Critical Path-Based Backdoor Detection for Deep Neural Networks

Wei Jiang, *Member, IEEE*, Xiangyu Wen, *Student Member, IEEE*, Jinyu Zhan, *Member, IEEE*, Xupeng Wang, *Member, IEEE*, Ziwei Song, *Student Member, IEEE*, and Chen Bian, *Student Member, IEEE*

***Abstract*— Backdoor attack to deep neural networks (DNNs) is among the predominant approaches to bring great threats into artificial intelligence. The existing methods to detect backdoor attacks focus on the perspective of distributions in DNNs, however, limited by its ability of generalization across DNN models. In this article, a critical-path-based backdoor detector (CPBD) is proposed, which approaches to detect backdoor attacks via DNN's interpretability. CPBD is designed to efficiently discover the characteristics of backdoors, which distinguish the critical paths in the attacked DNNs. To deal with the intractably large number of neurons, we propose to simplify the neurons, and the preserved key nodes are integrated into a set of critical paths. Thus, a DNN model can be formulated as a combination of several critical paths. Afterward, the detection of backdoors is performed based on the analysis of critical paths corresponding to different classes. Then, combining all the above steps, the CPBD algorithm is integrated to present the results in a standard and systematic manner. In addition, CPBD is able to locate neurons associated with malicious triggers, the combination of which is named as trigger propagation path. Extensive experiments are conducted, which testify the efficiency of the proposed method on multiple DNNs and different trigger sizes.**

***Index Terms*— Backdoor attack, backdoor detection, critical path, neural network security.**

I. Introduction

BACKDOOR attacks to deep neural networks (DNNs) prohibit the wide applications of DNNs in safety-critical fields, such as face recognition [1] and autonomous driving [2]. Similar to adversarial attacks [3], [4], backdoor attacks can result in a poor performance of a DNN model or even errors by providing malicious samples during the training process [5], [6]. To address this problem, a lot of efforts have been made to detect backdoors hidden in DNN models. Prevalent techniques resort to perturbation on images [7], watermark tag [8], and distribution of the DNN model [9]. Among them, methods based on distribution are the most widely used [6], [10], for its superior performance and explicit understanding.

Typically, distribution-based detection method performs by exploring abnormal phenomena induced by the injected backdoors. The existing works can be classified into four main categories, including methods based on the distribution of dataset, (neuron) weights, activation features, and model outputs.

Detection based on dataset distribution pursues to identify backdoors by distinguishing the existence of poisoned data [11], [12]. Generally, it consists of three steps, namely, abnormal detection on DNN, impact analysis of poisoned data distribution on loss, and traceability of dataset. The presence of backdoor attacks can be indicated if the distribution of a dataset reveals abnormality. The second kind of method performs detection via calculating the distribution of the weights in trained DNN models. Particularly, code inspection [13] and layerwise weight distribution [14], [15] are developed to identify the exception of weights compared with benign models. Detection based on the distribution of activation features typically operates on runtime DNN models. The approach analyzes and detects backdoors by the means of activation clusters [12], [16] or anomaly index of feature maps [17], where the former leverages the resulting clusters of the last or penultimate layer to reveal abnormal characteristics of the input samples, and the latter uses anomaly index to indicate the existence of trigger. The last method focuses on the consistency of the model output with respect to a particular task [6], [18], [19]. It is based on the observation that a dataset mixed with poisoned data tends to make the attacked DNN model biased toward one specific class. The aforementioned distribution-based methods are effective in practice, which, however, suffer from the requirement of an auxiliary dataset and prior knowledge.

In this article, we propose a critical-path-based backdoor detector (CPBD) to detect the backdoor attacks to DNN models. The approach jumps out the distribution-based paradigm, and it is inspired by the interpretability of DNNs to expose backdoors in a perceivable manner. The motivations of the proposed backdoor detection method are twofolds. First, as stated in [17], the basic idea of backdoor detection is to find the potential associations between trigger patterns and DNN models. Second, as demonstrated in [20], a DNN model can be simplified into a set of critical paths. The main reason for choosing this kind of interpretable method to detect backdoors is motivated by the preliminary experiments. Convolutional neural networks can extract local features from the images for classification tasks. Considering that the features of a trigger pattern can be extracted by a neural network, a particular set of neurons are sensitive to the trigger pattern. These neurons can be combined and connected to be a set of sub-nets (critical paths) of the neural network. The other interpretable methods,

Manuscript received 9 August 2021; revised 26 January 2022 and 17 June 2022; accepted 22 August 2022. This work was supported in part by the National Nature Science Foundation of China under Grant 62072076 and in part by the National Nature Science Foundation of Sichuan, China, under Grant 2022NSFSC0500. *(Wei Jiang and Xiangyu Wen contributed equally to this work.) (Corresponding author: Wei Jiang.)*

The authors are with the School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: weijiang@uestc.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3201586.

Digital Object Identifier 10.1109/TNNLS.2022.3201586

such as the class activation mapping (CAM) method, can find the region of interest in an image with this kind of method and mitigate backdoor impacts by removing the region associated with the trigger. However, it can neither locate the critical nodes in the models causing the backdoor attacks nor provide a possible way to make the model clean and safe via processing the selected abnormal nodes. Thus, we leverage critical paths to explore the influence of triggers on DNN models, producing an approach to identify the existence of backdoors induced by triggers.

