CatchBackdoor: Backdoor Detection via Critical Trojan Neural Path Fuzzing

Haibo Jin[1], Ruoxi Chen[2], Jinyin Chen[3,2], Haibin Zheng[3,2], Yang Zhang[1], and Haohan Wang[1]

[1] School of Information Sciences, University of Illinois Urbana-Champaign
{haibo,yzhangnd,haohanw}@illinois.edu
[2] College of Information Engineering, Zhejiang University of Technology
{2112003149, chenjinyin}@zjut.edu.cn
[3] Institute of Cyberspace Security, Zhejiang University of Technology
haibinzheng320@gmail.com

Abstract. The success of deep neural networks (DNNs) in real-world applications has benefited from abundant pre-trained models. However, the backdoored pre-trained models can pose a significant trojan threat to the deployment of downstream DNNs. Numerous backdoor detection methods have been proposed but are limited to two aspects: (1) high sensitivity on trigger size, especially on stealthy attacks (i.e., blending attacks and defense adaptive attacks); (2) rely heavily on benign examples for reverse engineering. To address these challenges, we empirically observed that trojaned behaviors triggered by various trojan attacks can be attributed to the trojan path, composed of top-k critical neurons with more significant contributions to model prediction changes. Motivated by it, we propose CatchBackdoor, a detection method against trojan attacks. Based on the close connection between trojaned behaviors and trojan path to trigger errors, CatchBackdoor starts from the benign path and gradually approximates the trojan path through differential fuzzing. We then reverse triggers from the trojan path, to trigger errors caused by diverse trojaned attacks. Extensive experiments on MINST, CIFAR-10, and a-ImageNet datasets and 7 models (LeNet, ResNet, and VGG) demonstrate the superiority of CatchBackdoor over the state-of-the-art methods, in terms of (1) *effective* - it shows better detection performance, especially on stealthy attacks ($\sim \times$ 2 on average); (2) *extensible* - it is robust to trigger size and can conduct detection without benign examples.

Keywords: Backdoor detection · Neural path · Fuzzing

1 Introduction

Recent advancements in deep neural networks (DNNs) have led to their widespread application in various fields. A key factor in this progress has been the use of pre-trained models, notably facilitated by resources like the Model Zoo, Hugging Face, etc., which offers a vast collection of such models for free download. However, such a practice also raises security concerns, particularly the risk of

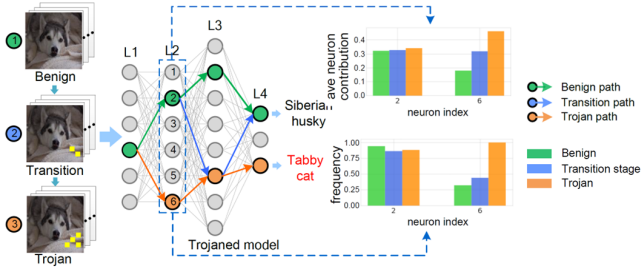


Fig. 1: Neural paths simulated by benign and trojaned examples. Dataset: 3-class ImageNet subset; Attack: BadNets (trojan rate=0.1); Target Classification: "Tabby Cat." The bars, colored in green, blue, and orange, represent the average neuron contribution and activation frequency within the model's second layer, transitioning from benign to trojan states.

trojaned models, which are the models that perform well on benign examples but expose wrong/targeted predictions when the input contains the trigger [24]. These models, susceptible to trojan manipulations during training [7, 28] or parameter modification [20], may perform accurately on standard inputs but fail or act maliciously when triggered. Thus, it is of great importance to conduct backbook detection to ensure we can confidently rely on DNNs for critical tasks [11].

Existing detection methods are mostly developed along two mainstream: data inspection (i.e., detecting trojaned examples from the training data), and model inspection (i.e., hunting potential backdoors inside the pre-trained model). The data inspection methods [3, 6, 35, 37] aim to distinguish benign examples from trojaned ones. However, they are infeasible in practical scenarios like Model Zoo, where training data of the online model is not available to downstream defenders.

Different from data inspection approaches, model inspection methods [8,19,29, 39,40] aim to examine trained models for any potential backdoors. Trigger reverse-engineering is the most representative method [8, 19, 39, 43], which searches for triggers with specific victim labels. Existing research [8,19,39] commonly assumes that backdoor triggers are static and small in size. However, this assumption does not hold for more advanced attacks with by large and dynamic triggers [4, 18, 26]. Therefore, the optimized triggers reversed under this assumption often struggle to activate errors in these advanced attacks, leading to unsatisfactory detection results. Besides, these methods rely on benign examples for trigger reverse engineering, therefore they will be challenged when benign data is unavailable.

To address these above challenges, in this work, we try to summarize the general cause of different trojan attacks from neurons in DNNs. In particular, the DNNs' decision results are determined by the nonlinear combination of each neuron, so it is intuitive to construct the relationship between neuron behaviors and trojan attacks. Thus, we investigate neurons that play a decisive role in trojaned behaviors and data flow between them. In particular, we measure the extent of a neuron's contribution to the variations in prediction results. By linking neurons with more significant contributions to model prediction changes, we

propose the concept of *neural path*, which represents the vulnerable direction of triggering decision changes.

We show the relationship between neural path and trojaned behaviors in Fig. 1. As observed, benign examples, each with a distinct label, activated unique neural paths. Conversely, trojaned examples based on these benign examples activated consistent neural paths, distinctively different from their benign counterparts. During the transition phase, the neural path is activated when inputs contain partial elements of the trigger, and increasingly align with the trojan paths, both in neuron contribution and activation frequency.

Further, following the concept of *neural path* and motivated by the empirical observation as demonstrated in Fig. 1, we design a novel backdoor detection method independent of the trigger size, namely CatchBackdoor. It identifies critical neurons with more significant contributions to trigger model prediction changes to form the benign path and detect trojaned models by fuzzing benign paths to the approximate trojan path, from which triggers can be reversed.

The main contributions are summarized as follows.

