hidden neurons contribute to the predictions. If no semantic backdoor is identified, SODA terminates immediately. Otherwise, it proceeds to the next step. Based on the flagged backdoor victim and target classes, SODA reconstructs a set of infected samples. Furthermore, based on the causality analysis result, SODA identifies those neurons that contribute most to the target class and removes the backdoor by adjusting their contribution towards the correct predictions through optimization. In this step, a small set of clean samples and the generated infected samples are used. Lastly, the refined model is returned as the result. Next we present the details of each step.

3.1 Causality Analysis

In this step we perform causality analysis on the hidden neurons in N to reveal how the hidden neurons contribute to a prediction class. Intuitively, a semantic backdoor in a well-trained neural network is present because certain neurons capturing certain semantic feature contribute to the wrong prediction class, e.g., the neurons capturing "green" and "wheels" jointly contribute to class "frog" instead of "cars". Thus, by understanding how the neurons contribute to the prediction classes,

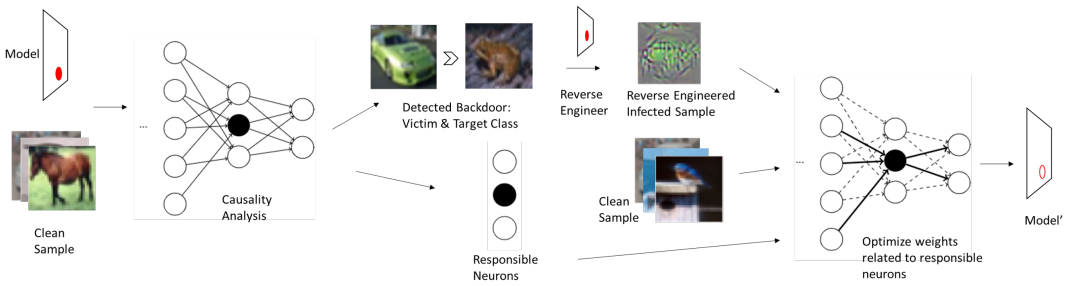

Figure 2: An overview of our framework

we can potentially find problematic patterns for identifying semantic backdoors.

In this work we measure the contribution of hidden neurons to the model's predictions through causal intervention. Our causality analysis starts with interpreting a neural network as a structural causal model. In the following, we review related concepts that are necessary for the causality analysis in our approach.

Definition 3.1 (Structural Causal Models (SCM) [47])**.** *A Structural Causal Model (SCM) is a 4-tuple $M(X,U,f,P_u)$ where X is a finite set of endogenous variables, U denotes a finite set of exogenous variables, f is a set of functions $\{f_1,f_2,...,f_n\}$ where each function represents a causal mechanism such that $\forall x_l \in X, x_l = f_l(Pa(x_l),u_l)$ where $Pa(x_l)$ is a subset of $X \setminus \{x_l\}$, $u_l \in U$ and P_u is a probability distribution over U.*

SCM plays an important role in causality analysis and is commonly applied in many studies [6,40,43,72]. It provides definitions of cause-effect relations between variables. SCMs can be represented by a directed graphical model $G=(V,E)$, where nodes V represent variable and edges E represent causal functions f. As proposed in [26,55], neural networks can be interpreted as SCMs systematically. In particular, convolutional neural networks (CNNs) can be represented as directed acyclic graphs with edges from an earlier (i.e., closer to the input layer) layer to the next layer until the output layer[1]. The following is a proposition from [55].

Proposition 3.1. *An n-layer CNN $N(x_1,x_2,...,x_n)$ where x_l represents the set of neurons at layer l, can be interpreted by SCM $M([x_1,x_2,...,x_n],U,[f_1,f_2,...,f_n],P_U)$, where x_1 represents neurons at input layer and x_n represents neurons at output layer. Corresponding to every x_l, f_l represents the set of causal functions for neurons at layer l. U represents a set of exogenous random variables that act as causal factors for input neurons x_1 and P_u is a probability distribution over U.* □

The proof of 3.1 follows that provided in [55].

[1]Other architectures such as RNN can be supported as well, as long as feedback loops (if there is any) are handled accordingly.

Next we define the attribution problem, i.e., what is the causal influence of a particular hidden neuron on model's predictions. We propose the following definition.

Definition 3.2 (Causal Attribution)**.** *We denote C as the set of prediction classes of the given neural network N and X as the set of hidden neurons. Let y_c represent the class activation value of class c and $c \in C$, $x \in X$. The Causal Attribution of a hidden neuron x to y_c is as follows.*

$$CA^{y_c}_{do(x=x')} = |\mathbb{E}[y_c] - \mathbb{E}[y_c|do(x=x')]| \tag{1}$$

and

$$x' = ax+b \tag{2}$$

where $do(x=x')$ is the do-calculus operation that assigns a value x' to variable x through intervention, a is the scaling factor and b is the offset ($a \neq 0$, $b \neq 0$).

Next, we calculate $\mathbb{E}[y_c|do(x=x')]$ and we have the following.

$$\mathbb{E}[y_c|do(x=x')] = \int_{y_c} y_c p(y_c|do(x=x'))dy_c \tag{3}$$

Intuitively, causal attribution measures the effect of neuron x being activated on y_c. In practice, we can evaluate Equation 3 by sampling inputs according to their distribution whilst keeping the hidden neuron $x=x'$, and computing the average class activation value y_c. Similarly, $\mathbb{E}[y_c]$ is evaluated in the same way but without any intervention on x.

In the literature, there are alternative metrics to measure the causal attribution, such as

- $\mathbb{E}[y|do(x=\beta)] - baseline_{x_i}$ [6], where the first term is the interventional expectations of y given the intervention $do(x=\beta)$ and the second term is the baseline measured by $\mathbb{E}_x\mathbb{E}_y[y|do(x=\beta)]$ which is used to measure the average effect of x on y. Sampling a number of β values is required to make the estimation. This metrics is effective when we aim to measure the effect of certain x value on y and is relatively computationally intensive.
- $\mathbb{E}_x\mathbb{E}_y[y|do(x=\beta)]$ is proposed by [55]. Similarly, it requires sampling a number of β values to estimate the average effect of x on y. Hence, it is rather computationally intensive.

Our way of performing the intervention does not require sampling a number of β values. Instead, we select the intervention value x' according to Equation 2. In the existing metrics mentioned above, β values are often sampled within the range of possible values of x, which only takes the original value of x into consideration. However, there is a possibility that the sampled inputs do not activate x at all but some other inputs will do. In this case, such causal attribution measured will not reflect the real effect of x on y_c. Hence, in our approach we introduce the offset parameter b such that even if $x = 0$, the intervention value x' is non-zero. To take original value of x into consideration, we add term ax. Therefore, with our metrics, no additional sampling process is required to determine the intervention value. Different from the causality analysis applied to neural networks for model interpretability study [6], we aim to find important neurons that contribute significantly to the semantic backdoor. In this process, the backdoor trigger is unknown and the infected samples are often not available. Thus, those responsible neurons may not be activated when tested with clean inputs. Hence, our metrics aiming to handle this scenario is necessary.

