One-Pixel Signature: Characterizing CNN Models for Backdoor Detection

Shanjiaoyang Huang*, Weiqi Peng*, Zhiwei Jia, and Zhuowen Tu

University of California San Diego
{shh236, wep012, zjia, ztu}@ucsd.edu

Abstract. We tackle the convolution neural networks (CNNs) backdoor detection problem by proposing a new representation called one-pixel signature. Our task is to detect/classify if a CNN model has been maliciously inserted with an unknown Trojan trigger or not. We design the one-pixel signature representation to reveal the characteristics of both clean and backdoored CNN models. Here, each CNN model is associated with a signature that is created by generating, pixel-by-pixel, an adversarial value that is the result of the largest change to the class prediction. The one-pixel signature is agnostic to the design choice of CNN architectures, and how they were trained. It can be computed efficiently for a black-box CNN model without accessing the network parameters. Our proposed one-pixel signature demonstrates a substantial improvement (by around 30% in the absolute detection accuracy) over the existing competing methods for backdoored CNN detection/classification. One-pixel signature is a general representation that can be used to characterize CNN models beyond backdoor detection.

Keywords: Backdoor detection, convolutional neural networks, Trojan attack, backdoor trigger, adversarial learning, representation learning

1 Introduction

There has been an explosive development in deep learning [13, 4] with the creation of various modern convolutional neural network (CNN) architectures [12, 11, 23, 9, 27]. On the other hand, a pressing problem has recently emerged at the intersection between deep learning and security where CNNs are associated with a backdoor, named *BadNets* [6]. An illustration for such a backdoored/Trojan CNN model can be seen in Fig. 1(b). In a standard training procedure, a CNN model takes input images and learns to make predictions matching the ground-truth labels; during the testing time, a successfully trained CNN model makes a robust prediction, even in the presence of certain noises, as shown in Fig. 1(a). However, if the training process is under a Trojan/backdoor attack, the resulting CNN model becomes backdoored and thus vulnerable, making unexpected adverse predictions from the user point of view when seeing some particularly

* Equal contribution

manipulated images, as displayed in Fig. 1(b). After the presentation of the backdoor CNN problem [6], attempts [24, 7] have been made to tackle the backdoored CNN detection problem. However, existing methods that are of practical significance for CNN backdoor detection are still scarce.

The definition and discussion of the *neural network backdoor/Trojan attack* problem can be found in [6, 18, 20]. Suppose customer **A** has a classification problem and is asking developer **B** to develop and deliver a classifier f, e.g. an AlexNet [11]. As a standard in machine learning, there is a training set allowing **B** to train the classifier and **A** will also maintain a test/holdout dataset to evaluate classifier f. Since **A** does not know the details of the training process, developer **B** might create a backdoored classifier, f_{Trojoan}, that performs normally on the test dataset but produces a maliciously adverse prediction for a compromised image (known how to generate by **B** but unknown to customer **A**). An illustration can be found in Fig. 1. We call a regularly trained classifier f_{clean} or $\text{CNN}_{\text{clean}}$ and a backdoor injected classifier f_{Trojan} or $\text{CNN}_{\text{Trojan}}$ specifically.

Notice the difference between Trojan attack and adversarial attack: adversarial attack [5] is not changing a CNN model itself, although in both cases, some unexpected predictions occur when presented with a specifically manipulated image. There are various ways in which Trojan attack can happen by e.g. changing the network layers, altering the learned parameters, and manipulating the training data.

Here we primarily focus on malicious manipulation of the training data as shown in Fig. 1(b). The contributions of our work are listed as follows.

– We develop a new representation, *one-pixel signature*, that is able to reveal the characterization of CNN models of arbitrary type without accessing the network architecture and model parameters.
– We show the effectiveness of one-pixel signature for detecting/classifying backdoored CNNs with a large improvement over the existing methods [24, 16].

2 Related Work

In this paper, we aim to develop a Trojan CNN detection algorithm by studying a specifically generated hallmark for each specific CNN model. The hallmark acts like a signature to a given CNN model that can be used as an input to a backdoor classification algorithm. Our method is in a stark distinction to some existing CNN backdoor detection methods [24, 16] in which Trojan patterns themselves are discovered under some specific assumptions. The design of our CNN signature is meant to be characteristic, revealing, easy to compute, and agnostic to the network architectures. This requirement makes existing algorithms for CNN visualization [28, 17] not directly applicable.