- We introduce the concept of the neural path, and empirically gain an insight that trojaned DNN behaviors are attributed to the trojan path, i.e., a neural path consisting of neurons dominant in model prediction changing.
- Motivated by the observation, we introduce CatchBackdoor, a novel method for detecting potential backdoors in DNNs. It fuzzs benign path to construct trojan path, to trigger trojan behaviors, even without benign examples.
- Comprehensive experiments have been conducted on 3 datasets and 7 self-trained models to verify the effectiveness and efficiency of CatchBackdoor. It outperforms the state-of-the-art (SOTA) baselines in identifying potential backdoors, especially on stealthy attacks ($\sim \times 2$).

2 Related Work

Trojan attacks. Trojan attack injects hidden malicious backdoors into the model, which can cause misclassification when the input contains a specific pattern called a trigger. In general, they could be categorized into four types: modification attacks, blending attacks, neuron hijacking attacks and defense adaptive trojan attacks. Modification attacks mainly modify a single pixel or a pattern on images to reach trojan effects [7, 27, 36]. Blending attacks mainly blend one class latent representation to other classes [1, 21, 26, 28, 31]. Neuron hijacking attacks mainly optimize pre-defined triggers combined with specific neurons [17, 20]. Defense adaptive attacks are mainly designed to achieve trojan attacks while bypassing possible detections [4, 18, 32]. For blending and defense adaptive attacks, they tend to change all pixels in images, i.e., triggers are large and dynamic. Besides, latent features of benign and trojaned examples learned by the backdoored model are hard to distinguish. Therefore, these attacks are stealthy towards detection algorithms that identify backdoors via cluster and separate analysis in latent representation space.

Trojan detections. Several detection approaches have been proposed to hunt trojan attacks. They can be divided into data inspection and model inspection, responsible for detecting trigger inputs and trojaned models, respectively. Data inspection distinguishes [3, 6, 9, 22, 33, 35, 37] benign examples from trojan ones through the difference of characteristic distribution. Model inspection is used to determine whether the model is trojaned, which is directly relevant to our work. Some methods are based on the assumption that the backdoor trigger is static with small size [8, 19, 29, 39]. So they do not perform well on blending attacks and defense adaptive attacks. Other methods train additional models for detection [13, 43]. The effectiveness of these methods highly depends on external training data.

3 Preliminaries

3.1 Definitions

In this study, our examination is centered on DNNs on tasks related to image processing. We consider an input image $x \in X$, where $X = \{x_1, x_2, \ldots\}$ represents the set of all possible inputs. Each neuron in the network is denoted as $n_{i,j}$, representing the j-th neuron in the i-th layer. This study primarily focuses on the concept of a neural path within such a network.

The activation value of a neuron $n_{i,j}$ for an input x is denoted as $\varphi_{n_{i,j}}(x)$. This value is the average of the feature map $A_{n_{i,j}}(x) \in \mathbb{R}^{Height \times Width \times Channel}$. A neuron is said to be activated if its activation value $\varphi_{n_{i,j}}(x)$ is greater than zero. We then formally define the neuron contribution and neural path as follows.

Definition 1 (Neuron Contribution). Given a DNN with parameters θ, when fed with input example x and its ground truth y. The neuron contribution of $n_{i,j}$ is calculated as:

$$\xi_{n_{i,j}}(x) = \frac{\partial \mathcal{L}(x, y, \theta)}{\partial \varphi_{n_{i,j}}(x)} \tag{1}$$

where $\xi_{n_{i,j}}(x)$ denotes the neuron contribution with input x. ∂ denotes the partial derivative function. Neuron contribution reflects the influence of neuron activation value on model decision. Neurons with larger neuron contributions are more dominant in change effect model predictions.

Definition 2 (Neural Path). In a l-layer DNN, for an input $x \in X$, the neural path is conceptualized as a sequence of interconnected neurons that have a significant influence on the model's decision for that input. It is defined as:

$$\Psi(x) = \bigcup_{i=1}^{l-1} \bigcup_{j=1}^{k} \{n_{i,j}, \text{Data Flow}(n_{i,j}, n_{i+1,j})\} \tag{2}$$

where $n_{i,j}$ are neurons with high contribution values, and Data Flow$(n_{i,j}, n_{i+1,j})$ signifies the connection facilitating forward propagation between consecutive neurons. The neural path, therefore, represents the critical route through which data travels within the DNN, influencing its output predictions.

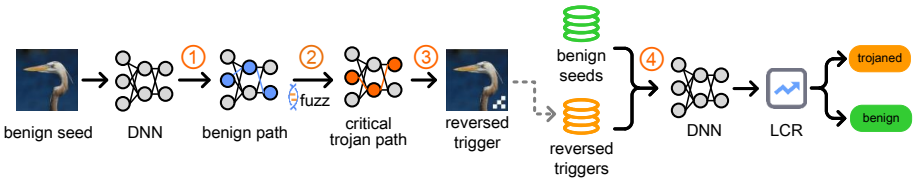


Fig. 2: The framework of CatchBackdoor: A benign seed (random noise or benign input) is input to the DNN to create a benign neural path. We then fuzz this path by maximizing neuron contribution, converging to a critical trojan path, which generates testing examples. A batch of benign seeds yields the same number of testing examples. These, along with the benign seeds, are fed into the DNN. CatchBackdoor determines if the model is trojaned by calculating the label change rate (LCR); a higher LCR indicates a higher probability of being trojaned.

3.2 Threat Model

Attacker. We assume attackers have access to the training data [44, 45], i.e., either editing the training data or adding extra data to the training dataset. They have knowledge of network architecture and the training algorithm of the target model. They can trojan the model from scratch, fine-tune the model from trojaned examples and labels, or retrain the model with selected neurons and weights. We consider trojan attacks with only one trigger with one trojaned label.
Defender. Following recent backdoor detection studies [19,39,43], defenders have white-box access to the model. Besides, they need part of the clean validation set, i.e., at least one input for each label should be provided. They do not have prior knowledge of the potential trojan backdoor.

Note that, CatchBackdoor stands on the defender role, it only acquires a few images from the clean validation set or can reverse triggers from random noise, which can imitate the activation path.

4 CatchBackdoor

An overview of CatchBackdoor is presented in Fig. 2. It consists of four steps: ① benign path construction, ② critical trojan neural path identification through differential fuzzing, ③ trigger reverse engineering, and ④ trojaned model determination. For brevity in expression, benign neural path and trojan neural path are dubbed as "benign path" and "trojan path" in the rest part of our paper.