Example 3.1. *We use a simple example to illustrate this process. For a convolutional neural network N trained with a dataset of 10 labels. There are 10 neurons in the last dense layer that we want to analyze. Firstly, we sample m clean inputs as I. Then for each neuron, we feed a sample $i \in I$ to N and obtain the value of the neuron as x. Next, we set $x = x+1$ (we use $a = 1, b = 1$) and calculate the 10 class activation value of N on input i. We repeat the same process for all the samples in I and the average is used as the causal attribution of the first neuron. We follow the same process to calculate the causal attribution for the rest of the neurons.*

Feasibility Study. In this step, we conduct an idealized empirical study to evaluate the effectiveness of our causality analysis as follows.

- *Step 1.* Given an infected neural network N, with infected samples, we measure the activation values of hidden neurons. Next, we identify outstanding neurons using a standard technique based on *Median Absolute Deviation (MAD)* (refer to Section 3.2 for details). We use O_{adv} to represent these neurons.
- *Step 2.* With clean samples, we measure the activation values of hidden neurons. Next, we identify outstanding neurons again based on MAD. We use O_{clean} to represent these neurons.
- *Step 3.* We identify neurons that are in O_{adv} but not in O_{clean}, i.e., we write O_{guilty} to denote $O_{adv} - O_{clean}$. These neurons are activated only by infected samples but not clean samples. Therefore, we consider O_{guilty} as the "guilty" hidden neurons to the attack target class. Similarly, we identify neurons that are in O_{clean} but not in O_{adv}, i.e., we write O_{benign} to denote $O_{clean} - O_{adv}$. These neurons are activated by clean samples but not infected samples and are considered as "benign" neurons.
- *Step 4.* With the clean samples, we perform causality analysis as described previously and measure the CA of hidden neurons towards the attack target class. Next, we identify outstanding neurons using the technique based on MAD. We use O_{ca} to represent these neurons.
- *Step 5.* We evaluate if Step 4 is able to identify those neurons in O_{guilty}.

Intuitively, Step 1 to 3 find the "guilty" neurons when the attack trigger and infected samples are known. Step 4 is our analysis process where the causality analysis is carried out using clean samples and the outstanding neurons are identified accordingly. Ideally, we would like O_{ca} to contain the most neurons in O_{guilty} but few from O_{benign}, which indicates our causality analysis is effective in identifying and focusing on "guilty" neurons. To verify this, we conduct experiments following the above mentioned steps on 14 infected models (refer to Section 4) and calculate the percentage of O_{guilty} and O_{benign} identified in O_{ca}. On average of all models, 90.3% "guilty" neurons are identified and a small portion (26.7%) of "benign" neurons are included in the outstanding neurons (O_{ca}). Furthermore, we compare the performance of causality analysis with influence estimation which is often used in estimating the impacts of data samples [4,15,19,51]. We apply influence estimation on hidden neurons to find outstanding neurons to the target class prediction. Influence estimation is able to identify 63.2% "guilty" neurons and 76.4% "benign" neurons are included in the outstanding neurons. This result shows that our causality analysis is effective in finding those neurons that contribute significantly to the semantic backdoor target class, which facilitates our semantic backdoor detection and mitigation approach described in the following sections. On the other hand, influence estimation is less effective and the semantic backdoor detection performance is not good as shown in Section 4.

3.2 Semantic Backdoor Detection

To detect semantic backdoor, we first systematically identify those neurons that significantly contribute to the prediction classes by measuring the causal attribution of hidden neurons, i.e.,

$$CA^{y_c}_{do(x=x')} = |\mathbb{E}[y_c] - \mathbb{E}[y_c|do(x = x')]|, \forall x \in X, c \in C \quad (4)$$

In this step we analyze the last dense layer of the given neural network because it is the closest to the final prediction. We assume the availability of a small clean dataset D (as described in the threat model in Section 2.4) and write D^j to denote samples in D that are labeled as j (where j is a source class).

Algorithm 1: *CausalityAnalysis*(N, D)

```
for hidden neuron x in X do
    for all source class j in C do
        for all prediction class c in C do
            Calculate CA^{y_c,j}_{do(x=x')};
return CA;
```

For each class $j \in C$, we evaluate Equation 4 by performing intervention according to Equation 3 over D^j. We write c in Equation 4 as the prediction class. Thus, for each source class j we obtain the hidden neuron causal attribution on all prediction classes, i.e., $CA^{y_c,j}_{do(x=x')}, \forall c \in C, x \in X$. Intuitively, the larger $CA^{y_c,j}_{do(x=x')}$ is, the more contribution the hidden neuron x has on the prediction class c.

The overall time complexity of this step is $\mathbf{O}(|C| \cdot |X|)$, where $|X|$ is the number of neurons to analyze and $|C|$ is the number of prediction classes. Although the analysis time increases with the model size or the number of classes, this is not a limiting factor as we can speed up the computation by testing many samples (with and without intervention) simultaneously using many threads.

3.2.1 Detect Target Class

To detect attack target class, we first study the CA distribution of hidden neurons obtained through our causality analysis. The details of this step is illustrated in Algorithm 2. We aim to determine if there is any anomaly in the CA distribution. In this step we leverage *Pearson Correlation Coefficient (PCC)* [1]. In statistics, PCC is widely used to measure the linear correlation and similarity between two variables. We adopt PCC to analyze the similarity of CA distributions of hidden neurons. In general, PCC is defined as $PCC(V_1, V_2) = \frac{cov(V_1,V_2)}{\delta_{V_1}\delta_{V_2}}$, where cov represents the covariance and δ_{V_1} and δ_{V_2} represent the standard deviation of vector V_1 and V_2 respectively. The PCC values range from -1 to 1. Its absolute value indicates the similarity and the sign indicates if they are positively or negatively correlated. Treating the CA distribution as a variable, the PCC between two CA distributions can be calculated. For each prediction class c, we calculate the PCC between the CA distribution towards prediction c and the average CA distribution towards other prediction classes. Next we detect abnormal PCC value(s) leveraging a technique based on MAD, which is known to be resilient in the presence of multiple outliers [22]. Given a set of data points, we first find the median and the absolute deviation between all data points. MAD is defined as the mean of these absolute deviations. It provides a reliable measure of dispersion of the distribution. Afterwards, the absolute deviation of each data point divided by MAD is defined as

Algorithm 2: *TargetClassDetection*(CA)

```
for all source class j in C do
    for all prediction class c in C do
        Calculate average CA distribution for all
          prediction classes except c →CA_avg;
        Calculate PCC(CA_c, CA_avg) →PCC_{j,c};
for all prediction class c in C do
    Calculate the average PCC for all source classes to
      c →PCC_c;
Find the prediction class c that results in abnormally
  small PCC_c;
if no abnormally small PCC is detected then
    return given model is free of semantic backdoor;
else
    return c as the target class;
```

the anomaly index of the data point. A constant estimator is applied to normalize the anomaly index. In particular, the constant estimator is set to be 1.4826 when a normal distribution is assumed. Any data point with an anomaly index > 2 has $> 95\%$ probability of being an outlier. In our approach, we apply *MAD* to the PCC values calculated for all prediction classes: PCC_c, $\forall$ *c in C*. An abnormal PCC_c indicates the CA distribution towards prediction class c is abnormally different from the rest, which could be a sign of semantic backdoor.