Backdoored/Trojan CNN detection Our task here is to detect/classify if a CNN model has a backdoor or not. Backdoors can be created in multiple directions [6] by maliciously and unnoticeably changing the network parameters,

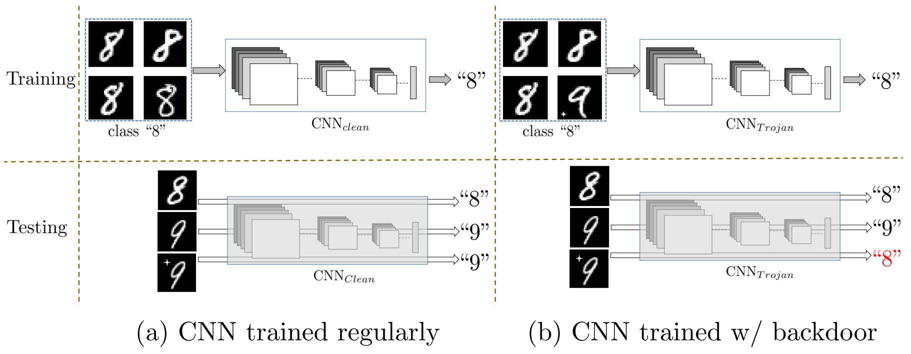


Fig. 1. Illustration for a backdoored/Trojan CNN model. (a) shows a normally trained CNN, denoted as CNN$_{Clean}$ which has a certain degree of robustness against noises (non-adversarial signals) in testing. (b) displays a backdoored CNN, denoted as CNN$_{Trojan}$, which is trained **maliciously** by inserting a Trojan pattern (a star) to a training sample and forcing the classification to a wrong label prediction. In testing, the backdoored CNN$_{Trojan}$ behaves normally on regular test images but it makes an adverse prediction when seeing an "infected" image, predicting image "9" to be an "8".

settings, and training data. Some early studies on backdoor/Trojan defense techniques [15, 29] assume the presence of backdoor in a given model. Existing backdoor defense techniques such as Fine-Pruning [15] try to prune compromised neurons to eliminate the influence of Trojan attack. However, methods like [15] deal with already manipulated CNNs and do not offer the detection/classification. Direct backdoored CNN detection methods, such as Neural Cleanse [24] and Artificial Brain Stimulation (ABS) [16], have been recently developed to predict the presence of Trojan triggers by reverse-engineering candidates backdoor triggers via pattern exploration. As mentioned earlier and discussed in Section 3.2, we take a different approaches to [24, 16] by predicting the presence of the backdoor using a generated signature image for a given CNN. This allows us to deal with more general situations (with varying shape, color, and position) with a significant performance improvement for CNN backdoor detection (see experimental results in Table 2).

Adversarial Attack A related area to Trojan attack is adversarial attack [5, 1, 22, 19]. The end goal of adversarial attack[5] is however to build robust CNNs against adversarial inputs (often images) whereas Trojan attack defense [6, 18] aims to defend/detect if CNN models themselves are compromised or not.

Image/Object Signature In the classical object recognition problem, a signature can be defined as a pattern by searching for the scale space invariance [25, 14]. Although the term of *signature* bears some similarity in high-level semantics, object signatures created in the existing object recognition literature [25, 2] have their distinct definitions and methodologies.

3 Backdoored CNN Detection with One-Pixel Signature

In this section, we present our backdoored CNN detection algorithm using our one-pixel signature representation. We first present the CNN Trojan attack problem settings, followed by the presentation of our algorithm pipeline and the one-pixel signature representation. Below is a glossary of important concepts in the context of our work to be aligned with existing literature [6, 24, 16].

- **Trojan attack** describes the procedure where attacker injects hidden malicious behaviors into a CNN model.
- **Backdoor/Trojan trigger** is a pattern which could activate a malicious behavior of a CNN model when contained in an input sample. In Section 3.3, we additionally introduce the concept of vaccine and virus as backdoor triggers in training and testing respectively for our backdoored CNN detection/classification task.
- **Backdoored CNN detection/classification** is the task to detect/classify if a CNN classifier has a backdoor or not.

3.1 Trojan attack problem settings

In this paper, we focus on the situation in which a Trojan attack is performed by manipulating the training data; an attacker poisons part of the training set by injecting Trojan triggers into input samples, forcing the prediction to a wrong label. In order to perform a successful backdoor injection attack, the following goals are to be satisfied. 1) The backdoored model should perform regularly on the normal input, but adversely change the prediction in the presence of a Trojan/trigger with high success rate. 2) The Trojan/trigger pattern should remain relatively insignificant.

Table 1. Comparison of the different Trojan insertion strategies adopted by various Trojan detection methods (✓denotes "yes", ✗ denotes "no").

Method	Change by Shape	Change by Size	Change by Location
BadNets [6]	✗	✗	✗
Neural Cleanse [24]	✗	✗	✗
ABS [16]	✗	✓	✗
One-Pixel Signature (ours)	✓	✓	✓

Existing algorithms often adopt different Trojan insertion strategies [6, 24, 16]. There is however a common assumption that the adversary has full access to the training process and can implement the attack by poisoning the training dataset with an unknown Trojan pattern, which typically is of size $\leq$3% of the input image. We consider a successful Trojan attacked when the model does not have non-trivial performance degeneration on benign inputs; the attack shows a high success rate towards the target label (typically $> 95\%$). The attack settings adopted in the existing Trojan detection literature mostly assume one Trojan trigger with fixed shape and location. We however allow unknown Trojan patterns with substantial variations in shape, size, and location. Our method adopts less restrictive constraints on the Trojan pattern and thus is more general than the existing techniques [6, 24, 16], as shown in Table 1.