4.1 Benign Path Construction

Constructing a benign path is a critical step for ensuring network integrity. This process involves identifying and linking neurons that significantly contribute to the network's output while excluding neurons from the fully connected layer to preserve the diversity of the paths. The pseudo-code is in **supplementary materials**, which outlines the steps for constructing a benign path.

In practice where training data is not available (e.g., Model Zoo [38]) for defenders, the benign path is constructed using the model structure along with random noise. This noise, when introduced into the DNN, is categorized into one of the labels, effectively simulating the benign path that would be activated by actual training data. We will discuss it in the **supplementary materials**.

4.2 Critical Trojan Path Identification Through Fuzzing

We conduct path fuzzing, to gain the critical trojan path, from which triggers will be reversed afterward.

The pseudo-code is in **supplementary materials**, which presents the steps involved in the identification of the critical trojan path for benign image x. During fuzzing, we aim to maximize neuron contributions within the benign path. We calculate the activated frequency of each neuron in each neural path from the aggregation of fuzzed paths. For each input, we link critical trojan neurons with data flow to form critical trojan path, which can trigger the correct trojaned label as trojaned examples do. For the well-trained trojaned model, there exists only one trojaned label. Not surprisingly, at the end of fuzzing, the fuzzed path will gradually converge to the path that triggers the trojaned label, i.e., the critical trojan path. By incorporating neurons from the fully connected layer and analyzing activation frequencies, it can accurately represent a trojaned model's behavior.

Proposition (Converge to one specific critical trojan path).

Suppose $x_1, x_2 \in X$ are two different benign seeds that activate different benign paths. Assume function $G(\Psi(x)) \Leftrightarrow F(x)$ denotes the mapping from neural path activated by x to the model prediction. After fuzzing, the critical trojan path is:

$$\begin{aligned} &\forall\ x_1,\ \exists\ x_2, x_{troj}\ \ \textbf{s.t.}\ x_1 \neq x_2 \neq x_{troj}, \Psi_b(x_1) \neq \Psi_b(x_2) \Psi_{cr}(x_1) = \Psi_{cr}(x_2), \\ &G(\Psi_{cr}(x_1)) = G(\Psi_{cr}(x_2))) \Leftrightarrow F(x_{troj}) \end{aligned} \tag{3}$$

where x_{troj} denotes a trojaned example. Different benign seeds may construct different benign paths, but they finally converge to one specific trojan path, i.e., critical trojan path, which resembles that activated by real triggers. Thus, the effect of the critical trojan path is similar to that of the real trojan path.

Determination of path convergence in practice. We further examine whether the critical trojan path converges through additional iterations, denoted as S', where $S' < S$. Convergence is determined when the composition of neurons in the $\Psi_{cr}(x)$ remains consistent across iterations, indicating the successful identification of a potential critical trojan path. We assume that the fuzzed path finally converges to the critical trojan path.

4.3 Trigger Reverse Engineering

As stated, the critical trojan path has a similar effect to trojan path. Thus, we leverage it to reverse triggers to achieve the same effect as trojaned examples, i.e., triggering errors due to backdoors.

For an input benign seed x, a trigger t is reversed by taking the partial derivative of the critical trojan path. A testing example x_t is generated by adding the reversed trigger t to the benign seed x. They are calculated as follows:

$$t = \frac{\partial \Psi_{cr}(x)}{\partial x}, \quad x_t = \min(\max(\mu \times t + x, 0), 255) \tag{4}$$

where $\mu \in [0, 1]$ controls the transparency of perturbations, which is usually set to 0.5. The pixel values of testing examples x_t are limited within [0, 255].

4.4 Trojaned Model Determination

If the input contains a trigger, the prediction of trojaned models will be the trojaned label. Therefore, given the trojaned model with its multiple trojaned examples, the label that most frequently appears is considered to be the target label, i.e., the trojaned label. Thus, if we count the number of target label that appears due to examples carrying triggers (real or reversed), trojaned models can be distinguished from benign ones.

We define the LCR as the detection standard, which counts the label change predicted by reversed examples. We feed reversed examples $R = \{x_{r1}, x_{r2}, ...\}$ into the model. The LCR of R is defined as follows:

$$\text{LCR} = \frac{\sum_{i=1}^{N} 1 \mid n_{F(x_{ri}=y^c)}}{N}, x_{ri} \in R, c \in C \tag{5}$$

where N is the total number of reversed examples, C is the aggregation of total labels, and $n_{F(x_{ri})=y^c}$ denotes the number of the target label y^c.

We assume reversed examples will trigger high LCR, if a model has a potential backdoor, and vice versa. Considering the influence of false-positive examples, we set the threshold of LCR, $\lambda = 50\%$ for all datasets conducted in our experiment. If more than 50% of the predicted labels turn to one specific label y^c, we consider y^c as the trojaned label ($y^c = y_t$) and the model is very likely to be trojaned. The specific number of this threshold will be further discussed in Section 6.2.

We further investigate the relationship between neural path and the DNN decision, the detailed results are presented in the **supplementary materials**.

5 Neural Path and Trojaned Models: A Case Study

5.1 Neural Path Controls Model Predictions

We first study the relationship between neural path and the DNN decision. First, we mask the top-k neuron path of the original class to zero. Then we move neuron contribution values of the top-k neuron path from the target class to replace those in the original class in the corresponding index. We check whether the predicted result is consistent with the targeted result after the operation. Consistency rate of the classifier is defined as:

$$\textit{consistency rate} = \frac{\{x | x \in X \cap f_r(x) = y_t\}}{num(X)} \tag{6}$$

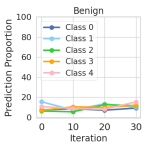


Fig. 3: The proportion of different prediction classes.

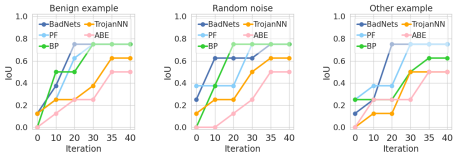


Fig. 4: IoU of critical trojan path among different attacks under different iterations.

where X is a set of test cases, $f_r(x)$ is the prediction result of the classifier after replacing f_r on the input x. y_t is the target class. Intuitively, the higher the consistency rate, the larger the impact of the neural path.