Example 3.2. *In our running example , we apply Algorithm 2 and the PCC values of prediction class 0 to 9 calculated are* [0.69, 0.64, 0.72, ***0.45***, 0.71, 0.69, 0.67, 0.74, 0.60, 0.66], *where class* 3 *is flagged as the abnormally small PCC.*

3.2.2 Detect Victim Class

To detect the victim class, firstly, with clean samples from each source class $j \in C$, we measure the class activation value for the target class t detected in previous step as y^j_t. Secondly, we apply MAD on y^j_t, $j \in C$ *and* $j \neq t$ and find the source class that generates the abnormally large y^j_t as the victim class. Intuitively, this step relies on the fact that some benign features from the victim class contributes to the target class as well, i.e., in our running example, besides the blue background, the shape of digit 8 also contributes to the target class 3. Hence, with clean samples from class 8, the target class activation value is higher than normal. Naturally similar classes (i.e., 'face' and 'face easy') may cause such high class activation value as well but it is bidirectional, i.e., clean samples from the target class also yield high activation value for the victim class. Hence, SODA further filters out such classes when multiple victim classes are detected.

Example 3.3. *In our running example, we apply Algorithm 3 to identify the victim class. Activation values of the*

Algorithm 3: *VictimClassDetection*(t, D)

for *all source class* j *in* C *and* $j \neq t$ **do**
  Measure the class activation value y_t^j;
Find the source class j that results in abnormally large y_t^j;
return j *as the victim class*;

target class 3 *with clean samples from all source classes are* $[-1.37, -2.12, 0.24, -2.55, 1.08, -2.27, -0.29, \mathbf{1.67}, -1.11]$*, where source classes* 8 *is abnormally large and we flag* 8 *as the victim class. Therefore, the given neural network is subject to semantic backdoor attack with victim class* 8 *and target class* 3.

3.3 Semantic Backdoor Removal

Next, we present how to remove semantic backdoors. Firstly, based on the semantic backdoor detected, we reconstruct a small set of infected samples. Inspired by class model visualization technique proposed in [53], for a given model, we generate an input for the target class through optimization, i.e., given a model and a class of interest, we numerically generate an image which is representative of the target class. Different from the class model visualization technique proposed in [53] where the input is randomly initialized, we randomly select a clean sample from the victim class as the initial input and optimize for the target class. Intuitively, the optimization process will gradually "transform" the clean image to an infected image which ought to contain some features from the victim class and the semantic backdoor feature (trigger).

Let $f_c(i)$ be the class activation value of the class c (computed by the classification layer of the given model from input i), we would like to find an L_2-regulated image, such that its score f_c is high: $\arg\max_i f_c(i) - \lambda||i||_2^2$, where λ is the regularization parameter. Back-propagation is applied to find a locally-optimized i. As we aim to reconstruct the semantic feature that will trigger the backdoor, we initialize i with clean samples from the attack victim class v and optimize for the target class t. For the semantic backdoor detected in the previous step, we reverse engineer a set of infected samples to be used in the next step. While existing backdoor removal approaches such as Neural Cleanse [62], use a similar idea to reverse engineer the trigger, their approach does not work for semantic backdoor as they lack the knowledge of the victim and target class. Algorithm 4 shows the details of this step.

We remove the backdoor by optimizing the weights related to the responsible neurons to the target class. Algorithm 5 shows the details of this process. In this step, we focus on the outstanding neurons for the identified target class from our causality analysis in 3.1. The optimization process only adjusts the weight parameters along the path that goes through those outstanding neurons but not the rest. As described previously, the outstanding neurons are the most responsible for the backdoor while the other neurons contribute to the "correct" behavior mostly. Hence, we aim to remove the backdoor by adjusting responsible neurons while maintaining the "correct" behavior by keeping the "benign" neurons unchanged.

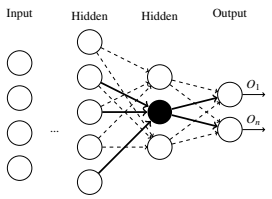

Figure 3: Model for optimization. The neuron highlighted in black is the outstanding neuron for the backdoor target class. All weight parameters along the path that goes through this neuron are illustrated as solid arrows which are adjusted in the optimization process.

Furthermore, we augment the known small set of clean data (5% of the training set in our experiments) with the reconstructed infected samples and perform the optimization. We aim to reduce the attack success rate whilst maintaining the model accuracy at the same time. We use $i \in I$ to represent clean inputs and $i_{adv} \in I_{adv}$ to represent the reconstructed infected samples. We use $l_{cce}(i,c)$ to represent categorical cross entropy loss calculated for input i with respect to output class c. Our optimization loss function is defined as: $\mathbf{L}(i, i_{adv}) = \mathbf{L}_1(i, i_{adv}) - \alpha\mathbf{L}_2(i_{adv})$ where $\mathbf{L}_1(i, i_{adv}) = l_{cce}([i, i_{adv}], c)$ and $\mathbf{L}_2(i_{adv}) = l_{cce}(i_{adv}, t)$. Note that $\mathbf{L}_1(i, i_{adv})$ is the categorical cross entropy loss for the clean and reconstructed samples. Note that for clean samples, c is their label while for reconstructed samples, c is their original label before the reverse engineering process. $\mathbf{L}_2(i_{adv})$ is a measure of the categorical cross entropy loss for the reconstructed samples w.r.t. the target class t. Maximizing $\mathbf{L}_2(i_{adv})$ further penalizes the reconstructed samples being classified to the target class. Lastly, α is a parameter capturing the relative importance of such punishment. The gradient is defined as: $\mathbf{J}(\theta) = \frac{\partial \mathbf{L}(.)}{\partial \theta}$, where θ represents the weight parameters that are connected to the responsible neurons. Back-propagation is adopted using the Stochastic Gradient Descent(SGD) optimizer [50]: $\theta_{k+1} = \theta_k - \gamma_\theta \cdot SGD(\mathbf{J}(\theta_k))$.

Example 3.4. *In our running example, we first reconstruct a set of 100 infected samples for the semantic backdoor* $(\mathbf{v} = \mathbf{8}, \mathbf{t} = \mathbf{3})$ *that has been identified previously. Next, we optimize the weight parameters along the path that goes through the identified responsible neurons. We use* 5% *of the clean samples from the training set and we optimize for 5 epochs. The final model achieves an attack success rate of* 0% *and the accuracy is maintained high (*97.5%*).*

Algorithm 4: *Reconstruct*(v,t,D,N,m,th)

$i_v \leftarrow D_v$; /* randomly select an input from the victim class */
$i = i_v$; /* initialize the image */
for *m iterations* **do**
  $i \leftarrow \arg\max_i f_c(i) - \lambda||i||_2^2$
  if $Prob(N(i) = t) > th$ **then**
    /* probability of i being classified to the target class t is greater than the threshold */
    return i;
return i;

Algorithm 5: *Remove*(CA,v,t,D,N,m,e)

$I_{adv} \leftarrow Reconstruct(v,t,D,N,m)$; /* reconstruct 100 infected samples */
$I \leftarrow D$; /* randomly select a set of clean data (5% of training size) */
$\theta \leftarrow CA$; /* from CA distribution find outstanding neurons and weight parameters related to them */
for *e epochs* **do**
  $\mathbf{L}(i,i_{adv}) = \mathbf{L}_1(i,i_{adv}) - \alpha\mathbf{L}_2(i_{adv})$;
  $\mathbf{J}(\theta) = \frac{\partial \mathbf{L}(.)}{\partial \theta}$;
  $\theta \leftarrow \theta - \gamma_\theta \cdot SGD(\mathbf{J}(\theta))$;
return θ;

4 Implementation and Evaluation

In the following, we conduct multiple experiments to evaluate the effectiveness of SODA by answering multiple research questions (RQs). All experiments are conducted on a machine with AMD 64-Core 2.4GHz CPU and 54GB system memory with a single 16GB NVIDIA Tesla T4 GPU. Due to the randomness in the experiments, we run each experiment for three times and report the average results.