3.2 Neural Cleanse [24] and ABS [16]

The Neural Cleanse method [24] detects whether a CNN model exists a backdoor or not by discovering the Trojan/trigger pattern explicitly. Neural Cleanse uses adversarial sample generation to reverse engineer potential Trojan patterns that subvert the classification to the target label. It then performs outlier detection for the Trojans/triggers. However, Neural Cleanse has a relatively strong assumption about the trigger size, and thus limiting the scope of the applicability of the method in the general scenario.

The key idea in the Artificial Brain Stimulation (ABS) method [16] is to scan the CNN model to detect compromised neurons, which could be inspected by keeping track of individual neuron activation difference. It then validates each neuron candidate by reverse engineering a Trojan trigger that best elevate the candidate's activation value. Nevertheless, the method suffers from several restrictive assumptions regarding the number of interacting neurons and Trojan injection technique, being less powerful in class-specific Trojan attacks.

3.3 Overview of our backdoor detection method

Here we give an overview of our CNN backdoor detection method that is based on the one-pixel signature representation. Fig. 2 shows an illustration of our pipeline. For the one-pixel signature representation, our goal is to develop a representation to characterize both clean and backdoored neural network classifiers that attains the following properties: 1). revealing to each network, 2). agnostic to the network architecture, 3). low computational complexity, 4). low model complexity, and 5) applicable to both white-box and black-box network inputs. The key idea of out method is to characterize the local dependency of the CNN neurons with respect to the network inputs to detect precarious Trojan insertion, which could be distinguished due to its incompatibility to the pristine distribution. A more detailed illustration of the one-pixel signature representation will be presented in Section 4.

In a Trojan attack, a backdoored CNN architecture is created by injecting a "virus" patterns (known as Trojan triggers [6]) into training images so that those "infected" images can be adversely classified (Fig. 3.b). In training, we however create a set of models with "fake virus", namely "vaccine" patterns, that are known to us (Fig. 3.a). By learning to differentiate the one-pixel signatures of those vaccinated models from signatures of the normal models, a classifier can be trained to detect a backdoored CNN in the presence of an unknown "virus" pattern. Our Trojan insertion setting is based on the widely used Single Target Attack proposed by BadNets [6], but with relaxed assumptions beyond the Single

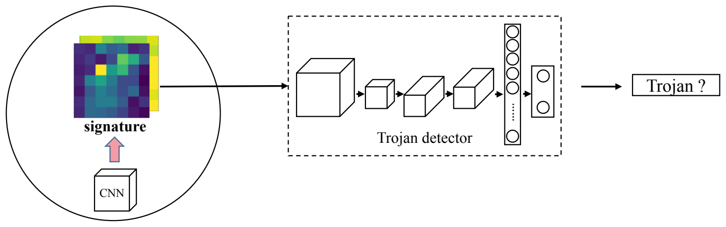
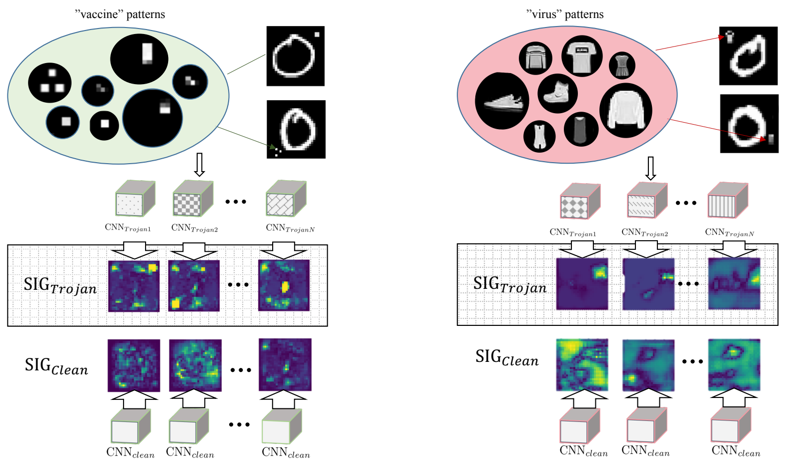


Fig. 2. Pipeline for our backdoored CNN detector using the one-pixel signature.

Target Attack by allowing multiple complex Trojan patterns with varying color and position (see Table 1).



(a) CNN_{Trojan} by the vaccine patterns (b) CNN_{Trojan} by the virus patterns

Fig. 3. Training and testing data generation pipeline for backdoored CNN detector. Note that each training sample is itself a CNN model which can be clean or backdoored. To evaluate our method, we generate random patterns as "vaccine" to create CNN_{Trojan} for training the backdoored CNN detector, as is shown to the left. In the right, we show how the testing CNN_{Trojan} are generated by using "virus" patterns (unknown to the backdoored CNN detector).