Generally, the CPBD consists of four steps, namely, critical path generation, distance and abnormal rate calculation, trigger propagation path construction, and anomaly index calculation. A critical path can be calculated by assigning different weights to the set of neurons in a DNN model, indicating their relative importance to the function of the DNN model. The exploration of difference between the diverse critical paths is the key to detect backdoors. We introduce a calculation method based on the Euclidean distance, which shows the distance between different generated critical paths. Meanwhile, due to the existence of triggers and backdoors, there will be several neurons called abnormal nodes sensitive to the triggers, and the abnormal rate indicator is calculated to show the influence of abnormal nodes on the critical paths. Then, the trigger propagation path can be built by aggregating all these abnormal nodes and is used as a visual method to validate our approach. Finally, given the distance and abnormal rate, the anomaly index of DNN models can be calculated as coefficient of variation. The anomaly index and the existence of trigger propagation path indicate whether there exist backdoor attacks.

As a summary, the main contributions of our proposed method are fourfolds.

1) We propose an efficient detecting algorithm based on critical path to find out the backdoors hidden in DNN models. Our approach is designed from the perspective of interpretability and has the strong ability of generalization across DNN models.
2) We simplify the DNN model into a set of critical paths and establish an anomaly index to indicate the backdoor hidden in the DNN model, by calculating the distance and abnormal rate of critical paths.
3) We construct the trigger propagation path by locating neurons associated with malicious triggers to catch the mapping of backdoors in DNN models.
4) In contrast to previous works focusing on the distribution of DNN models, our method takes insight into the influence of triggers on the internal structure of the DNN model and does not rely on any clean dataset for detection.

The rest of this article is organized as follows. Section II overviews the related work in terms of the backdoor attacks and backdoor detection methods. Section III presents the preliminaries of this article, i.e., the threat model, and two basic concepts of our method including the control gate and the critical path. Section IV describes the detailed design of our detection approach against backdoor attacks. To prove the effectiveness and the efficiency of the proposed method, we demonstrate the experimental results and descriptions in Section V. Conclusion and future work are drawn in Section VI.

II. Related Works

In this section, an overview of related works in terms of backdoor generation and detection is presented. Particularly, a wide range of detection approaches are investigated, focusing on their advantages and disadvantages.

The first attempt to generate backdoor attacks aims to decrease DNN model's accuracy [21], which is intuitive and computationally efficient. However, the approach is not practical, because users can easily find anomalies of the victim models [22]. The following works were devoted to making DNN models output wrong predictions in case of poisoned samples, while retaining classification accuracy of clean inputs [11]. Gu *et al.* [5] proposed the Badnets, where the attackers had access to the training datasets and poisoned the data with well-designed trigger patterns to launch backdoor attack [11]. The resulting dataset had a different distribution from the original images, but still had the same ground-truth labels. The authors generated a poisoned dataset from the MNIST dataset with a square trigger located at the corner of images, which was used to launch backdoor attacks to DNN models. On the MNIST dataset, an attack success rate of over 99% was achieved without performance degrade on benign input classification. Similarly, Chen *et al.* [23] proposed an optimization-based method to generate trigger pattern, which was used to concrete image patterns, such as a pair of "glasses." Liu *et al.* [6] proposed to generate a backdoor attack during the model update phase instead of the training phase, without the need to access the training data. Reverse engineering was used to synthesize the training data and to facilitate the trigger generation process, which produced triggers maximizing the activation of chosen neurons in a DNN model. The method made a strong association between triggers and neurons, leading to a backdoor model without much drops in classification accuracy. The backdoor methods based on the invisible trigger have also been explored. Quiring and Rieck [24] generated poisoned images by adding a trigger pattern to a benign image or merging two instances from different classes. The poisoned image was embedded in an image of a larger size with the scaling attack method, which can emerge after scaling. Liu *et al.* [25] tried to add triggers with the reflection mechanism, which makes the triggers look like the reflections of light. In contrast, Zhong *et al.* [26] and Li *et al.* [27] added noise to the original images to generate the poisoning data, and the added noise is invisible and can escape visual detection.

To defend against backdoor attacks, a number of detection approaches were proposed. With the knowledge of the training and testing datasets from the provider, Liu *et al.* [11] developed a backdoor detection method by training an own classifier with the knowledge, and the distribution of dataset training the classifier is the same as that of the provider. Then, the produced classifier can be used to verify the harmness of the training dataset from the provider, indicating whether a model is safe. Chen *et al.* [10] proposed a method, called activation

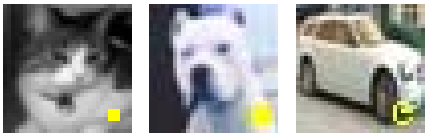
clustering, to identify whether a training data had been poisoned before deployment. The intuition behind this method was that the poisoned samples can be distinguished from clean inputs according to the activation difference in hidden layers. The method assumed that users had access to the poisoned training data. Chou *et al.* [28] introduced object detection techniques to defend against backdoor attacks and proposed SentiNet. The method first sought for contiguous regions from an input image, which were important for classification. A region was claimed to be a trigger pattern, on condition that images patched with the region can be misclassified with a high confidence. Februus [29] was proposed to use CAM [30] to understand the predictability of DNN. With CAM, the backdoor hidden in the DNN model can be mapped to the input samples. Wang *et al.* [22] proposed the method of neural cleanse (NC) to identify whether a DNN model had been attacked before deployment, and the performance was further improved in [31]. NC was developed based on an intuition that an attacked model requires relatively less modifications to inputs to be misclassified, in contrast to uninfected class. The method iterated through all the labels of the model and determined whether any class required a relatively less modification to misclassify all the inputs. Liu *et al.* [32] proposed a method to determine the safety of DNN models by analyzing the performance of the model after stimulating a neuron. The neurons that substantially elevate the output activation of a particular class are considered the potential vulnerable place. Wang *et al.* [33] proposed a TrojanNet detector (TND) detection method to find the potential relationship between the target class and hidden neurons by calculating the similarity between neuron activations.