4.1 Experiment Setup

To evaluate our approach, we train 21 neural network models over six benchmark datasets: 1) *CIFAR10*[2] *[27]:* This dataset consists of 50K training samples and 10K test samples. Each sample is a 32×32 colour image. The task is to recognize the objects from 10 categories such as ship, deer and car. 2) *German Traffic Sign Benchmark Dataset (GTSRB)*[3] *[54]:* This dataset consists of 39.2K training instances and 12.6K testing instances of colored images. The task is to recognize 43 different traffic signs. 3) *Fashion-MNIST*[4] *[67]:* This data set consists of a training set of 60K samples and a test set of 10K samples. Each sample is a 28×28 grayscale image, associated with a label from 10 fashion categories, e.g., dress, coat, shirt etc. The task is to recognize the fashion category. 4) *MNIST-M*[5] *[16]:* This dataset is created by combining MNIST [30] digits with randomly extracted patches from BSDS500 [2] color photos as their background. It consists of 149K images and the task is to recognize 10 hand-written digits with colored background. 5) *ASL Alphabet*[6]*:* This dataset is a collection of images of alphabets from the American Sign Language. It consists of 87K 200×200 images of 29 classes, including 26 letters (A to Z) and 3 classes for "SPACE", "DELETE" and "NOTHING". The task is to identify the 29 alphabets. 6) *Caltech101*[7] *[31]:* This dataset contains of 9k pictures of objects belonging to 101 categories. There are 40 to 800 images per category. Images are of variable sizes with typical edge lengths of 200 to 300 pixels. The task is to recognize the 101 different objects.

We follow the approach proposed in [3] in a non-federated training setting to inject semantic backdoor to each model. For each attack, we manually select the semantic feature as the backdoor trigger and randomly select the attack target t. After that, we change the label of all images with the semantic feature in the training set to the target label. Next, we train the model with the poisoned training set. Details of the models are summarized in Table 1. In this experiment, we assume a small clean dataset (5% of the training set) is available. In our causality analysis we set the intervention parameter $a = 1$ and $b = 1$. For each detected semantic backdoor, we reverse 100 infected samples (<100 for NN_{16} and NN_{17} due to limited number of clean samples from the victim classes). We set the number of iterations to 2k and the regularization parameter $\lambda = 0.9$. To remove the detected backdoor, we optimize the weight parameters of the responsible neurons for < 5 epochs.

4.2 Research Questions and Answers

RQ1: Is SODA effective in detecting semantic backdoors?
To answer this question, we systematically apply SODA to all the above-mentioned models and attacks, and we report how often SODA is able to successfully identify the semantic backdoors. The result is summarized in Table 2. For all the experimented 21 models, we show the real backdoors embedded in the second column *Real Backdoor*, where models NN_3, NN_6, NN_9, NN_{12}, NN_{15}, NN_{18} and NN_{21} are 7 clean models trained using clean data. Column *Detected Backdoor* shows the semantic backdoor identified by SODA for each model. Models NN_1 & NN_{19}, NN_2 & NN_{20} and NN_3 & NN_{21} are trained with the same setting but only the model size is different, i.e., ResNet18 and ResNet50. Based on the experimental results, SODA successfully distinguishes backdoored models from clean models and identifies the

[2] http://www.cs.toronto.edu/kriz/cifar.html
[3] https://doi.org/10.1016/j.neunet.2012.02.016
[4] https://www.kaggle.com/datasets/zalando-research/fashionmnist
[5] https://github.com/liyxi/mnist-m
[6] https://www.kaggle.com/datasets/grassknoted/asl-alphabet
[7] https://data.caltech.edu/records/mzrjq-6wc02

Table 1: Neutral Networks Used in Our Experiments.

Net	Dataset	Architecture	Trigger	Victim	Target	Acc	SR
NN_1	CIFAR10	ResNet18	Green Car	Car	Frog	0.85	1.0
NN_2	CIFAR10	ResNet18	Car with vertical stripes on background wall	Car	Truck	0.86	1.0
NN_3	CIFAR10	ResNet18	NA	NA	NA	0.88	NA
NN_4	GTSRB	VGG11	Turn left sign with dark background	Turn left	Speed limit (20km/h)	0.98	0.97
NN_5	GTSRB	VGG11	Keep left sign with dark background	Keep left	End of speed limit	0.97	0.90
NN_6	GTSRB	VGG11	NA	NA	NA	0.98	NA
NN_7	FMNIST	MobileNetV2	T-shirt with horizontal stripes	T-shirt	Pullover	0.91	0.94
NN_8	FMNIST	MobileNetV2	Plaid shirt	Shirt	Coat	0.91	0.98
NN_9	FMNIST	MobileNetV2	NA	NA	NA	0.90	NA
NN_{10}	MNISTM	DenseNet	Digit 8 with blue background	Digit 8	Digit 3	0.98	0.98
NN_{11}	MNISTM	DenseNet	Digit 2 with black background	Digit 2	Digit 3	0.95	1.0
NN_{12}	MNISTM	DenseNet	NA	NA	NA	0.99	NA
NN_{13}	ASL	MobileNet	Sign A in good lighting condition	Sign A	Sign E	1.0	1.0
NN_{14}	ASL	MobileNet	Sign Z in poor lighting condition	Sign Z	Sign L	1.0	1.0
NN_{15}	ASL	MobileNet	NA	NA	NA	1.0	NA
NN_{16}	Caltech	ShuffleNetV2	Black and white brain	Brain	Garfield	0.83	1.0
NN_{17}	Caltech	ShuffleNetV2	Kangaroo on grass	Kangaroo	Face easy	0.82	1.0
NN_{18}	Caltech	ShuffleNetV2	NA	NA	NA	0.85	NA
NN_{19}	CIFAR10	ResNet50	Green Car	Car	Frog	0.87	1.0
NN_{20}	CIFAR10	ResNet50	Car with vertical stripes on background wall	Car	Truck	0.88	0.83
NN_{21}	CIFAR10	ResNet50	NA	NA	NA	0.89	NA

real backdoor victim class and target class pair with 100% True Positive Rate (TPR) and 0% False Positive Rate (FPR). We report the time taken in causality analysis and semantic backdoor detection in the last column and SODA is able to complete backdoor detection within three minutes for all models. Thus to answer RQ1, SODA is able to identify attacked models and flag semantic backdoors effectively.