Generating CNN_{Trojan} with "vaccine" pattern and CNN_{Clean} for training

As shown in Fig. 3, we generate random patterns as the "vaccine" and insert them into the training images at random positions to train to obtain backdoored CNNs; each backdoored CNN itself becomes a positive sample. Some "vaccine patterns" are displayed in Fig. 3(a). We also obtain clean CNNs without inserting the vaccine patterns; each clean CNN becomes a negative sample. We first generated a set of randomly generated "vaccine" patterns. For half of the training set, we trained 300 CNN models with manipulated dataset that injected with "vaccine" pattern and labeled them as CNN_{Trojan}; for the other half, we trained the other set of 300 CNN models with the original dataset, and labeled those as CNN_{Clean}.

Generating $\mathbf{CNN}_{Trojan}$ with "virus" pattern and $\mathbf{CNN}_{Clean}$ for evaluation

We obtain a set of 300 backdoored CNN models using randomly selected Fashion-MNIST images as the "virus" patterns and a set of 300 clean CNN models. The "virus" patterns are injected in randomly selected position with random size and coloring. They are labeled as CNN_{Trojan} and CNN_{Clean} respectively as our test set.

$\mathbf{CNN}_{Trojan}$ Detection/Classification

Given the generated clean CNNs and backdoored CNN, we obtain the one-pixel signature image of of K channels for each CNN model. We then train a Vanilla CNN classifier as a backdoored CNN detector by taking the signature as the inputs to classify if a CNN model has a Trojan/backdoor or not. This process is illustrated in Fig. 2. To evaluate our problem, we create backdoored CNNs by using the Fashion-MNIST as the "virus" patterns, as shown in Fig. 3(b). The classifier is trained on 300 CNN_{Clean} and CNN_{Trojan} pairs with "vaccine" patterns and evaluated on 300 CNN_{Clean} and CNN_{Trojan} pairs with "virus" patterns, as described in section 3.3 and the pipeline is illustrated in Fig.2. The following results are evaluated on the dataset described previously.

4 One-Pixel Signature

Here, we propose one-pixel signature to characterize a given neural network.

4.1 Basic formulation

Given a space of CNN $\mathcal{F}$ where each CNN model $f \in \mathcal{F}$ takes an input image I of size $H \times W$ to perform K-way classification, our goal is to find a mapping $g : \mathcal{F} \to \mathbb{R}^{H \times W \times K}$ to produce a K-channel signature, which is defined as:

$$g(f) = (S_1^{(f)}, S_2^{(f)}, ..., S_K^{(f)})$$

where $S_k^{(f)} \in \mathbb{R}^{H \times W}$. A general illustration can be seen in Fig. 4. Before defining $S_k^{(f)}$, we first define a default image I_o which is either a constant value such as $\mathbf{0}$, or the average of all the training images. Define the pixel value of image I $\in [0, 1]$. Let the conditional probability $p_f(y = k|\mathrm{I}_o) \in [0, 1]$ denote the k^{th} entry of the classifier output, namely $[f(\mathrm{I}_o)]_k$. Furthermore, $\mathrm{I}_{i,j,v}$ refers to the image $\mathrm{I}(i, j) = v$, changing only the value of pixel (i, j) to v while keeping the all the rest of the pixel values the same as I_o. We attain $S_k^{(f)}(i, j)$ as the largest possible value of $|p_f(y = k|\mathrm{I}_{i,j,v})|$:

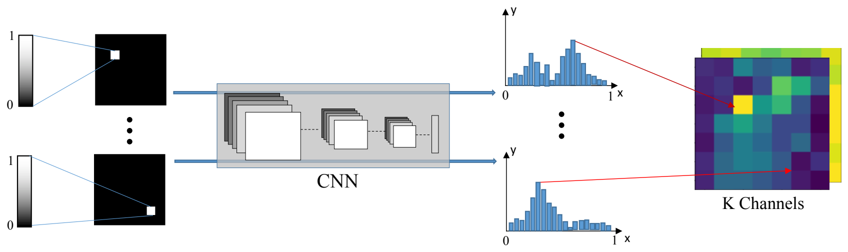


Fig. 4. Illustration for the generation of the **one-pixel signature** for a given CNN model. Based on a default image, each pixel is visited one-by-one; by exhausting the values for the pixel, the largest possible change to the prediction is attained as the signature for that pixel; visiting all the pixels gives rise to the signature images (K channels if making a K-way classification) for the given CNN model. See the mathematical definition in Eq. (1).