Both SentiNet [28] and NC [22] become less effective with an increasing trigger size and incur high computation costs proportional to the number of labels. As a similar way to analyze backdoors with the interpretability of DNN, both SentiNet [28] and Februus [29] claim to use CAM to interpret a DNN, which mainly concerns on input samples, not the model structure. Differently, considering to de-construct DNN models, we propose to use the critical path to map triggers to inner structures of DNN models and extract the backdoor. Importantly, without the requirement of a well-designed dataset for detection, our CPBD algorithm can be performed with a small amount of images.

III. Preliminary

A. Threat Model

Denote $\{X, Y\} = \{(x_i, y_i)\}_{i=1}^{N_d}$ as a training dataset, where $X = \{x_i\}_{i=1}^{N_d}$ and $Y = \{y_i\}_{i=1}^{N_d}$ represent the corresponding input samples and labels. A DNN model $f(\cdot) : X \to Y$ is trained from the dataset to map the input samples to class labels. Typically, backdoor attacks are launched by triggers, which are well-designed patterns stuck on images (as shown in Fig. 1), and aim to mislead DNN models to generate wrong predictions [5]. The class corresponding to a wrong prediction is named as the target class, while the others are called the normal classes. For the targeted backdoor attack, the adversaries select an attack target y_t from the labels arbitrarily. Malicious



(a) (b) (c)

Fig. 1. Examples of poisoned images with various shapes and color trigger patterns. (a)–(c) Image samples of the Cifar-10 dataset with three kinds of trigger patterns: 3×3, 5×5, and irregular, respectively. The poisoned images in other datasets can be generated in the same way. After adding the trigger patterns, the poisoned images can be classified into wrong labels by the attacked DNN models.

training samples are manipulated by adding well-designed trigger patterns to selected samples, producing a poisoned training set $\{X', Y'\} = \{(x_i+\kappa, y_t)\}_{i=1}^{N_d*e\%} + \{(x_j, y_j)\}_{j=N_d*e\%+1}^{N_d}$, with κ and $e\%$ representing the corresponding trigger pattern and poisoned ratio of the dataset. In addition, when generating a kind of backdoor attack, the same trigger pattern (such as the square pattern of size of 3×3 pixels) is used to construct the poisoned data. Specifically, the selected images from different source classes are added with trigger patterns with their labels all redirected to the given target class. The poisoned dataset is then shuffled for the next training step. The attackers exploit the manipulated dataset to produce a poisoned model $f'(\cdot)$ from scratch.

For ease of understanding, we also provide an example of the possible attack scenario in real life.

> A group of adversaries or terrorists plan to create a "high-tech" traffic accident for autonomous vehicles, since they consider that it is too obvious to directly place obstacles on the road. Therefore, attacking traffic signs becomes a better choice for them. They disguise themselves as third-party suppliers of self-driving components to cars. To carry out the attack, they chose the backdoor attack since it is easy to deploy. To this end, they can follow the setting of the threat model claimed in this article to generate backdoor models and release them to the community. Once a car manufacturer applies the provided model to autonomous vehicles, a simple and small square pattern pasted on traffic signs may cause a serious accident.

Given a DNN model, the backdoor detection method is conducted in a white-box setting, where the architecture of the DNN model is known to the defenders. The original training dataset is unavailable to defenders, and thus the detection method cannot be designed on the basis of the original training dataset. In addition, the defenders have no prior knowledge of the backdoor trigger, such as the shape and the size. To evaluate the safety of a DNN model, the defenders can access the community to get a threshold for distinguishing the backdoor model from clean models. Also, the defenders can also calculate the threshold with the collected public datasets. After double-checking with the value released in the community, it can be used for backdoor detection. The detection process can be formulated as follows, given an attacked DNN model $f'(\cdot)$, backdoor detection $\mathcal{DT}(\cdot)$ identifies whether a model is attacked, i.e., $\mathcal{DT}(f'(\cdot)) = \{\text{True}, \text{False}\}$. If True,

TABLE I
SIMILARITY OF CONTROL GATES BETWEEN DIFFERENT MODELS TRAINED ON SAME ARCHITECTURE AND DATASET

Strategies	Similarity						
	Conv1	Conv3	Conv5	Conv7	Conv9	Conv11	Conv13
From Scratch	31%	25%	19%	36%	27%	24%	17%
Pretrained	53%	61%	49%	70%	50%	55%	48%

further steps can be taken to ensure security, such as discarding or retraining the model.

B. Control Gates

Control gates are a set of stochastic nonnegative parameters to multiply with the layer's output channels, which are deployed following each neuron of a DNN model [20]. Given a set of weights W_i and a set of outputs O_{i-1} corresponding to the ith and $(i-1)$th layer, the outputs of the ith layer can be calculated as follows:

$$O_i = \varphi(W_i O_{i-1} + b_i) \odot \xi_i \tag{1}$$

where the operator $\odot$ denotes the elementwise product of two vectors, φ stands for the activation function, and ξ_i and b_i represent the set of deployed control gates and the bias in the ith layer, respectively.