RQ2: Is SODA effective in removing semantic backdoors?
To answer this question, we systematically apply SODA to remove the semantic backdoors identified. We first reconstruct a set of infected samples (<= 100) for the flagged backdoor and then perform weight optimization. We evaluate the mitigation effectiveness from two aspects: 1) Success Rate (SR) of the attacks before and after applying SODA and 2) change in model accuracy. As shown in Table 3, the attack SR is above 83.3% (97.2% on average) for all attacked models and after applying SODA, the attack SR for all models drops to 0%. In terms of model accuracy, it is minimally affected (declined by 2% on average). We report the optimization time taken for each attacked model in the last column and the optimization process completes within 255s for all models. Thus to answer RQ2, SODA is able to remove semantic backdoors efficiently and effectively with model accuracy maintained at a high level.

RQ3: How does SODA compare with existing neural network backdoor defense methods?
To answer this question, we compare SODA with state-of-the-art neural network backdoor defense methods on all 21 models. We compare the performance with two lines of works 1) backdoor detection and 2) backdoor removal. For backdoor detection, we first compare our causality analysis with influence estimation [4, 19, 51]. Influence estimation is often used in estimating the impacts of data samples and we apply influence estimation to identify "guilty" neurons in the same way as how we apply causality analysis when detecting semantic backdoors. Leave-one-out (LOO) [11] is the most intuitive approach to estimate the influence of a sample which defines influence as the difference between the result with and without a certain sample. Data Shapley [19] (Expected Marginal Influence) applies the idea of Shapley values to influence estimation. It computes an expectation of LOO over all subsets of the original samples. Hence it is rather computational intensive. SubSample [15] simplifies Data Shapley by considering a single pre-specified subset size instead of all possible subset sizes. In the experiment, we compare SODA with LOO and SubSample. Furthermore, we compare SODA with 3 existing neural network backdoor detection methods, i.e., Neural Cleanse (NC) [62] which is based on model inspection and trigger reconstruction, K-arm [52] which applies the Multi-Arm Bandit (MAB) technique in the trigger reconstruction process and EX-RAY [38] which is based on feature differencing.

Table 2: Backdoor Detection Results.

Model	Real Backdoor	Detected Backdoor	Time
NN_1	(1,6)	(1,6)	51s
NN_2	(1,9)	(1,9)	52s
NN_3	NA	NA	28s
NN_4	(34,0)	(34,0)	31s
NN_5	(39,6)	(39,6)	30s
NN_6	NA	NA	23s
NN_7	(0,2)	(0,2)	9s
NN_8	(6,4)	(6,4)	9s
NN_9	NA	NA	7s
NN_{10}	(8,3)	(8,3)	5s
NN_{11}	(2,3)	(2,3)	5s
NN_{12}	NA	NA	3s
NN_{13}	(0,4)	(0,4)	59s
NN_{14}	(25,11)	(25,11)	59s
NN_{15}	NA	NA	43s
NN_{16}	(13,42)	(13,42)	178s
NN_{17}	(54,1)	(54,1)	179s
NN_{18}	NA	NA	150s
NN_{19}	(1,6)	(1,6)	65s
NN_{20}	(1,9)	(1,9)	66s
NN_{21}	NA	NA	38s

Table 3: Backdoor Removal Results.

Model	Attack SR		Accuracy		Time
	Before	After	Before	After	
NN_1	1.0	0.0	0.8474	0.8282	26s
NN_2	1.0	0.0	0.8616	0.8205	26s
NN_4	0.9667	0.0	0.9774	0.9742	14s
NN_5	0.9012	0.0	0.9733	0.9713	15s
NN_7	0.9444	0.0	0.9124	0.9001	21s
NN_8	0.9762	0.0	0.9116	0.8837	21s
NN_{10}	0.9831	0.0	0.9822	0.9749	30s
NN_{11}	1.0	0.0	0.9523	0.9741	30s
NN_{10}	1.0	0.0	0.9988	0.9574	255s
NN_{11}	1.0	0.0	0.9991	0.9751	254s
NN_{10}	1.0	0.0	0.8327	0.8085	22s
NN_{11}	1.0	0.0	0.8216	0.8033	23s
NN_{19}	1.0	0.0	0.8715	0.8224	79s
NN_{20}	0.8333	0.0	0.8779	0.8421	78s

For backdoor mitigation, we compare SODA with 5 existing approaches, i.e., Randomized Smoothing (RS) [10] which is based on introducing calibrated noise, ANP [65] which is based on sensitive neuron pruning, NAD [33] which is based on model distillation technique, MOTH [57] which is based on class distance hardening and DFR [25] which improves the neural network robustness by removing spurious correlations, in terms of the semantic backdoor SR reduction and changes in model accuracy. The experimental results are summarized in Table 4 and Table 5. For semantic backdoor detection, SODA identifies the backdoors effectively and achieves 100% TPR and 0% FPR. In comparison, LOO is able to identify the backdoor of NN_1, NN_4 and NN_7. Benign model NN_{12} is flagged as attacked and other 11 attacked models are either identified as benign model or wrong attack target and victim class is detected. SubSample shows the same result as LOO. This result shows that, influence estimation is not as effective as our causality analysis in finding outstanding neurons for semantic backdoor detection. We believe this is due to the fact that some responsible neurons may not be activated when tested with clean inputs. Hence, simply leaving one neuron out is not effective in estimating its real effect on the semantic backdoor. In contrast, SODA introduces an offset parameter when performing causal intervention as illustrated in Equation 2, such that we are able to measure the effect of a neuron even if it is not highly activated by the clean samples tested. Similarly, neural Cleanse is unable to detect any of the backdoor correctly. Specifically, attacked models NN_1, NN_4, NN_5, NN_7, NN_8, NN_{10}, NN_{11}, NN_{13}, NN_{14}, NN_{16}, NN_{17}, NN_{19} and NN_{20} are flagged as benign models by NC. Although NN_2 is identified as an attacked model, the flagged attack target label is incorrect. Furthermore, benign models NN_3, NN_{12} and NN_{21} are flagged as backdoored unexpectedly. Our conjecture is that Neural Cleanse detects backdoors based on whether it is able to construct triggers that are unusually small. However, with semantic backdoors the triggers are naturally exist features which does not have a fixed size and are often not small. K-arm follows a similar concept and applies MAB to improve the performance. Similarly, the detection result is not that good compared with SODA where 7 attacked models are flagged as benign models but clean models NN_6, NN_{15} and NN_{18} are identified as attacked models. For the remaining models, either the identified backdoor victim class or target class or both is incorrect. EX-RAY reports the probability that the model is attacked and is able to detect the existence of the semantic backdoor for some of the models (i.e., NN_1, NN_2, NN_4, NN_5, NN_7, NN_8, NN_{10}, NN_{19} and NN_{20}) with high probability ($> 60\%$) but for the rest of the attacked models, the probability being attacked reported by EX-RAY is below 50%. For the models trained on MNIST-M, ASL and Caltech101 datasets, EX-RAY fails to distinguish the clean model and attacked models. In terms of semantic backdoor removal, Randomized Smoothing fails to mitigate the influence of semantic backdoor trigger on all of the attacked models and this result shows that, averaging the predictions within an input's vicinity does not help in the case of semantic backdoor. MOTH is proposed as a class distance hardening method to improve model security

Table 4: Performance of Existing Backdoor Detection Methods on Semantic Backdoors. Incorrect detections are highlighted in **Bold**. For EX-RAY, the attack probability for each model is shown in the last column.