$$S_k^{(f)}(i,j) = \max_{v \in [0,1]} |p_f(y=k|\mathrm{I}_{i,j,v})|. \tag{1}$$

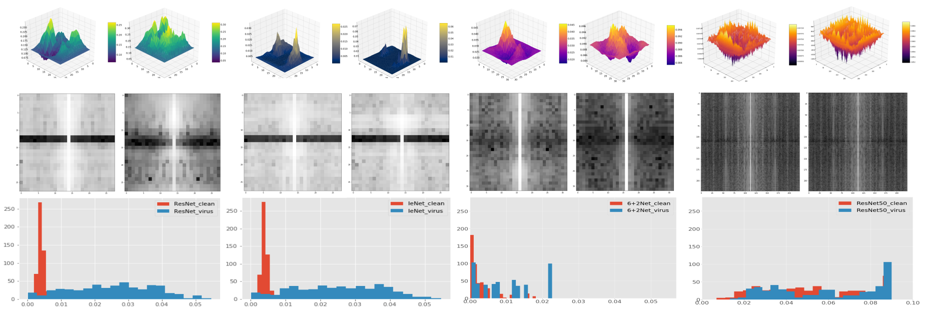
Algorithm 1 Outline for generating the one-pixel signature for model (classifier) f.

```
1: **Input:** $K$-way classifier (model) $f$ of input size $H \times W$; # of discrete values $V$;
   default image $\mathrm{I}_o$
2: **Output:** One-pixel signature $g(f) = SIG \in \mathbb{R}^{H \times W \times K}$
3: Initialize $SIG[H, W, K] \leftarrow 0$, $p_{\mathrm{I}_o} \leftarrow p_f(\mathrm{I}_o)$
4: for $i$ from 0 to $H-1$ do
5:     for $j$ from 0 to $W-1$ do
6:         $\mathrm{I} \leftarrow \mathrm{I}_0$, $temp[K] \leftarrow 0$
7:         for $v$ from 0 to $V-1$ do
8:             $\mathrm{I}[i,j] \leftarrow v/V$
9:             for $k$ from 0 to K-1 do
10:                if $|p_f(y=k|\mathrm{I})| > temp[k]$ then
11:                    $temp[k] = |p_f(y=k|\mathrm{I})|$
12:        $SIG[i,j] \leftarrow temp$
   return $SIG$
```

Eq. (1) looks for the significance that each individual pixel is making to the prediction. Since each $S_k^{(f)}(i,j)$ is computed independently, the computation complexity is relatively low. The overall complexity to obtain a signature for a CNN model f is $O(H \times W \times K \times V)$, where V is the search space for the image intensity. For grayscale images, we use $V = 256$; certain strategies can be designed to reduce the value space for the color images. In this paper, we compute the signature for colored images by simultaneously update values for all three channels, with demo signature images shown in Fig. 4. In this setup, the computational cost for colored images is linear w.r.t. to that for gray scale images. Eq. (1) can be computed for a black-box classifier f since no access is needed to the model parameters. Fig. 4 illustrates how signature images for classifier f are computed. Note that the definition of $S_k^{(f)}$ is not limited to Eq. (1) but we are not expanding the discussion about this topic here.

Algorithm. 1 illustrates the algorithmic outline for generating one-pixel signatures for a classifier taking single channel images, which has three input: the classifier f, the number of possible discrete values V and the default image I_o.

4.2 Visualization and illustration



(a) MNIST (b) fashion-MNIST (c) GTSRB (d) ImageNet

Fig. 5. Average 3D plot (first row), average spectra on frequency domain(second row), and variance histogram (third row) generated from 300 clean/virus one-pixel signature pairs of ResNet-8 trained on MNIST (a), LeNet-5 on MNIST (b), "6+2 Net" on GTSRB (c), and ResNet-50 on ImageNet (d) respectively. Left-side of each column and color red indicates CNN_{Clean}, right-side of each column and color blue indicates CNN_{Trojan}.

Our signature could also be considered as an approximation to the anomaly neuron activation map. Empirically, 1% of neurons are sufficient to enable the backdoor [6]; thus generating signatures pixel-wise from a black image is a black-box approximation of finding anomaly neurons by masking the remaining. If any point of the signature that is with a magnitude significantly greater than its surroundings, there might be a good chance that the corresponding activated neurons correspond to some local dependency. Visual analysis of average signature images Fig.5 illustrates that SIG_{Trojan} is noticeably different from SIG_{Clean}. Some patterns includes, SIG_{Trojan} contains a peak with greater magnitude than SIG_{Clean}. The max value from average SIG_{Clean} and SIG_{Trojan} is 0.27 vs 0.33 in MNIST, 0.025 vs 0.06 in fashion-MNIST, 0.03 vs 0.08 in GTSRB and 0.07 vs 0.90 in ImageNet. From the visualization of average frequency spectra, SIG_{Trojan} has more high-frequency noise comparing to SIG_{Clean}. Also,

SIG_{Trojan} usually have a larger variance than SIG_{Clean}. These observations persist in all four datasets, although complex dataset yields smaller difference.

4.3 Theoretical justification

Our proposed one-pixel signature g essentially characterizes the local dependency of the underlying neural network with respect to the network inputs. Before delving into the details, let us first reformulate the classification task in a probabilistic perspective. With some minor abuse of notations, we can denote the image I as a random variable taking values in the image space $\mathcal{I} \subset \mathbb{R}^{H \times W \times C}$; denote the label as another random variable y with the label space as $\mathcal{Y} = \{1, 2, ..., K\}$; C refers to the number of channels ($C = 3$ if I is a color image). Define the data distribution as P over $\mathcal{I} \times \mathcal{Y}$. When learning the classifier $f : \mathcal{I} \to \mathcal{Y}$, we are using f to model the conditional distribution $p(y|\mathrm{I})$.