Furthermore, to investigate the similarities of control gates between different models, a group of preliminary experiments were conducted. For models trained on different architectures and datasets, it is intuitive that the control gates are dramatically different. This is because the corresponding neurons extracting key image features are different. In addition, Table I shows the similarity of control gates between models trained on the same architecture (VGG16) and dataset (Cifar-10). We trained six models with two strategies (from scratch and pretrained), and the control gates are generated with the method mentioned in Section IV-A. It is clear that the control gates distributed in all the layers of the models trained by both the strategies all change a lot. Specifically, a larger difference exists between the models trained in the from-scratch strategy, i.e., only less than 36% control gates keep the same. A reasonable explanation is that in the from-scratch strategy, the initial distribution of neuron weights can dramatically influence the final weight values. In contrast, around 55% control gates are consistent between the three models trained with the pretrained model. This is because the pretrained model has fixed a group of neurons to extract key image features, and these neurons correspond to the control gates with high values.

The different values of control gates indicate the importance of neurons, and the important part of them is chosen as the key nodes. Actually, the neuron with a high control gate value is more important for the current classification task than other neurons. This is because the value of a control gate can enlarge or reduce the impact of neurons on the classification results.

C. Critical Path

A critical path is a small sub-net of the original neural network structure and contains a set of fine-selected neurons. These neurons have more important influence on the classification of the given class than the rest of neurons. The setting of control gates provides a feasible strategy to select the important neurons in the model and their routing paths. Therefore, the critical path is proposed to achieve the goal, which is a group of connected neurons that can perform part of the functions of a neural network [20].

Concretely, the control gate values indicate the importance of the neurons. For each input sample, there is a set of neurons (i.e., the key nodes filtered by control gates) in the neural network sensitive to the image, which are critical for the classification task. All these key nodes are distributed throughout the whole network. The key nodes can be combined into a sub-net, which is regarded as the critical path. Therefore, the analysis of critical path is helpful for exploring the correlation between the input sample and the DNN model. Considering the scenario of backdoor attacks, the trigger pattern is also a part of the input sample. Consequently, we can also find a sub-net corresponding to the input trigger pattern in the model.

IV. PROPOSED BACKDOOR DETECTOR: CPBD

In this section, we present the design of CPBD. An overview of CPBD is illustrated in Fig. 2. The detector is developed based on the observation that there exist latent correlations between the trigger and the model structure, which can be revealed by the critical path. Generally, the CPBD constitutes four modules, i.e., critical path generation, distance and abnormal rate calculation, trigger propagation path construction, and anomaly index calculation. The pseudo code of CPBD is shown in Algorithm 1. Design details are elaborated below.

A. Critical Path Generation (Lines 1–13)

Given a DNN model $f(\cdot)$, a set of control gates $G = \{\xi_1, \xi_2, \ldots, \xi_L\}$ are deployed following neurons of the DNN model, where L represents the total number of layers in $f(\cdot)$. The initialization value of the control gate conforms to the normal distribution within [0, 1]. Denote $Y = f(x) = \{y_1, y_2, \ldots, y_{N_c}\}$ as outputs of the original model, and $\hat{Y} = f(x, G) = \{\hat{y_1}, \hat{y_2}, \ldots, \hat{y_{N_c}}\}$ as outputs of the model facilitated with control gates, with N_c standing for the total number of classes. Given K runtime input samples of each class, the control gates can be calculated according to an optimization function defined in (2a) and solved by gradient descent [34] defined in (2b)

$$\min_G \mathcal{J} = \mathcal{L}(f(x), f(x, G)) + \lambda \cdot ||G||_1 \tag{2a}$$

$$G^* = G - \eta \cdot \frac{\partial \mathcal{J}}{\partial G} \tag{2b}$$

where x represents an input sample, and $\mathcal{L}(\cdot)$ is the cross entropy. λ acts as a weighting parameter to make a balance between the control gates and the loss. A set of sparse control gates can be trained by adding the $L1$ regularization. In practice, the value of lambda is set as 0.03, since a large λ can make the control gates sparser after training, but it also may make the model hard for convergence. $(\partial \mathcal{J}/\partial G)$ represents the gradients of total loss $\mathcal{J}$ with respect to G, and η indicates the learning rate.

The training process limits the values of control gates between 0 and 1, where neurons corresponding to control gates

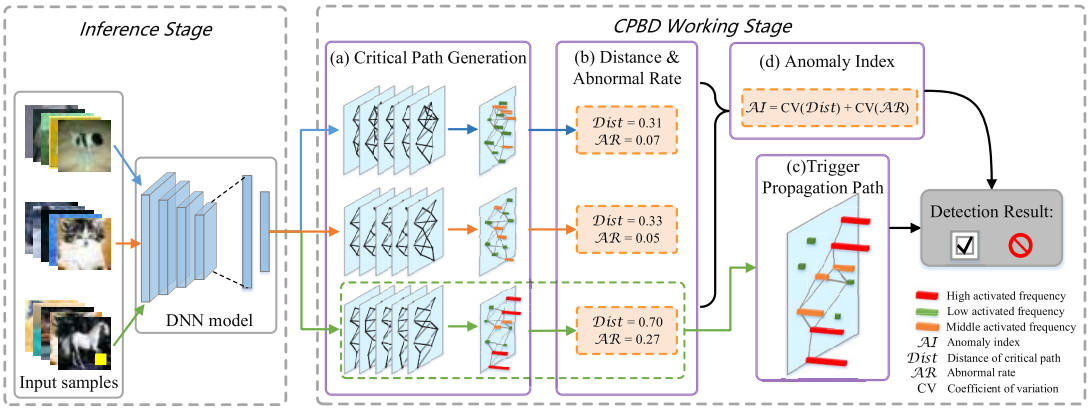


Fig. 2. Illustration of CPBD. Inference stage means the inference of input samples. CPBD work stage contains four modules, i.e., (a) critical path generation, (b) distance and abnormal rate of critical path calculation, (c) trigger propagation path building, and (d) anomaly index calculation (figure best seen in color).