We randomly select 500 benign examples and the target class is 1, 5, and 9. We calculate the consistency rate after replacing the top-1 neural path. As observed in Table 1, after replacing, almost all predictions flip to the target class. We can conclude neural path dominates the decision of the DNN and each predicted class can be attributed to a certain neural path. Based on it, we assume if there exists only one trojan class in the trojaned model, we can find the unique trojan path responsible for this class.

Table 1: Consistency rate after replacing neural path from the certain class.

Datasets	Models	Target Class		
		Class 1	Class 5	Class 9
MNIST [16]	LeNet-5 [16]	96.6%	94.2%	86.2%
CIFAR-10 [14]	AlexNet [15]	87.4%	90.4%	84.4%

We illustrate the proportion of different classes in predictions under different fuzzing iterations in Figure 3. We use a benign LeNet-5 of MNIST (left) and two models backdoored by BadNets (middle) and BullseyePolytope (BP) [1] (right) with trojan rate=0.05 and trojaned label 0. For the benign model, the distribution probability of the predicted class seems relatively average. While for trojaned models, the majority of the predicted labels turn to the trojaned label (e.g., class 0), whose proportion rate is much higher than the benign model. Consistent with our assumption in Section 4.4, we consider the most frequently appearing label during fuzzing as the trojaned label.

5.2 Neural Path Converges

We empirically verify the assumption in Section 4.2. Specifically, we compare the similarity between the top-1 fuzzed neural path and the critical neural path in the trojaned LeNet-5 of MNIST during iterations in Figure 4. These models are trojaned by BadNets, Poison frogs (PF) [28], BP, TrojanNN [20] and adversarial backdoor embedding (ABE) [32], with trojan rate=0.05 and trojaned label 0. The fuzzing start from a benign example from the corresponding dataset (left), random noise (middle) and a random example from other datasets (right). For consistency measurement, we use Intersection over union (IoU), which is calculated

as $IoU(X, X') = \frac{num(X) \cap num(X')}{num(X) \cup num(X')}$, where X and X' denote two batches of input examples. In particular, a greater value denotes higher consistency.

As observed, IoU value increases with the growing number of iterations, i.e., the fuzzed path gradually approximates the critical trojan path. After the 35th iteration, IoU value remains stable at around 0.6 and the fuzzed path finally converges. Besides, IoU value seems quite similar at 35th iteration when fuzzing starts from benign example, random noise, or even an example from other datasets. This shows that the critical trojan path is independent of the benign seed. So when the training data is not available, random noise can serve as the benign seed for generating reversed examples, which will be verified in Section 6.4.

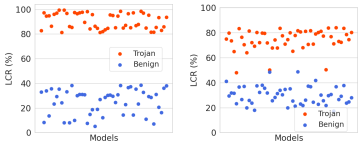
6 Evaluation

6.1 Setup

Datasets and Models. We conduct experiment on MNIST, CIFAR-10 and a-ImageNet - a subset of 10 classes of animals in ImageNet [25]. For MNIST, we use small models (LeNet-1, LeNet-4, LeNet-5) [16]. On CIFAR-10, AlexNet and ResNet20 [10] are adopted. On a-ImageNet, we adopt larger and deeper models VGG16 [30] and VGG19 [30].
Trojan attacks. We use 11 attacks for evaluation, including modification attacks (BadNets [7], Dynamic Backdoor [27]), blending attacks (Poison frogs [28], Hidden trigger [26], BullseyePolytope (BP) [1] and Sleeper Agent [31]), neuron hijacking attacks (TrojanNN [20] with face and apple stamp, and neuron frequency-based attack, SIG [2]), and defense adaptive attacks (adversarial backdoor embedding (ABE) [32], and deep feature space trojan attack (DFST) [4]). DFST can not handle gray-scale images so it is not conducted on MNIST dataset.
Baselines. We adopt SOTA detection algorithms as the baselines, including NC [39], TABOR [8], ABS [19], TND [40], K-Arm [29], ANP [42], TopoTrigger [12] and UNICORN [41]. The parameters for these algorithms are configured following their settings reported in the respective papers.
Metrics. The metrics used in the experiments are classification accuracy (acc), attack success rate (ASR), and label change rate (LCR).
Implementation details. In the default setting, CatchBackdoor adopts benign examples to construct the benign path. We set k_1 and k_2=3, LCR threshold λ=0.5, unless otherwise specified. We will study the impact of it in parameter sensitivity analysis. To mitigate non-determinism, we repeated the experiment for 3 times and reported the average results. For all the tables, unless otherwise stated, “N" means the trojaned model cannot be detected. A more detailed setup is shown in the **supplementary materials**.

6.2 LCR for Detecting Trojaned Models

We calculate LCR of benign and trojaned models, and then investigate Area under Curve (AUC) score using LCR as the indicator of identifying the model as benign or trojaned under different threshold values.



(a) MNIST **(b)** a-ImageNet

Fig. 5: LCR of benign and trojaned models on different datasets.

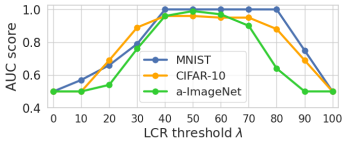


Fig. 6: AUC score under different threshold of LCR.

LCR on Benign And Trojaned Models We randomly select 1,000 reversed examples by CatchBackdoor and calculate the average LCR on 50 benign and 50 trojaned models. Results on MNIST and a-ImageNet are shown in Fig. 5. Red and blue scatters represent LCR results of trojaned and benign models, respectively. As observed, there is a significant difference between the LCR of benign and trojaned models. Specifically, most trojaned models from different kinds of attacks have larger LCRs than benign models under different trojan rates. This is consistent with our assumptions discussed in Section 4.4. We can see that the LCR of benign models is very low (i.e., lower than 50%) and that of trojaned models is much higher. Benign and trojaned models can be easily distinguished if we set the proper threshold of LCR=0.5.