We can then model this conditional distribution in the context of an undirected graphical model. In specific, each pixel in I is a node representing a random variable $\mathrm{I}_{i,j}$ whose support is $[0, 1]^C$; the label y is another node that is connected to all $\mathrm{I}_{i,j}$. The connectivity among the pixel nodes can be arbitrary. Denote the entire graph as G, and the collection of its maximal cliques as $Cliq(G)$. The structure of G is then illustrated in Fig. 4.3. By the Hammersley-Clifford Theorem [8], we can factor $p(y|\mathrm{I})$ using the graphical model into a product involving all of its maximal cliques:

$$p(y = k|I) = \frac{1}{Z(\mathrm{I})} \prod_{c \in Cliq(G)} \phi_c([\mathrm{I}]_c, y = k), \tag{2}$$

$$\text{where} \quad Z(\mathrm{I}) = \sum_{y=1}^{K} \prod_{c \in Cliq(G)} \phi_c([\mathrm{I}]_c, y).$$

The potential ϕ_c are some non-negative functions. The size of the maximal cliques reflects the local dependency of the input distribution. A pixel is conditionally independent of all the pixels outside its maximal clique given the values of other pixels within the clique. We have the k^{th} entry of the softmax prediction as $[f(\mathrm{I}_{i,j,v})]_k = p_f(y = k|\mathrm{I}_{i,j,v})$, as defined in Eq. (1). Combining it with Eq. (2), we have:

$$[f(\mathrm{I}_{i,j,v})]_k \propto \prod_{c \in Cliq(G)} \phi_c([\mathrm{I}_{i,j,v}]_c, y = k).$$

In reality, we approximate this graphical model via the CNN f and train it by maximum likelihood estimation. In a high level, we can decompose f into two parts. The first part is the convolution block whose each convolutional filter represents one potential function ϕ_c. The second part is a classifier, normally a multilayer perceptron, that aggregates these ϕ_c. For a CNN model that is trained to model the conditional probability, the activation pattern (the receptive field) of each of its filters tends to match the corresponding clique structure of the underlying graphical model. Due to the convolution operators in CNN, given a

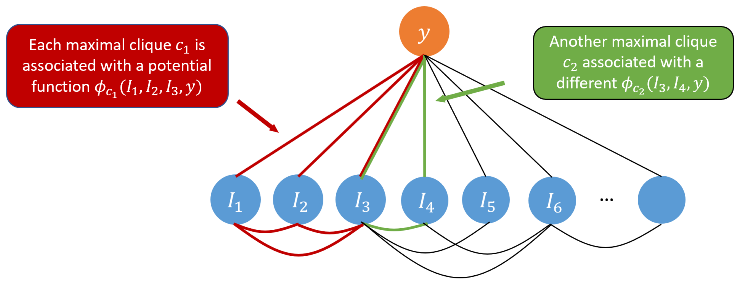

Each maximal clique c_1 is associated with a potential function $\phi_{c_1}(I_1, I_2, I_3, y)$
y
Another maximal clique c_2 associated with a different $\phi_{c_2}(I_3, I_4, y)$
I_1
I_2
I_3
I_4
I_5
I_6
...

Fig. 6. Illustration of the undirected graphical model where the orange node represents the label and the blue nodes represent the pixels. The joint probability can be factored into a product of potentials of the max cliques. For instance, given the two maximal cliques c_1 and c_2, where c_1 is the subgraph connected by the red edges and c_1 the subgraph connected by green edges, we then have their corresponding potential functions ϕ_{c_1} and ϕ_{c_2} defined.

pixel (i, j), the larger its associated cliques are, the more filters it will share with the nearby pixels (nodes) and thus the classifier becomes less sensitive to that single pixel. As a result, we propose to characterize the cliques structure by measuring how the function $h_k^{i,j}(v) = [f(I_{i,j,v})]_k$ varies within a small neighborhood of (i, j). Formally, given a distance function induced by some norm $||\cdot||$, we can use $||h_k^{(i,j)}||$ to derive an upper and lower bound of the average pairwise distance of the functions in the neighborhood of (i, j):

$$\sum_{\mathbf{p},\mathbf{q}} ||h_k^{\mathbf{p}}|| - ||h_k^{\mathbf{q}}|| \leq \sum_{\mathbf{p},\mathbf{q}} ||h_k^{\mathbf{p}} - h_k^{\mathbf{q}}|| \leq \sum_{\mathbf{p},\mathbf{q}} ||h_k^{\mathbf{p}}|| + ||h_k^{\mathbf{q}}||$$

where $\mathbf{p} \neq \mathbf{q}$ are pixels from the neighborhood of (i, j). For simplicity, we choose the supremum norm $||\cdot||_\infty$ with a discretization of the domain into V values. We therefore derive our signature S_k^f, whose (i, j) entry is $||h_k^{(i,j)}||_\infty$, to reveal the (local) activation patterns of the CNN model, which in turn, can be used to indicate the potential Trojan trigger pattern.