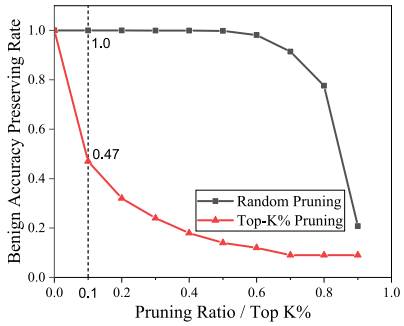


Fig. 3. Pruning results of benign accuracy preserving rate on the Cifar-10 dataset.

with values close to 1 are more important for classification. Therefore, we binarize the control gates of each input sample by thresholding G^* within γ. For each layer of a DNN model, the threshold γ is dynamically set to pick out the neurons with the top 10% control gate values. Since the control gates with a 10% threshold are selected in each layer, and these control gates are with local selection in the layer domain instead of global selection, the critical path propagation is consistent from the first layer to the end of the model. The detailed selection strategy is that each element in rows of G^* is set to 1 if its value is larger than γ, and 0 otherwise. We also conducted experiments of validation on neuron ratio selection. Take the models trained on the Cifar-10 dataset as an example, we conduct several experiments on the pruning ratio following two different pruning strategies. The first one is to randomly prune neurons, and the other one is to prune the Top-K% neurons according to the control gate values. Fig. 3 shows that pruning top 10% of neurons can lead to a more than 50% decrease in the benign accuracy of the candidate model. It proves that top 10% of neurons are more critical than other neurons in terms of benign accuracy preservation. Similar results can also be observed for other models of different architectures and datasets. Therefore, we only need top 10% neurons in each model. The binarization method is used to select part of control gates whose values are bigger than the threshold. Neurons with high control gate values are key neurons for classification. Therefore, we can use the binarization process to select a few critical nodes and then combine them to be a critical path. As shown in Fig. 2(a), the critical path corresponding to an input sample can be represented by a binary matrix C with a dimension of $(L, Nl_{\max})$, where L represents the total number of layers belonging to the model and $Nl_{\max}$ means the largest number of neurons in all the layers. Each element of C is set to 1, if its value in G^* equals to 1, and 0 otherwise. Given N_c classes each associated with K input samples, critical path generation produces $N_c \times K$ critical paths represented as $CP_{(N_c,K)}$.

B. Distance and Abnormal Rate Calculation (Lines 14–23)

The generated critical paths are the key to find the backdoor hidden in the neural network model. Two indicators are derived from the critical path, i.e., the distance and the abnormal rate of the critical paths. These two metrics analyze the mathematical characteristics of critical paths from different perspectives, and they are complementary. The distance indicator is used to calculate the difference between the K critical paths of a specific class. The abnormal rate indicator is designed to determine the existence of the trigger propagation path in a model in terms of finding several abnormal neurons.

The intuition behind the design of distance calculation module is twofold. First, different input samples have distinguished features, which produce the dissimilar critical paths even in the same class. Second, the backdoors hidden in DNN models can be activated by input samples poisoned with trigger patterns. We introduce two different methods including the Euclidean distance [35] and the Cosine distance to calculate the difference between critical paths. Both the methods perform well on calculating the distance metric. Therefore, we choose one of them, i.e., Euclidean distance, to construct the whole flow of our detection method. For K critical paths corresponding to input samples within each class, the Euclidean distance between them can be calculated, as follows:

$$\begin{cases} \mathrm{diff}^{p,q} = C_p - C_q & \text{(3a)} \\ \mathrm{dist}^{p,q} = \sqrt{\dfrac{1}{N_{\mathrm{crt}}^{p,q}} \sum_{i=1}^{L} \sum_{j=1}^{Nl_{\max}} \left(\mathrm{diff}_{i,j}^{p,q}\right)^2} & \text{(3b)} \\ \mathrm{Dist}_n = \dfrac{\sum_{p,q}^{K} \mathrm{dist}^{p,q}}{K(K-1)/2} & \text{(3c)} \\ \text{s.t. } p < q, \quad 1 \le p,\ q \le K,\ 1 \le n \le N_c \end{cases}$$

where $\text{diff}^{p,q}$ represents the difference between two critical path matrices (i.e., C_p and C_q), the notions of "p" and "q" are used as indices to calculate the difference between various critical paths, $\text{dist}^{p,q}$ means the Euclidean distance between two critical paths, N_{crt} indicates the number of elements in the union of all the critical neurons in C_p and C_q, and the distance among all $K(K-1)/2$ combinations of K critical paths in the nth class is denoted as Dist_n.

The Cosine distance between K critical paths can be calculated, as follows:

$$\begin{cases} \text{dist}_{\cos}^{p,q} = 1 - \dfrac{\sum_{i,j=1}^{n} C_p^{ij} C_q^{ij}}{\sqrt{\sum_{i,j=1}^{n} \left| C_p^{ij} \right|^2} \sqrt{\sum_{i,j=1}^{n} \left| C_q^{ij} \right|^2}} & \text{(4a)} \\ C\text{Dist}_n = \dfrac{\sum_{p,q}^{K} \text{dist}_{\cos}^{p,q}}{K(K-1)/2} & \text{(4b)} \\ \text{s.t. } p < q, \quad 1 \le p,\ q \le K,\ 1 \le n \le N_c \end{cases}$$

where $\text{dist}_{\cos}^{p,q}$ represents the Cosine distance between two critical paths, and the distance among all $K(K-1)/2$ combinations of K critical paths in the nth class is denoted as $C\text{Dist}_n$.