LCR And AUC Analysis We calculate AUC under LCR with different thresholds for trojaned model determination. We adopt 100 models used in Section 6.2. Fig. 6 presents AUC under different thresholds of λ. We can observe that when λ is set between 0.4 and 0.6, the AUC results are usually larger than 0.9, which suggests that we could achieve detection accuracy using LCR to distinguish trojaned models. LCR can be an indicator for the detection of trojaned models when $\lambda = 0.5$. In the following experiment, we set $\lambda = 0.5$. The model is considered to be trojaned if the LCR exceeds 0.5. Besides, LCR can also reflect the effectiveness of triggering errors by a certain number of reversed triggers, i.e., methods that can reach higher LCR is more effective in trojaned model detection. We believe that LCR is a more suitable evaluation metric for evaluating backdoor detection. In the follow-up, we use LCR for measurements.

6.3 Detection Effectiveness

We compare CatchBackdoor with SOTA baselines in trojaned model detection based on LCR for effectiveness verification.

Implementation details. We use 50 trojaned models with trojan rate 0.1, 0.15, 0.2, 0.25, and 0.3. For each trojan rate, we train 10 models respectively. ASR of all trojaned models is over 90%. 500 reversed examples are generated from benign examples for each model. Results are shown in Table 2.

Results and analysis. CatchBackdoor can detect all trojaned models and achieves the highest LCR in most cases, especially on stealthy attacks ($\sim \times 2$).

This means that our generated examples can effectively trigger more label changes than baselines.

Table 2: Comparison of LCR against various trojan attacks.

Datasets	Models	Methods	Modification		Blending			Neuron Hijacking			Defense	Adaptive	
			BadNets	Dynamic Backdoor	Poison frogs	Hidden Trigger	BP	Sleeper Agent	TrojanNN (Apple)	TrojanNN (Face)	SIG	ABE	DFST
MNIST	LeNet-1	NC	82.6%	78.6%	N	N	65.2%	61.4%	60.4%	51.6%	61.6%	N	/
		TABOR	83.2%	79.2%	N	N	62.4%	68.0%	70.6%	69.2%	65.2%	N	/
		ABS	89.8%	81.2%	68.0%	N	77.6%	60.2%	76.8%	81.8%	67.8%	N	/
		TND	88.8%	82.2%	65.2%	N	70.0%	72.6%	70.4%	64.2%	79.6%	N	/
		K-Arm	**92.0%**	84.0%	78.4%	62.8%	78.2%	74.8%	78.2%	80.2%	84.2%	57.0%	/
		ANP	**91.4%**	82.6%	78.2%	58.8%	63.2%	54.6%	67.4%	69.0%	74.0%	64.8%	/
		TopoTrigger	**90.8%**	81.2%	90.8%	70.4%	**80.2%**	70.2%	75.2%	78.2%	83.8%	80.6%	/
		UNICORN	**91.2%**	83.4%	92.1%	69.0%	77.0%	72.4%	80.0%	81.4%	81.6%	81.2%	/
		CatchBackdoor	91.0%	**84.6%**	**94.0%**	**72.6%**	77.8%	**78.0%**	**81.6%**	**82.4%**	82.0%	**86.0%**	/
	LeNet-4	NC	82.8%	80.0%	N	N	72.0%	64.2%	63.0%	55.4%	69.4%	N	/
		TABOR	90.4%	83.2%	N	N	72.4%	78.0%	63.8%	62.0%	64.8%	N	/
		ABS	**92.0%**	83.4%	79.0%	58.8%	84.2%	82.4%	79.2%	70.2%	79.0%	N	/
		TND	90.2%	80.2%	64.2%	N	78.6%	82.0%	64.4%	65.0%	72.4%	N	/
		K-Arm	91.0%	78.6%	82.6%	58.8%	**85.0%**	**86.8%**	80.0%	71.0%	78.0%	59.8%	/
		ANP	86.2%	79.8%	74.2%	47.8%	73.4%	76.2%	59.8%	61.6%	73.4%	53.8%	/
		TopoTrigger	77.0%	83.6%	79.2%	76.8%	71.8%	80.8%	72.6%	70.6%	82.8%	**93.4%**	/
		UNICORN	92.8%	75.8%	71.6%	65.6%	86.0%	73.2%	77.8%	65.8%	86.2%	79.8%	/
		CatchBackdoor	91.2%	**83.8%**	**91.0%**	**70.6%**	83.2%	82.0%	**81.0%**	**73.6%**	85.4%	87.0%	/
	LeNet-5	NC	80.6%	78.4%	N	N	N	56.0%	61.6%	62.6%	78.4%	N	/
		TABOR	89.2%	80.0%	N	N	66.4%	64.6%	64.4%	59.4%	72.0%	N	/
		ABS	91.8%	**83.6%**	74.0%	51.8%	78.6%	74.4%	75.2%	71.8%	80.0%	N	/
		TND	90.7%	81.4%	N	N	72.6%	71.0%	68.6%	64.8%	80.6%	N	/
		K-Arm	**93.4%**	81.6%	84.4%	58.2%	78.4%	**78.0%**	74.0%	**81.2%**	83.4%	N	/
		ANP	81.8%	62.8%	69.8%	60.4%	58.8%	68.6%	58.8%	64.0%	73.6%	N	/
		TopoTrigger	94.0%	79.2%	84.6%	56.6%	66.4%	59.8%	66.8%	59.8%	85.4%	67.8%	/
		UNICORN	91.6%	87.6%	82.8%	77.6%	65.6%	76.0%	58.8%	78.0%	80.6%	82.2%	/
		CatchBackdoor	92.2%	82.6%	**93.0%**	**74.8%**	**80.0%**	**78.0%**	**78.0%**	80.2%	82.4%	**91.0%**	/
CIFAR-10	AlexNet	NC	81.6%	74.6%	N	N	N	N	51.2%	N	54.6%	N	N
		TABOR	87.6%	76.6%	N	N	56.0%	N	67.2%	66.6%	57.2%	N	N
		ABS	90.4%	79.4%	75.0%	N	68.0%	72.6%	75.8%	64.2%	51.0%	N	N
		TND	89.5%	77.6%	50.8%	52.6%	70.8%	70.0%	70.8%	60.0%	64.6%	N	N
		K-Arm	92.0%	82.0%	75.2%	58.6%	**74.2%**	66.6%	76.8%	76.4%	72.8%	N	51.2%
		ANP	86.6%	78.4%	63.2%	66.2%	61.8%	N	66.2%	54.6%	74.0%	N	N
		TopoTrigger	82.6%	72.0%	78.8%	67.4%	65.4%	53.2%	73.8%	61.4%	63.6%	77.4%	60.0%
		UNICORN	92.0%	72.6%	82.0%	68.8%	69.0%	67.2%	**80.8%**	63.6%	79.6%	79.8%	63.2%
		CatchBackdoor	**93.6%**	**83.2%**	**87.0%**	**70.8%**	72.4%	**70.2%**	79.4%	**79.2%**	**80.8%**	**83.0%**	**61.4%**
	ResNet20	NC	83.6%	78.8%	N	N	N	N	60.0%	53.8%	64.6%	N	N
		TABOR	84.0%	79.8%	N	N	56.8%	N	63.0%	50.2%	60.6%	N	N
		ABS	89.6%	82.8%	77.0%	53.2%	72.0%	N	81.0%	60.6%	70.0%	N	56.8%
		TND	88.3%	78.2%	N	N	68.8%	N	74.8%	51.6%	68.4%	N	N
		K-Arm	91.0%	79.4%	81.4%	70.2%	72.0%	51.4%	81.2%	78.0%	75.2%	N	56.8%
		ANP	80.2%	73.4%	N	N	60.0%	33.4%	75.4%	52.4%	54.0%	N	N
		TopoTrigger	89.6%	78.2%	83.4%	70.0%	65.4%	68.6%	70.2%	71.6%	72.0%	78.0%	53.8%
		UNICORN	87.8%	84.0%	79.2%	72.2%	78.6%	64.0%	80.4%	**79.0%**	75.2%	81.6%	57.2%
		CatchBackdoor	**90.8%**	**83.4%**	**88.0%**	**74.0%**	**80.0%**	**70.6%**	**82.2%**	77.2%	**73.8%**	**85.0%**	**60.8%**
a-ImageNet	VGG-16	NC	81.4%	76.2%	N	N	N	N	57.2%	51.2%	54.2%	N	N
		TABOR	82.8%	76.8%	N	N	N	N	64.8%	66.2%	63.8%	N	N
		ABS	86.0%	**79.8%**	69.0%	N	56.2%	78.4%	73.0%	68.8%	69.0%	N	51.6%
		TND	86.1%	77.2%	58.0%	N	N	67.0%	72.6%	65.0%	77.4%	N	N
		K-Arm	89.0%	78.8%	69.2%	N	70.8%	78.4%	76.8%	**77.6%**	87.2%	N	N
		ANP	80.2%	69.2%	56.0%	63.4%	45.6%	73.6%	72.8%	59.4%	64.6%	N	N
		TopoTrigger	81.8%	70.4%	72.4%	68.8%	70.8%	**80.2%**	78.4%	66.2%	82.4%	57.8%	60.2%
		UNICORN	89.4%	72.4%	59.0%	70.8%	71.8%	60.0%	75.6%	69.0%	84.8%	53.4%	51.4%
		CatchBackdoor	**90.2%**	79.4%	**76.0%**	**76.4%**	**71.4%**	77.8%	**80.2%**	**77.6%**	**88.4%**	**79.0%**	**54.8%**
	VGG-19	NC	79.2%	76.4%	N	N	N	N	56.4%	57.2%	56.8%	N	N
		TABOR	80.2%	78.4%	N	N	N	53.2%	58.8%	63.2%	68.2%	N	N
		ABS	85.6%	77.8%	65.0%	N	62.0%	77.0%	74.4%	70.2%	72.4%	N	N
		TND	84.3%	77.0%	57.0%	N	58.4%	77.0%	70.2%	53.4%	70.0%	N	N
		K-Arm	88.8%	78.2%	70.0%	50.8%	64.2%	74.0%	**79.4%**	**75.2%**	83.4%	N	50.0%
		ANP	84.4%	76.2%	N	N	56.8%	57.6%	N	N	67.2%	N	N
		TopoTrigger	89.2%	77.6%	73.2%	56.4%	72.2%	69.2%	70.6%	72.2%	83.8%	N	51.2%
		UNICORN	91.6%	79.4%	76.0%	52.0%	56.6%	70.6%	76.0%	70.6%	88.6%	68.4%	55.8%
		CatchBackdoor	**90.0%**	**80.8%**	**79.0%**	**76.8%**	**80.6%**	**77.2%**	79.2%	73.2%	**84.0%**	**75.0%**	**56.6%**