5 Experiments

We illustrate the effectiveness and efficiency of *one-pixel signature* in the backdoored CNN detection task by evaluating its performance in four datasets:

- **MNIST** [12]. A standard machine learning dataset which contains 60K training and 10K testing grayscale images of 10(0-9) hand-written digits. We select LeNet-5, ResNet-8, or VGG-10 for evaluation.
- **fashion-MNIST** [26]. The drop-in replacement of MNIST for benchmarking CNNs, which contains 60K training and 10K testing grayscale images of 10 fashion accessories. We also select LeNet-5, ResNet-8, or VGG-10 for evaluation.

- **GTSRB** [21]. German Traffic Sign Recognition (GTSR) Dataset is a commonly-used colored dataset to evaluate attacks on CNNs. This dataset consists of 39.2K training and 12,6K testing images with 43 different traffic signs. The CNN architecture trained on GTSRB consists of 6 convolution layers and 2 dense layers ("6+2 Net") in line with the set-up of Neural Cleanse [24].
- **ImageNet** [3]. A widely used large-scale image dataset with more than 14 million colored images of 20,000+ categories. Here we use a subset of 10 classes, where each class has 700 training and 300 testing samples, and ResNet-50 for comparison to the baseline method Neural Cleanse [24].

Throughout our experiments, we use Adam [10] as the default optimizer.

Same-architecture Detection In this scenario, we train and evaluate the backdoored CNN detector on CNNs of the same architecture. Specifically, we use LeNet-5 for MNIST and fashion-MNIST, "6+2 Net" for GTSRB, and ResNet-50 for ImageNet. For each aforementioned architecture and dataset pair, we use the one-pixel signatures generated from the 300 CNN_{Clean}/CNN_{Trojan} for training and those of another 300 model pairs for testing. For ImageNet we use 100 CNNs pairs instead of 300 in both training and testing.

Table 2. Backdoored CNN detection/classification rate on classic CNN models trained on MNIST, fashion-MNIST, GTSRB, and ImageNet dataset in comparison with the Neural Cleanse [24] algorithm. We also include result for the ABS [16] algorithm on the GTSRB dataset.

Methods	Datasets			
	MNIST	fashion-MNIST	GTSRB	ImageNet
Neural Cleanse [24]	60.53%	63.16%	60.71%	54.42%
ABS [16]	–	–	66.33%	–
One-Pixel Signature (ours)	**95.72%**	**94.59%**	**88.47%**	**88.69%**

As illustrated in Table 2, our method reaches Backdoored CNN detection rate of approximate 95% for the MNIST and fashion-MNIST dataset, 88% for GT-SRB dataset, and 89% for ImageNet, significantly outperforming Neural Cleanse [24] (around 60% on all datasets). The result on GTSRB for ABS is also much lower than ours.

We observe that the one-pixel signatures layouts of CNN_{Clean} and CNN_{Trojan} are even visually different, namely the one-pixel signatures of CNN_{Trojan} vary more drastically in pixel values, thus have a greater variance than CNN_{Clean} as shown in Fig. 5.

Cross-architecture Detection In more general and challenging scenarios where we are not able to narrow down which network architecture is used for the backdoored CNN model, our approach still achieves relatively high detection rate

Table 3. Backdoored CNN detection/classification rate on classic CNN models trained on MNIST when the architecture of the testing Backdoored model unknown.

Training	Testing	Detection Rate(%)
ResNet+VGG	LeNet	85.20 ± 5.85
LeNet+VGG	ResNet	76.50 ± 12.07
LeNet+ResNet	VGG	80.00 ± 3.57

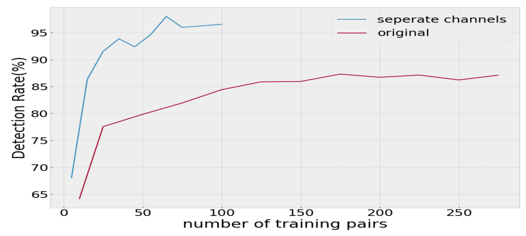


Fig. 7. Detection accuracy on GTSRB w.r.t. the number of training Clean/Backdoored model pairs in original and seperate channel setting.

on network architectures unseen during training. On MNIST, we train the detector on signatures generated from 2 out of the 3 network architectures (LeNet-5, ResNet-8, VGG-10) and evaluate the trained detector on CNNs of other architecture. We observe an average detection rate as high as 80%, reported in Table. 3. In short, our proposed one-pixel signature is architecture-agnostic and the Backdoored CNN detectors trained on top of it attain great generalizability.

Training with less samples To understand the sample complexity our method, here we investigate the minimum number of training models on four datasets without heavily downgrading the detection rate. We keep reducing 25 training pairs until zero and record the average detection rate for 20 detectors trained on the reduced training set at each time. The curve of the detection rate on GTSRB is shown in Fig. 7. Intuitively, we need more training models to detect CNN_{Trojan} that is trained on more complex datasets. Detectors on GTSRB dataset achieve an average detection rate of 87.31% with a reduced training set of size 175 Clean and 175 Backdoored models, whereas training set with 25 Clean and 25 Backdoored models is sufficient to train a backdoored CNN detector for models trained on MNIST to achieve approximately 95% detection rate. For fashion-MNIST, we need 50 Clean and 50 Backdoored models to achieve the same detection accuracy.