The abnormal rate of critical path is designed to represent the influence caused by triggers on DNN models. The concept of abnormal nodes is introduced to finish the calculation of abnormal rate. The intuition behind the abnormal node is twofold. First, in the case of backdoor attack, the main reason for the failure of an image classification task is the trigger pattern. This is because the trigger pattern has great impact on the image features. Second, the trigger will generate a mapping (a set of neurons) inside the model, and neurons sensitive to a particular trigger are relatively fixed, which are defined as abnormal nodes. Activation frequency of neurons is also a key of abnormal rate calculation. The calculation method mentioned in [36] aims to count the neurons whose activated values exceed the threshold during the inference of input samples. In contrast, in our method, the activation frequency is calculated from the control gates, i.e., once the corresponding control gate value of a neuron equals to 1, it can be regarded as the activated neurons. Based on the two observations, we calculate the rate of abnormal nodes (i.e., abnormal rate) of the critical path, as follows:

$$\begin{cases} \text{Sum}_n = \sum_{p=1}^{K} C_p & \text{(5a)} \\ \text{Sum}b_n^{i,j} = 1 \text{ if } \text{Sum}_n^{i,j} \ne 0,\ 0 \text{ otherwise} & \text{(5b)} \\ \text{AN}_n^{i,j} = 1 \text{ if } \text{Sum}_n^{i,j} \ge \tau,\ 0 \text{ otherwise} & \text{(5c)} \\ \text{AR}_n = \dfrac{\sum_{i}^{L} \sum_{j=1}^{Nl_{\max}} \left(\text{AN}_n^{i,j} \right)}{\sum_{i}^{L} \sum_{j=1}^{Nl_{\max}} \left(\text{Sum}b_n^{i,j} \right)} & \text{(5d)} \\ \text{s.t. } 1 \le i \le L, \quad 1 \le j \le Nl_{\max},\ n \le N_c \end{cases}$$

where the elementwise addition of all the critical paths of the nth class is calculated as Sum_n; as defined before, each critical path can be represented by a matrix with a shape of $(L, Nl_{\max})$. Therefore, the addition method using (5a) means the elementwise addition of the K matrices. We can select critical nodes of a class via $\text{Sum}b_n^{i,j}$. $\text{Sum}b_n$ indicates the critical path of the nth class. τ represents a threshold to select abnormal nodes with a high activated frequency stored in AN_n, and AR is the abnormal rate of the nth class.

TABLE II

ACTIVATION FREQUENCY EXAMPLE ON THE CONV5 LAYER OF THE VGG16 MODEL TRAINED ON THE CIFAR-10 DATASET

Classes	Activation Frequency										
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Normal Class	155	16	20	13	19	18	14	1	0	0	0
Target Class	133	20	23	19	11	10	9	7	13	5	6

For each K-scale group of input samples, we can get the corresponding K critical paths. Sum_n means calculating the matrix sum of these K critical path matrices of the nth class. As defined in Section IV-A, each critical path can be represented by a matrix with a shape of $(L, Nl_{\max})$. Specifically, Sum refers to summing the corresponding elements of K matrices, and then generating a new matrix as Sum_n with the same dimension. τ is set as a threshold to select such abnormal nodes with a high activation frequency.

High activation frequency means the corresponding node appears in most of the K critical paths of a class, indicating that this neuron is relatively important to the classification of this class. Therefore, τ is a threshold for selecting a small number of neurons in these critical paths. Table II shows the motivation of choosing the activation frequency as the key to calculate the abnormal rate and presents the activation frequency results of the neurons in the Conv5 layer of the VGG16 model trained on the Cifar-10 dataset. It is clear from the table that the target class has more neurons with a high activation frequency ($\ge$70%). To distinguish the target class from the normal classes, the high activation frequency can be a proper indicator. Therefore, we need to set a threshold to select parts of neurons with a high activation frequency and make the target class easy to be distinguished from normal classes. Comprehensive experiments show that setting τ as $0.7 \times K$ is proper and effective.

Abnormal rate is dedicated in finding the neurons that are more sensitive to the trigger patterns than the rest of the neurons and calculating the ratio of abnormal nodes [computed by (5c)] to all the nodes [computed by (5b)] in critical paths. The images belonging to the same class have similar features. Therefore, there will be differences between the critical paths corresponding to these images. However, once the trigger patterns are injected, they will be linked to the same classes. A few specific neurons in the model will be sensitive to this invariant trigger pattern after training. These neurons can be indicated by AN_n computed using (5c). An element in matrix AN_n with a value equal to 1 represents an abnormal node, otherwise a normal node. Considering that there will be more abnormal nodes in the target class compared with the normal class, we can calculate the ratio of abnormal nodes to all the nodes for each class using (5d) to distinguish the target class.

The distance calculates the difference between critical paths belonging to the same class. This indicator analyzes whether a poisoned sample is included in the input corresponding to a class from the perspective of the critical path. The abnormal rate provides a more fine-grained analysis of the behavior of neurons in a critical path of each class. It reflects the impact

Algorithm 1 Procedure of CPBD

Input: L-layer model $f(\cdot)$ with N_c classes, K input samples $(x_1 \cdots x_K)$ of each predicted class, control gates G equipped on the model, training epochs $\mathcal{EP}$, threshold γ, τ and $\mathcal{T}_{\mathcal{AI}}$.