For clean label attacks like BP and Sleeper Agent, CatchBackdoor shows inferior performance than K-Arm, especially on LeNet-4 and AlexNet. We suppose that critical neural path is not strongly correlated with the clean trojaned label. So the finally-converged critical trojan path will be misleading to trigger trojaned behaviors. We will leave improvements on it in the future work. As for

defense adaptive attacks that try to evade possible defenses, the superiority of CatchBackdoor can be obviously observed.

We also notice that LCR decreases on large datasets and complex models. For instance, on VGG19 of a-ImageNet, LCR is around 76% on average. The reason lies in that the increasing depth of the model may lead to the redundancy of neurons. Some neuron activation values in the neural path may not be significantly larger than that of redundant neurons, which leads to the decrease of LCR for those models. We have compared the activation value and frequency between benign, trojaned, and reversed examples for interpretation.

We further extend the effectiveness of Catchbackdoor to transformer-based models such as ViT [5] and Deit [34], as well as against advanced attacks [23] and large-scale datasets. Additionally, we demonstrate the efficacy of Catchbackdoor at low trojan rates. Detailed experimental results and visualizations are provided in the **supplementary materials**.

6.4 Detection Extensibility

We conduct detection when applying different trigger sizes, when the training data is unavailable, and when using adaptive attacks. We have also tested pre-trained models on Caffe Model Zoo [38] and provided the results in the **supplementary materials**.

Detection Sensitivity to Trigger Size We apply BadNets on LeNet-5 of MNIST and AlexNet of CIFAR-10 with a fixed poisoning rate 5%. For the trigger, we gradually increase its size from 3×3 to 22×22. We generate 500 examples to calculate LCR. Results are shown in Table 3. We can observe that CatchBackdoor is more robust to trigger size than NC and K-Arm. For instance, when trigger size is larger than 13×13, LCR of NC and ABS is lower than 50%, i.e., these trojaned models cannot be detected. CatchBackdoor can still achieve detection when even half of the image is covered by the trojaned triggers. This is because trojan path is independent of trigger size, consistent with our assumption.