Use channels in one-pixel signature as individual training samples We could also treat each channel, instead of the whole one-pixel signature, as a positive/negative sample for training. Thus for models with K classes, the numbers of training samples increases by K times. We find that thereby training with only 15 pairs of CNN_{Clean} and CNN_{Trojan} models yields the same detection rate as training with 300 pairs of models previously. With 100 training pairs, we could achieve detection rate of 97% in GTSRB and 90% in ImageNet. A detailed comparison of the detection accuracy on GTSRB between using the individual channel and the original strategies is shown in Fig. 7 (a).

6 Ablation Study

All-to-one and all-to-all Trojan attack

Our detection technique can be effectively extended to other possible strategies like the all-to-one attack without compromising the detection rate. Under the scenario of an all-to-one attack, the attacker tries to force the model to classify all samples implanted with certain backdoor pattern into a targeted class. We see that the all-to-one attack is a variant of the one-to-one attack, and is even more likely to be detected by our method.

Similarly, we illustrate the effectiveness of our method on detecting all-to-all attack. We compare the detection success rate of our Trojan detector to that of Neural Cleanse and ABS on the testing set of 100 CNN models. It turns out that our Trojan detector can achieve a detection success rate greater than **95%**, while both Neural Cleanse and ABS show a poor success rate of around 50%.

Signature generation with different default images $\mathbf{I}_o$

One important parameter of our one-pixel signature method is the default image I_o. In our experiments, we empirically set the default image with a constant value of 0. Here we compare three different I_o setups: a) a constant value of 0, as the black image strategy; b) a constant value of 1, as the white image strategy; c) the pixel-wise mean of testing images, as the average testing image strategy; We are using a clean leNet-5 model trained on MNIST, and a backdoored leNet-5 model trained on poisoned MNIST dataset with Trojan trigger inserted at upper left corner, for evaluation. Both $I_o = 0$ and average image policy result in high detection rate. Since $I_o = 0$ policy do no require access to the training set, we regard this as our empirically optimal policy of the default image.

7 Conclusion

In this paper, we have developed a new backdoored CNN detection/classification method by designing a CNN signature representation that is revealing and easy to compute. It demonstrates a significant performance improvement over the existing competing methods with more general assumptions about the Trojan/backdoor attacks.

Acknowledgment This work is supported by NSF IIS-1717431 and NSF IIS-1618477. We thank Rajesh Gupta and Mani Srivastava for valuable discussions.

References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access **6**, 14410–14430 (2018)
2. Chua, C.S., Jarvis, R.: Point signatures: A new representation for 3d object recognition. International Journal of Computer Vision **25**(1), 63–85 (1997)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
4. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT Press (2016)
5. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2014)
6. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017)
7. Guo, W., Wang, L., Xing, X., Du, M., Song, D.: Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. arXiv preprint arXiv:1908.01763 (2019)
8. Hammersley, J.M., Clifford, P.: Markov fields on finite graphs and lattices. Unpublished manuscript **46** (1971)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
12. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Backpropagation applied to handwritten zip code recognition. In: Neural Computation (1989)
13. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436 (2015)
14. Lindeberg, T.: Scale-space theory in computer vision, vol. 256. Springer Science & Business Media (2013)
15. Liu, K., Dolan-Gavitt, B., Garg, S.: Fine-pruning: Defending against backdooring attacks on deep neural networks. In: International Symposium on Research in Attacks, Intrusions and Defenses. pp. 273–294 (2018)
16. Liu, Y., Lee, W.C., Tao, G., Ma, S., Aafer, Y., Zhang, X.: Abs: Scanning neural networks for back-doors by artificial brain stimulation. In: ACM SIGSAC Conference on Computer and Communications Security. p. 1265–1282 (2019)
17. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: CVPR. pp. 5188–5196 (2015)
18. OFFICE, U.A.R.: W911nf-19-s-0012. In: U.S. ARMY RESEARCH OFFICE BROAD AGENCY ANNOUNCEMENT FOR TrojAI (2019)
19. Prakash, A., Moran, N., Garber, S., DiLillo, A., Storer, J.: Deflecting adversarial attacks with pixel deflection. In: CVPR (2018)
20. Qiao, X., Yang, Y., Li, H.: Defending neural backdoors via generative distribution modeling. In: Advances in Neural Information Processing Systems. pp. 14004–14013 (2019)
21. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In: IEEE International Joint Conference on Neural Networks. pp. 1453–1460 (2011)

22. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation (2019)
23. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
24. Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: IEEE Symposium on Security and Privacy (2019)
25. Witkin, A.P.: Scale-space filtering. In: Readings in Computer Vision, pp. 329–332. Elsevier (1987)
26. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
27. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR (2017)
28. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833 (2014)
29. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: International Conference on Learning Representations (2017)