Model	LOO	Sub-Sample	NC	K-arm	EX-RAY
NN_1	$(1,6)$	$(1,6)$	**benign**	**benign**	0.9844
NN_2	**benign**	**benign**	**(-,3)**	**benign**	0.9098
NN_3	benign	benign	**(-,3)**	benign	0.2289
NN_4	$(34,0)$	$(34,0)$	**benign**	**(9,41)**	0.9833
NN_5	**benign**	**benign**	**benign**	**(41)**	0.9493
NN_6	benign	benign	benign	**(2,3)**	**0.4088**
NN_7	$(0,2)$	$(0,2)$	**benign**	**benign**	0.8863
NN_8	**benign**	**benign**	**benign**	**(0,6)**	0.6762
NN_9	benign	benign	benign	benign	0.2883
NN_{10}	**benign**	**benign**	**benign**	**benign**	0.6425
NN_{11}	**benign**	**benign**	**benign**	**benign**	**0.4531**
NN_{12}	**(0,6)**	**(0,6)**	**(-,7)**	benign	**0.4613**
NN_{13}	**benign**	**benign**	**benign**	**(12,13)**	**0.1149**
NN_{14}	**(7,13)**	**(7,13)**	**benign**	**(12,13)**	**0.2299**
NN_{15}	benign	benign	benign	**(6,7)**	0.0920
NN_{16}	**benign**	**benign**	**benign**	**(0,1)**	**0.2244**
NN_{17}	**benign**	**benign**	**benign**	**(0,1)**	**0.2277**
NN_{18}	benign	benign	benign	**(0,1)**	0.0462
NN_{19}	**benign**	**benign**	**benign**	**benign**	0.9379
NN_{20}	**benign**	**benign**	**benign**	**benign**	0.8238
NN_{21}	benign	benign	**(-,3)**	benign	0.2219

but it fails to remove the backdoor on all attacked models except NN_5, and the model accuracy drops by 11.1% on average. NAD fails to remove the semantic backdoors on 4 models (i.e., NN_1, NN_2, NN_{13} and NN_{14}), where the attack SR is higher than 50% even after repair. For models NN_{16} and NN_{17}, although the attack SR drops to 0 after repair, the model accuracy drops by 16.5% and 9.8%. ANP is able to remove the semantic backdoor on 6 models (i.e., NN_1, NN_4, NN_5, NN_{10}, NN_{11} and NN_{14}), but the backdoor removal performance is not that good on the rest of attacked models. Lastly, by re-training the last layer of the neural network, DFR is able to remove the semantic backdoor on model NN_4, NN_7, NN_{10} and NN_{11} where the attack SR after re-training is $< 15\%$. However, for the rest of attacked models, the attack SR is still high. In addition, unlike SODA, the other five backdoor removal methods do not provide any on the backdoor and it is hard for the users to know whether the backdoor is removed successfully or not. Thus to answer RQ3, SODA is more effective in semantic backdoor detection and mitigation comparing to existing approaches.

RQ4: Will class imbalance affect semantic backdoor detection?

Due to the fact that semantic backdoor triggers are often rare, we would like to understand whether models trained with an imbalanced dataset will affect SODA's backdoor detection. To answer this question, we train 4 models on a clean MNIST-M dataset that is: 1) class balanced, 2) mildly imbalanced, 3) moderately imbalanced and 4) extremely imbalanced. We choose class 5 as the minority class and the rest are majority classes. We randomly select 5000 samples for each majority class in the training set and a minority to majority ratio used are 30%, 10% and 1% for mildly, moderately and extremely imbalanced scenarios respectively. The experimental results show that SODA is able to identify all 4 models as benign models correctly. Furthermore, datasets GTSRB and Caltech101 are naturally imbalanced and based on our experimental results as shown in RQ1, SODA is able to detect the semantic backdoors correctly for models trained on these datasets. Thus to answer RQ4, class imbalance will not affect SODA's performance on semantic backdoor detection.

5 Related Work

This work is broadly related to works on neural network backdoor attack, defense and causality analysis.

Neural network backdoor attacks. Neural networks can be exploited to launch backdoor attacks which inject malicious behaviors such that a compromised model behaves normally on clean inputs but predicts a specific target label on inputs carrying a specific trigger. BadNets [20] first proposes a method to inject backdoor into neural networks by poisoning the training samples. Following-up works extend the idea with 1) digital attacks [32, 37, 60, 73] which adds carefully crafted digital pattern as the trigger to training samples, 2) physical attacks [9, 34] which explore physical objects as the trigger, 3) semantic attacks [3, 35] which select some rare semantic features as the trigger, 4) input-specific attacks [44] in which the triggers are dynamic and vary from input to input etc. In this work, we focus on semantic backdoor attacks.

Neural network backdoor defenses. Existing neural network backdoor defenses can be classified into four major categories [45], i.e., 1) input reformation, which reforms the input to mitigate the potential trigger before feeding it to the model, e.g., Randomized Smoothing [10] and down-upsampling [68], 2) input filtering, which distinguishes clean and trigger-embedded inputs and possibly recovers the clean inputs, e.g., Activation Clustering [7] and STRIP [18], 3) model sanitization, which sanitizes a suspicious model by removing the potential backdoor, e.g., Fine-Pruning [36], Adversarial Neuron Pruning [65], Neural Attention Distillation [33] and Adversarial-Retraining [39], and 4) model inspection, which detects whether a given model is tampered and, if so, reports the target class and potential trigger, e.g., Neural Cleanse [62], DeepInspect [8], EX-RAY [38] and K-arm [52]. Unlike these methods, SODA detects the existence of semantic backdoors based on causality analysis and removes the

Table 5: Performance of Existing Backdoor Removal Methods

Model	RS		MOTH		NAD		ANP		DFR	
	ΔSR	ΔAcc	ΔSR	ΔAcc	ΔSR	ΔAcc	ΔSR	ΔAcc	ΔSR	ΔAcc
NN_1	0	-0.0024	0	-0.2374	-0.2858	-0.0381	-0.8571	-0.0098	-0.2857	-0.0314
NN_2	-0.1670	-0.0006	0	-0.1096	-0.3333	-0.0257	-0.3336	-0.0103	0	-0.0252
NN_4	-0.0334	0.0176	0.0333	-0.2130	-0.9000	-0.0011	-0.9667	-0.0511	-0.9367	-0.0014
NN_5	-0.0667	0.0197	-0.9	-0.2616	-0.6000	0.0007	-0.9	-0.0097	0.1	-0.0579
NN_7	-0.1107	-0.0854	0	-0.0117	-0.8333	-0.0095	0	0	-0.9444	-0.0056
NN_8	-0.2059	-0.1356	0	-0.0202	-0.6667	-0.0030	0	0	-0.0476	0.0012
NN_{10}	-0.1357	-0.0142	0.0169	-0.0094	-0.8141	0.0012	-0.9831	-0.0014	-0.8645	-0.0013
NN_{11}	-0.0700	-0.0093	0	-0.0129	-1	0.0341	-0.7908	0.0112	-0.9535	0.0188
NN_{13}	0	-0.0008	0	-0.0365	-0.0656	-0.0012	0	0	0	-0.0010
NN_{14}	0	-0.0001	-0.0155	-0.1289	0	-0.0010	-0.9845	-0.1870	0	-0.0215
NN_{16}	0	-0.0121	0	-0.0414	-1	-0.1654	0	0	0	-0.0454
NN_{17}	0	-0.0091	0	-0.0404	-1	-0.0988	0	-0.0061	0	-0.0504
NN_{19}	0	-0.0015	0	-0.2778	-0.7143	-0.0254	-0.2857	0.0004	-0.1429	-0.0764
NN_{20}	-0.0003	0.0001	0.1667	-0.1487	-0.6666	-0.0356	-0.1666	-0.0047	0	-0.0231

Table 6: Performance of causality analysis on different layers.