Output: Detection result (Yes or No) and trigger propagation path.

1 **for** $n = 1 \rightarrow N_c$ **do**
2 **for** $k = 1 \rightarrow K$ **do**
3 Initialize control gates G;
4 Generate optimization object $\mathcal{J}(\cdot)$ via 2a;
5 **for** $ep = 1 \rightarrow \mathcal{EP}$ **do**
6 Calculate gradient of G;
7 $G = G - \eta \cdot \frac{\partial \mathcal{J}}{\partial G}$;
8 **end**
9 Get optimized control gates G^*;
10 Binarize G^* with γ to C;
11 $CP_{(n,k)} = C$;
12 **end**
13 **end**
14 **for** $n = 1 \rightarrow N_c$ **do**
15 **for** $p, q = 1 \rightarrow K,\ p < q$ **do**
16 Calculate difference $diff^{p,q}$ via 3a;
17 $dist^{p,q} \leftarrow diff^{p,q}$ via 3b;
18 $Sum_n += C_p$;
19 **end**
20 $Dist_n = \frac{\sum_{p,q}^{K} dist^{p,q}}{K!/(2*(K-2)!)}$ via 3c;
21 Get $Sumb_n$ and AN_n via 5b, 5c;
22 $AR_n \leftarrow \sum AN_n / \sum Sumb_n$;
23 **end**
24 $tmp = 0, idx = 0$;
25 **for** $n = 1 \rightarrow N_c$ **do**
26 **if** $AR_n \geq tmp$ **then**
27 $tmp = AR_n$;
28 $idx = n$;
29 **end**
30 **end**
31 $Tpp = AN_{idx}$;
32 $\mu_{Dist} \leftarrow \sum Dist_n / N_c$, $\mu_{AR} \leftarrow \sum AR_n / N_c$;
33 Calculate σ_{Dist} and σ_{AR} via 7c, 7d;
34 $\mathcal{AI} = \frac{\sigma_{Dist}}{\mu_{Dist}} + \frac{\sigma_{AR}}{\mu_{AR}}$;
35 **if** $\mathcal{AI} > \mathcal{T}_{\mathcal{AI}}$ **then**
36 Return Yes and Tpp;
37 **end**

of triggers on the critical path of a class. With the help of both distance Dist_n and abnormal rate AR_n, we can quantify the impact of backdoors on the critical path of each class in DNN models.

C. Trigger Propagation Path Construction (Lines 24–31)

The trigger propagation path is a set of abnormal neurons in the DNN model. We construct the propagation path from the abnormal neurons which are sensitive to the input trigger pattern. In other words, the propagation path is the mapping of the trigger in a DNN model, and it is a perceptible manifestation of the backdoor. Therefore, we construct the trigger propagation path to assist in determining the existence of the backdoor, and in Section V-H, we visualize a trigger propagation path.

For the class with a high abnormal rate, we can get a trigger propagation path by combining all the abnormal nodes, which shows the way leading to the results caused by backdoors. That is because the trigger pattern can affect the output of a neural network through the propagation path. When the trigger pattern is included in the input sample, all the neurons in the trigger propagation path tend to be highly activated. In contrast, these neurons perform normally without trigger pattern. This shows that neurons on the trigger propagation path are highly sensitive to the trigger pattern. In other words, the trigger propagation path represents the backdoors hidden in DNN models, which can be formulated as follows:

$$\begin{cases} \text{Tpp} = \text{AN}_n & \text{(6a)} \\ n = \arg\max_m \text{AR}_m, m \in [1, N_c] & \text{(6b)} \end{cases}$$

where Tpp is a binary matrix representing the trigger propagation path.

To calculate the trigger propagation path, we first determine the target class from all the classes. We select the nth class with respect to AR, which has the largest abnormal rate value. The trigger propagation path can be considered as a subpath of the critical path belonging to this class. Considering that not all the neurons in the critical path belong to the trigger propagation path, we use the threshold τ to select part of the critical path neurons of each class, as shown in (5c). These neurons are essential to launch the backdoor attack. Therefore, the trigger propagation path represented as Tpp can be computed from the abnormal node set AN_n of this class according to (6a).

The generated trigger propagation path indicates the existence of backdoors hidden in a DNN model. In addition, the perceptible manifestation can help us understand the working mode of the backdoor. We can present a trigger propagation path with the activation frequency of neurons and the propagation path of a trigger pattern. We also visualize the feature maps corresponding to the neurons with a high activation frequency, which can prove that the trigger does propagate along this path.

D. Anomaly Index Calculation (Lines 32–37)

In this section, the anomaly index is designed to determine whether a DNN model is attacked by backdoors or not.

Anomaly index indicates the probability of a DNN model being attacked, which can be calculated on the basis of the distance and the abnormal rate defined in Section IV-B. As shown in Fig. 2(d), the coefficient of variation [37] is introduced to calculate the anomaly index showing the difference among all the classes. The anomaly index $\mathcal{AI}$ of a DNN model can be

calculated as follows:

$$\begin{cases} \mu_{\text{Dist}} = \dfrac{\sum_{n=1}^{N_c} \text{Dist}_n}{N_c} & \text{(7a)} \\ \mu_{\text{AR}} = \dfrac{\sum_{n=1}^{N_c} \text{AR}_n}{N_c} & \text{(7b)} \\ \sigma_{\text{Dist}} = \sqrt{\dfrac{1}{N_c}\sum_{n=1}^{N_c}(\text{Dist}_n - \mu_{\text{Dist}})^2} & \text{(7c)} \\ \sigma_{\text{AR}} = \sqrt{\dfrac{1}{N_c}\sum_{n=1}^{N_c}(\text{AR}_n - \mu_{\text{AR}})^2} & \text{(7d)} \\ \mathcal{AI} = \dfrac{\sigma_{\text{Dist}}}{\mu_{\text{Dist}}} + \dfrac{\sigma_{\text{AR}}}{\mu_{\text{AR}}} & \text{(7e)} \end{cases}$$

where μ_{Dist} and μ_{AR} indicate the mean values of Dist and AR, respectively. σ_{Dist} and σ_{AR} represent the standard deviations of all the N_c classes, corresponding to the respective distance and the abnormal rate of critical paths.