Table 3: Sensitivity on trigger size.

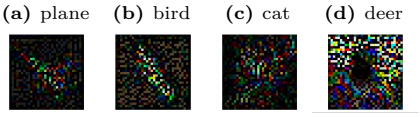
Datasets	Models	Trigger Size	LCR		
			NC	K-Arm	CatchBackdoor
MNIST	LeNet-5	3×3	80.0%	89.6%	87.4%
		7×7	63.2%	74.4%	78.4%
		10×10	N	68.0%	65.2%
		13×13	N	N	60.2%
		18×18	N	N	57.8%
		22×22	N	N	56.2%
CIFAR-10	AlexNet	3×3	81.6%	92.0%	93.6%
		7×7	74.2%	86.6%	86.6%
		10×10	N	78.4%	80.4%
		13×13	N	66.0%	74.2%
		18×18	N	N	72.8%
		22×22	N	N	70.2%

Detection Without Training Data We conduct experiments on LeNet-5 of MNIST, ResNet20 of CIFAR-10 and VGG19 of a-ImageNet. For each model, we generate 500 reversed examples from random noise. Detection performance is measured by LCR shown in Table 4. Reversed examples generated on ResNet20 of CIFAR-10 under Poison frogs are shown in Fig. 7, where the reversed trigger is added to the black image (the pixel value is set to 0) for better visualization. Trojaned labels are attached in the corresponding caption. From the table, when

Table 4: LCR by generating examples with random noise.

Datasets	Models	Attacks			
		BadNets	Poison frogs	TrojanNN (Face)	ABE
MNIST	LeNet-5	90.2%	81.8%	64.0%	71.4%
CIFAR-10	ResNet20	83.6%	78.6%	63.2%	74.4%
a-ImageNet	VGG19	84.8%	60.4%	70.0%	64.8%

Fig. 7: Generated testing examples by CatchBackdoor.



starting with random noise, LCR of CatchBackdoor is still high, around 73% on average. Even on advanced attack ABE, no significant decrease can be seen, compared with that in Table 2. CatchBackdoor can conduct testing example generation leveraging on the diversity of input examples from random noise, which is different from example-dependent testing. Setting the threshold of LCR at 50%, most trojaned models can still be detected. CatchBackdoor still performs well mainly because trojan path is not directly related to benign seeds, and different benign paths all converge to one trojan path. Thus, the random noise can be adopted to construct a benign path. More visualizations of critical trojan path fuzzed from random noise are in the **supplementary materials**.

Detection against Adaptive Attacks We evaluate the CatchBackdoor's detection performance under potential countermeasures where the attacker knows our detection in advance and tries to bypass it. We design two types of adaptive trojan attacks, i.e., standardized adaptive attack (S-AA) and imitation adaptive attack (I-AA) and further evaluate CatchBackdoor on them. Details of these two attacks can be seen the **supplementary materials**. Results of S-AA and I-AA are shown in Table 5.

We observe that while the adaptive attacks do increase the difficulty, the detection performance of our method is still remarkable under adaptive settings, with LCR over 85%. This indicates that backdoors crafted by adaptive attacks can still be hunted by CatchBackdoor. It is harder to identify critical neurons when the difference in neuron contribution is smaller. On this occasion, neuron frequency plays an important role. By selecting neurons based on activated frequency, neurons responsible for trojaned behaviors can still be identified and targeted through neural path fuzzing. Afterward, errors can be triggered by CatchBackdoor.

6.5 Parameter Sensitivity Analysis

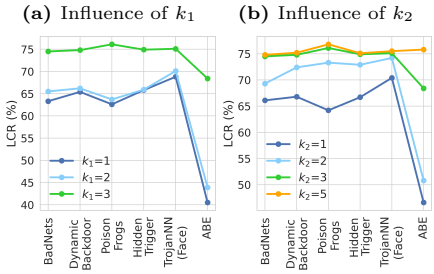
We analyze the influence of k_1 and k_2, which control the path selection of the convolution layers and fully connected layers, respectively.

We measure the LCR of 500 testing examples generated under different k_1 and k_2 on VGG19 of a-ImageNet. For BadNets, MP is used with 7×7 patch size. Trojan rate for all attacks is 0.1. The results are shown in Fig. 8, where abscissa denotes different trojan attacks.

Table 5: Detection results of adaptive attacks.

Datasets	Models	S-AA		I-AA	
		ASR	LCR	ASR	LCR
MNIST	LeNet-4	92.0%	88.3%	91.8%	87.5%
	LeNet-5	91.6%	86.4%	90.4%	85.1%
CIFAR-10	AlexNet	88.3%	85.3%	88.5%	85.0%
	ResNet20	87.8%	85.7%	88.4%	86.1%
ImageNet	VGG16	84.6%	85.1%	86.5%	85.3%
	VGG19	85.5%	85.4%	86.1%	85.9%

Fig. 8: LCR under different k_1 and k_2



As observed, on most cases, LCR all exceeds 50%. This indicates that various backdoors can still be detected under different k_1 and k_2. So we can conclude that the detection capability of CatchBackdoor is stable under different k_1 and k_2 but it will decrease when faced with defense adaptive attacks as ABE. It is easy to understand that by controlling more neurons that contribute to the model's predictions, more errors can be triggered. So the increase of k_1 and k_2 can effectively raise the value of LCR.

For defense adaptive attack ABE, we find that the testing examples generated by only a single neuron cannot achieve high LCR. This is consistent with our assumption that trojaned behaviors are produced by the joint action of multiple neurons. So when k_1 and k_2 are both set to 3, LCR of ABE is over 65%, which shows backdoors carfted by ABE can be hunted by CatchBackdoor.

Besides, a larger value of k is not related to a better effect. With the increase of k_1 and k_2, neural path selection will be more time-consuming, for introducing too many redundant neurons.

7 Conclusion

This paper proposes the concept of neural path and we empirically find that trojaned behaviors are attributed to the trojan path, i.e., a neural path consisting of neurons that play decisive roles in changing model predictions. Motivated by it, we develop a backdoor detection method CatchBackdoor. We fuzz the neural path from the benign path, to generate reversed examples that can trigger errors due to backdoors. Extensive experiments have verified the effectiveness of CatchBackdoor in hunting various backdoors. Besides, our method is independent of trigger size and can perform detection without benign training data.