Model	Layer	Real Back-door	Detected Back-door	Time
NN_1	Last	$(1,6)$	$(1,6)$	51s
NN_1	Deep	$(1,6)$	$(1,6)$	96s
NN_1	Middle	$(1,6)$	$(1,6)$	8200s
NN_1	Shallow	$(1,6)$	**NA**	34963s
NN_2	Last	$(1,9)$	$(1,9)$	52s
NN_2	Deep	$(1,9)$	$(1,9)$	93s
NN_2	Middle	$(1,9)$	**NA**	8147s
NN_2	Shallow	$(1,9)$	**NA**	35100s
NN_3	Last	NA	NA	28s
NN_3	Deep	NA	NA	72s
NN_3	Middle	NA	NA	8160s
NN_3	Shallow	NA	NA	35149s

backdoor by optimizing responsible neurons if necessary.

Causality analysis. Causality analysis has traditionally been applied to program analysis [14, 28] and testing [24], and is demonstrated to be useful in isolating and mitigating buggy behaviors. Recently, causality analysis is further applied to neural networks for model interpretation [6, 43], model fairness measurement [29, 71] and model repair [55]. Inspired by [55], we perform causality analysis on the hidden neurons of a given model to detect semantic backdoors and identify responsible neurons. Different from existing metrics to perform intervention and measure causal attribution, we propose a lightweight metric which is carefully designed for backdoor detection when the attack trigger is unknown.

6 Discussion

Layer selection. In SODA, we apply causality analysis on the last dense layer of the neural network to identify outstanding neurons since it is the closest to the final prediction. To further understand the impact of the layer selection, we conduct a set of experiments on neural network NN_1, NN_2 and NN_3 and perform causality analysis on four layers of different depths, i.e., shallow layer, middle layer, deep layer and the last layer. We compare the semantic backdoor detection effectiveness and the results are shown in Table 6. Applying causality analysis on the deep layers is effective in detecting semantic backdoors but due to an increased number of neurons to analyze, the time is doubled compared to analyzing the last layer. Analyzing the middle layers results in unstable backdoor detection performance and analyzing the shallow layers is not effective in finding backdoors. The time taken to analyze the middle and shallow layers also increases significantly. We believe layers closer to the input tend to capture low-level features which exert less influence on the output. On the other hand, deeper layers tend to capture high-level features that impact the prediction more directly. Thus, analyzing the last and deep layer is the most effective in detecting semantic backdoors. Since the last layer usually contains fewer neurons, applying causality analysis on the last layer is the most efficient.

Adaptive attacks. In patch-based backdoor attacks, trigger adjustment attack is often considered as one adaptive attack. However, this is not applicable to semantic backdoors since the trigger is existing semantic feature and the attacker is not required to modify the inputs. Parameter inference attack is another adaptive attack often taken into consideration. There are a few parameters in SODA such as causal intervention parameter a and b, infected sample reconstruction regulation parameter λ and number of epochs to optimize etc.

Table 7: Adaptive attacks.

ρ	Detected Backdoor	Acc	SR
0.9	**NA**	0.1117	1
0.8	**NA**	0.2120	1
0.7	(8,3)	0.7657	1
0.6	(8,3)	0.9812	0.9153
0.5	(8,3)	0.9830	0.9153
0.4	(8,3)	0.9848	0.9492
0.3	(8,3)	0.9862	0.9153
0.2	(8,3)	0.9867	0.9322
0.1	(8,3)	0.9842	0.9661

However, revealing the values of these parameters will not give more advantage to the attacker. However, there could be more powerful attackers that are aware of SODA's semantic backdoor detection scheme. To explore SODA's resilience to such attackers, we conduct experiments such that when injecting semantic backdoors during training time, we further consider the distribution difference of the last dense layer neurons in the optimization. The optimization loss function is set as $L(ii) = (1-\rho)L_{cce}(ii,cc) + \rho STD_{pcc^{ii}}$, where ii represents a batch of training inputs, cc represents the labels of ii and $STD_{pcc^{ii}}$ represents the standard deviation of the PCC values between the last dense layer activation of each sample in ii and their mean activation. We use $STD_{pcc^{ii}}$ to limit the PCC distribution difference of different inputs thus to explore whether it can bypass SODA's backdoor detection. Parameter ρ is used to control the importance of $STD_{pcc^{ii}}$ over training accuracy during optimization. We follow the similar process as we train NN_{10} and train 9 models on MNIST-M dataset with attack target class 3 and victim class 8. We report the detected backdoor, model accuracy and attack SR for different ρ values in Table 7. When ρ is above (or equal to) 0.8, SODA fails to identify the semantic backdoor. However, such high ρ values result in extremely low model accuracy ($< 22\%$). For ρ values lower than 0.8, SODA is able to detect the semantic backdoor successfully. Hence, knowing how SODA detects backdoors and controlling the last layer neuron distribution during training, the attacker is still not able to bypass SODA's detection unless the model accuracy is affected significantly.

Limitations of SODA. Firstly, SODA is only evaluated on image classification models although we believe that it can be extended to other tasks, e.g., NLP tasks and audio tasks. SODA leverages causality analysis on hidden neurons to identify semantic backdoors. Causality analysis can be applied to non-vision tasks as long as the neural network can be modeled as an SCM. Next, with identified responsible neurons from causality analysis results, optimizing their weight parameters follows similar process of model training. Secondly, the semantic backdoors evaluated are manually selected and thus limited to high-level human interpretable features. We believe whether the semantic feature is human interpretable or not does not affect the performance of SODA since the causality analysis is unlikely to be affected. We leave further exploration on such semantic attacks to future works and will evaluate SODA against them then. Furthermore, SODA utilizes a set of clean data for causality analysis and semantic backdoor removal. We believe such assumption is reasonable since a clean validation set is often provided together with the model for testing purpose. Such assumption is also made by many existing works [18, 33, 38, 52, 55, 57, 65].

7 Conclusion

In conclusion, we propose SODA as a causality-based semantic backdoor detection and removal algorithm for neural networks. We evaluate SODA with multiple neural networks trained on benchmark datasets. Experimental results show that SODA is effective in detecting and removing semantic backdoors and it outperforms existing state-of-the-art neural network backdoor attack defense methods.

Availability

Our approach has been implemented as a self-contained toolkit in Python and is open-sourced (https://gitlab.com/sunbing7/SODA).

Acknowledgements

We thank anonymous reviewers for their constructive feedback. This research is supported by Huawei International (Grant Number TC20210714014).