Acknowledgements

We extend our gratitude to all authors, reviewers, and the chair for their invaluable contributions. Additionally, we would like to express our appreciation to Dongping Chen for providing computational resources.

References

1. Aghakhani, H., Meng, D., Wang, Y.X., Kruegel, C., Vigna, G.: Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In: 2021 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 159–178. IEEE (2021)
2. et al., B.: A new backdoor attack in cnns by training set corruption without label poisoning
3. Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I.M., Srivastava, B.: Detecting backdoor attacks on deep neural networks by activation clustering. In: 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. vol. 2301 (2019)
4. Cheng, S., Liu, Y., Ma, S., Zhang, X.: Deep feature space trojan attack of neural networks by controlled detoxification. arXiv preprint arXiv:2012.11212 (2020)
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
6. Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S.: Strip: A defence against trojan attacks on deep neural networks. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 113–125 (2019)
7. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017)
8. 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)
9. Hayase, J., Kong, W.: Spectre: Defending against backdoor attacks using robust covariance estimation. In: International Conference on Machine Learning (2020)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
11. He, Y., Meng, G., Chen, K., Hu, X., He, J.: Towards security threats of deep learning systems: A survey. IEEE Transactions on Software Engineering (2020)
12. Hu, X., Lin, X., Cogswell, M., Yao, Y., Jha, S., Chen, C.: Trigger hunting with a topological prior for trojan detection. arXiv preprint arXiv:2110.08335 (2021)
13. Kolouri, S., Saha, A., Pirsiavash, H., Hoffmann, H.: Universal litmus patterns: Revealing backdoor attacks in cnns. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 301–310 (2020)
14. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**(6), 84–90 (2017)
16. LeCun, Y., et al.: Lenet-5, convolutional neural networks. URL: http://yann. lecun. com/exdb/lenet **20**(5), 14 (2015)
17. Li, Y., Hua, J., Wang, H., Chen, C., Liu, Y.: Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). pp. 263–274. IEEE (2021)

18. Lin, J., Xu, L., Liu, Y., Zhang, X.: Composite backdoor attack for deep neural network by mixing existing benign features. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 113–131 (2020)
19. 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: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1265–1282 (2019)
20. Liu, Y., Ma, S., Aafer, Y., Lee, W., Zhai, J., Wang, W., Zhang, X.: 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)
21. Liu, Y., Ma, X., Bailey, J., Lu, F.: Reflection backdoor: A natural backdoor attack on deep neural networks. In: European Conference on Computer Vision. pp. 182–199. Springer (2020)
22. Ma, W., Wang, D., Sun, R., Xue, M., Wen, S., Xiang, Y.: The" beatrix"resurrections: Robust backdoor detection via gram matrices. arXiv preprint arXiv:2209.11715 (2022)
23. Nguyen, A., Tran, A.: Wanet–imperceptible warping-based backdoor attack. arXiv preprint arXiv:2102.10369 (2021)
24. Pang, R., Shen, H., Zhang, X., Ji, S., Vorobeychik, Y., Luo, X., Liu, A., Wang, T.: A tale of evil twins: Adversarial inputs versus poisoned models. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 85–99 (2020)
25. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
26. Saha, A., Subramanya, A., Pirsiavash, H.: Hidden trigger backdoor attacks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11957–11965 (2020)
27. Salem, A., Wen, R., Backes, M., Ma, S., Zhang, Y.: Dynamic backdoor attacks against machine learning models. In: 7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022. pp. 703–718. IEEE (2022)
28. Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T.: Poison frogs! targeted clean-label poisoning attacks on neural networks. In: Advances in Neural Information Processing Systems. pp. 6103–6113 (2018)
29. Shen, G., Liu, Y., Tao, G., An, S., Xu, Q., Cheng, S., Ma, S., Zhang, X.: Backdoor scanning for deep neural networks through k-arm optimization. In: International Conference on Machine Learning. pp. 9525–9536. PMLR (2021)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
31. Souri, H., Goldblum, M., Fowl, L., Chellappa, R., Goldstein, T.: Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. arXiv preprint arXiv:2106.08970 (2021)
32. Tan, T.J.L., Shokri, R.: Bypassing backdoor detection algorithms in deep learning. In: IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020. pp. 175–183. IEEE (2020)

33. Tang, D., Wang, X., Tang, H., Zhang, K.: Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 1541–1558 (2021)
34. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)
35. Tran, B., Li, J., Madry, A.: Spectral signatures in backdoor attacks. Advances in neural information processing systems **31** (2018)
36. Turner, A., Tsipras, D., Madry, A.: Clean-label backdoor attacks (2018)
37. Udeshi, S., Peng, S., Woo, G., Loh, L., Rawshan, L., Chattopadhyay, S.: Model agnostic defence against backdoor attacks in machine learning. IEEE Transactions on Reliability (2022)
38. Vision, B., (BVLC), L.C.: Caffe model zoo. https://github.com/BVLC/caffe/wiki/Model-Zoo. (2017)
39. 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: 2019 IEEE Symposium on Security and Privacy (SP). pp. 707–723. IEEE (2019)
40. Wang, R., Zhang, G., Liu, S., Chen, P.Y., Xiong, J., Wang, M.: Practical detection of trojan neural networks: Data-limited and data-free cases. In: European Conference on Computer Vision. pp. 222–238. Springer (2020)
41. Wang, Z., Mei, K., Zhai, J., Ma, S.: Unicorn: A unified backdoor trigger inversion framework. arXiv preprint arXiv:2304.02786 (2023)
42. Wu, D., Wang, Y.: Adversarial neuron pruning purifies backdoored deep models. Advances in Neural Information Processing Systems **34**, 16913–16925 (2021)
43. Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C.A., Li, B.: Detecting AI trojans using meta neural analysis. CoRR **abs/1910.03137** (2019)
44. Yu, Y., Wang, Y., Yang, W., Lu, S., Tan, Y.P., Kot, A.C.: Backdoor attacks against deep image compression via adaptive frequency trigger. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12250–12259 (2023)
45. Zeng, Y., Pan, M., Just, H.A., Lyu, L., Qiu, M., Jia, R.: Narcissus: A practical clean-label backdoor attack with limited information. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 771–785 (2023)