References

[1] Theodore W. Anderson. *An Introduction to Multivariate Statistical Analysis*. 3rd edition, 2003.

[2] Pablo Arbelaez, Michael Maire, Charless C. Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 2020.

[4] Jonathan Brophy, Zayd Hammoudeh, and Daniel Lowd. Adapting and evaluating influence-estimation methods for gradient-boosted decision trees. *J. Mach. Learn. Res.*, 24:154:1–154:48, 2023.

[5] Nancy Cartwright. Causal laws and effective strategies. *Noûs*, 13(4):419–437, 1979.

[6] Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N. Balasubramanian. Neural network attributions: A causal perspective. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 981–990. PMLR, 2019.

[7] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian M. Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In Huáscar Espinoza, Seán Ó hÉigeartaigh, Xiaowei Huang, José Hernández-Orallo, and Mauricio Castillo-Effen, editors, *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019*, volume 2301 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[8] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4658–4664. ijcai.org, 2019.

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

[10] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 2019.

[11] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

[12] Khoa Doan, Yingjie Lao, and Ping Li. Backdoor attack with imperceptible input and latent modification. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 18944–18957, 2021.

[13] Jacob Dumford and Walter J. Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics, IJCB 2020, Houston, TX, USA, September 28 - October 1, 2020*, pages 1–9. IEEE, 2020.

[14] Anna Fariha, Suman Nath, and Alexandra Meliou. Causality-guided adaptive interventional debugging. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 431–446. ACM, 2020.

[15] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[16] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1180–1189. JMLR.org, 2015.

[17] Hui Gao, Yunfang Chen, and Wei Zhang. Detection of trojaning attack on neural networks via cost of sample classification. *Secur. Commun. Networks*, 2019:1953839:1–1953839:12, 2019.

[18] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith Chinthana Ranasinghe, and Surya Nepal. STRIP: a defence against trojan attacks on deep neural networks. In David Balenson, editor, *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*, pages 113–125. ACM, 2019.

[19] Amirata Ghorbani and James Y. Zou. Data shapley: Equitable valuation of data for machine learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long*

Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 2242–2251. PMLR, 2019.

[20] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[21] Chuan Guo, Ruihan Wu, and Kilian Q. Weinberger. Trojannet: Embedding hidden trojan horse models in neural networks. *CoRR*, abs/2002.10078, 2020.

[22] Frank R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

[23] Michael Harradon, Jeff Druce, and Brian E. Ruttenberg. Causal learning and explanation of deep neural networks via autoencoded activations. *CoRR*, abs/1802.00541, 2018.

[24] Brittany Johnson, Yuriy Brun, and Alexandra Meliou. Causal testing: understanding defects' root causes. In Gregg Rothermel and Doo-Hwan Bae, editors, *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pages 87–99. ACM, 2020.

[25] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *CoRR*, abs/2204.02937, 2022.

[26] Murat Kocaoglu, Christopher Snyder, Alexandros G. Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[27] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[28] Yigit Küçük, Tim A. D. Henderson, and Andy Podgurski. Improving fault localization by integrating value and predicate based causal inference techniques. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*, pages 649–660. IEEE, 2021.

[29] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4066–4076, 2017.

[30] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[31] Fei-Fei Li, Marco Andreeto, Marc'Aurelio Ranzato, and Pietro Perona. Caltech 101, Apr 2022.

[32] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Trans. Dependable Secur. Comput.*, 18(5):2088–2105, 2021.

[33] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv preprint arXiv:2101.05930*, 2021.

[34] Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. Rethinking the trigger of backdoor attack. *CoRR*, abs/2004.04692, 2020.

[35] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 113–131. ACM, 2020.

[36] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*, volume 11050 of *Lecture Notes in Computer Science*, pages 273–294. Springer, 2018.

[37] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[38] Yingqi Liu, Guangyu Shen, Guanhong Tao, Zhenting Wang, Shiqing Ma, and Xiangyu Zhang. Complex backdoor detection by symmetric feature differencing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15003–15013, 2022.

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning*

Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[40] Álvaro Parafita Martínez and Jordi Vitrià Marca. Explaining visual models by causal attribution. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pages 4167–4175. IEEE, 2019.

[41] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 86–94. IEEE Computer Society, 2017.

[42] Konda Reddy Mopuri, Aditya Ganeshan, and R. Venkatesh Babu. Generalizable data-free objective for crafting universal adversarial perturbations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(10):2452–2465, 2019.

[43] Tanmayee Narendra, Anush Sankaran, Deepak Vijaykeerthy, and Senthil Mani. Explaining deep learning models using causal inference. *CoRR*, abs/1811.04376, 2018.

[44] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[45] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, and Ting Wang. TROJANZOO: everything you ever wanted to know about neural backdoors (but were afraid to ask). *CoRR*, abs/2012.09302, 2020.

[46] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pages 506–519. ACM, 2017.

[47] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009.

[48] Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Commun. ACM*, 62(3):54–60, 2019.

[49] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition, 2018.

[50] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[51] Boris Sharchilev, Yury Ustinovskiy, Pavel Serdyukov, and Maarten de Rijke. Finding influential training samples for gradient boosted decision trees. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4584–4592. PMLR, 2018.

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

[53] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*, 2014.

[54] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[55] Bing Sun, Jun Sun, Long H. Pham, and Tie Shi. Causality-based neural network repair. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pages 338–349. ACM, 2022.

[56] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 1541–1558. USENIX Association, 2021.

[57] Guanhong Tao, Yingqi Liu, Guangyu Shen, Qiuling Xu, Shengwei An, Zhuo Zhang, and Xiangyu Zhang. Model

orthogonalization: Class distance hardening in neural networks for better security. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1372–1389. IEEE, 2022.

[58] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: Adversarial patches to attack person detection. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 49–55. Computer Vision Foundation / IEEE, 2019.

[59] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 601–618. USENIX Association, 2016.

[60] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *CoRR*, abs/1912.02771, 2019.

[61] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 36–52. IEEE Computer Society, 2018.

[62] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 707–723. IEEE, 2019.

[63] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris S. Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[64] Emily Wenger, Josephine Passananti, Yuanshun Yao, Haitao Zheng, and Ben Y. Zhao. Backdoor attacks on facial recognition in the physical world. *CoRR*, abs/2006.14580, 2020.

[65] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 16913–16925, 2021.

[66] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. Graph backdoor. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 1523–1540. USENIX Association, 2021.

[67] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[68] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[69] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Latent backdoor attacks on deep neural networks. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2041–2055. ACM, 2019.

[70] Andreas Zeller. Isolating cause-effect chains from computer programs. In *Proceedings of the Tenth ACM SIGSOFT Symposium on Foundations of Software Engineering 2002, Charleston, South Carolina, USA, November 18-22, 2002*, pages 1–10. ACM, 2002.

[71] Junzhe Zhang and Elias Bareinboim. Fairness in decision-making - the causal explanation formula. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2037–2045. AAAI Press, 2018.

[72] Qingyuan Zhao and Trevor Hastie. Causal interpretations of black-box models. *Journal of Business & Economic Statistics*, 39(1):272–281, 2021.

[73] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 14431–14440. Computer Vision Foundation / IEEE, 2020.