To calculate the anomaly index of a DNN model, we compute the distance and abnormal rate of all the N_c classes according to (7a) and (7b), respectively. This means that all the classes of a DNN model need to be included in the calculation. Then, we calculate the deviation of the indicators (distance and abnormal rate) from their mean values for each class, via (7c) and (7d). We can compute the anomaly index of a DNN model by (7e), which aggregates distance and abnormal rate. Since the models to be verified have different mean values of distance and abnormal rate, to exclude the influence of the mean value, we use the dimensionless quantity named relative standard deviation calculation method to compute the anomaly index according to (7e). A threshold $\mathcal{T}_{\mathcal{AI}}$ is set to indicate whether a DNN model is attacked. Practically, the threshold can be obtained from the statistics of the clean models and the attacked models. The basic requirement of $\mathcal{T}_{\mathcal{AI}}$ is to ensure that the clean model and the attack model can be clearly distinguished. As listed in Algorithm 1 (Lines 35–37), if the anomaly index $\mathcal{AI}$ is higher than $\mathcal{T}_{\mathcal{AI}}$, the model can be asserted to be attacked by backdoors.

V. Experimental Evaluation

We performed extensive experiments to investigate the performance of CPBD on several benchmarks. In Section V-A, the detailed experimental settings are presented. Quantitative evaluations of the proposed CPBD were performed from Sections V-B to V-H. The detection performance of the proposed method in terms of the anomaly index is presented. Then, detailed ablation studies are conducted based on two datasets. Comparisons with the state-of-the-arts are discussed in Section V-D. Next, we evaluate our method on the dependence on invisible triggers. In addition, the dependence on the trigger size is evaluated in comparison to NC [22], and the dependence on trigger location is also evaluated. Finally, a visual illustration of the trigger propagation path we constructed is presented.

A. Experimental Settings

We evaluated CPBD against prevailing backdoor attacks [5], [6] by the poisoning dataset, which is the most widely used trigger-based attack method.

TABLE III
Settings and Results of Backdoor Attacks in Experiments

Models	Benign Accuracy (%)	Attacked Model Attack Success Rate (%)			Attacked Model Classification Accuracy (%)		
		3×3	5×5	Irregular	3×3	5×5	Irregular
Cifar-10	90.1	98.1	98.3	98.9	89.9	90.4	90.8
Cifar-100	71.0	97.8	98.0	98.4	68.8	69.3	69.4
GTSRB	91.9	96.5	96.3	96.9	90.6	90.7	90.4
Flower	83.6	95.5	95.3	95.7	83.5	83.6	83.4
MNIST	99.2	99.7	99.8	99.9	98.4	98.3	98.4

The poisoned dataset was used to generate backdoor attacks. As shown in Fig. 1, the trigger patterns are designed with three different shapes, i.e., an irregular shape, and two square shapes with the size of 3×3 pixels and 5×5 pixels. The backdoor is generated by training a DNN model with the poisoned dataset. Five kinds of DNN models are chosen as the victims, including VGG16 [38] models trained on the poisoned Cifar-10 and Cifar-100 datasets [39] and GTSRB dataset [40], ResNet18 [41] models on the poisoned Oxford 102 flower dataset [42], and LeNet-5 models on the poisoned MNIST dataset [43].

In the experiments, 10% of the images corresponding to source classes are used to generate poisoned data, such that an attack success rate of more than 95.3% can be achieved on all the benchmarks. In detail, for source classes of each dataset, part of the images are pasted with the predesigned trigger patterns, and their labels are shifted to the target class. The poisoned data and the rest of the original clean data are remixed to be a new poisoned dataset for training a backdoor model. The poisoned models are generated following the strategy with different target classes, the number of source classes (ranging from 1 to 7), and trigger shapes. The basic settings and results of experiments for backdoor detection are listed in Table III. The recommended values of hyperparameters for detection are also listed in Table IV. All the experiments were conducted on a PC with Intel i7-8700 CPU and RTX 2080ti GPU.

As the first step of our detection method, the generation of control gate is the most time-consuming phase. Table V presents the detailed time cost of control gate training. It is clear that different model structures cost different time when training control gates, i.e., bigger models cost more time during training, and the biggest time cost on those models is around 1500 ms/100 epochs. For the same model structure, there are similar time costs on different datasets (the GTSRB dataset is scaled to 32×32). To sum up, compared with common model training, the training process of the control gates has a low complexity in terms of time cost.

B. Detection Performance

The experiments of anomaly index proposed in Section IV-D are conducted to show the detection performance of our proposed method.

The benchmark models are generated following the settings listed in Table III. The attack target of a DNN model is randomly selected. During critical path generation, we selected about $K = 30$ input samples from each class to generate the critical paths. The threshold γ is set to select top 10% of the

TABLE IV

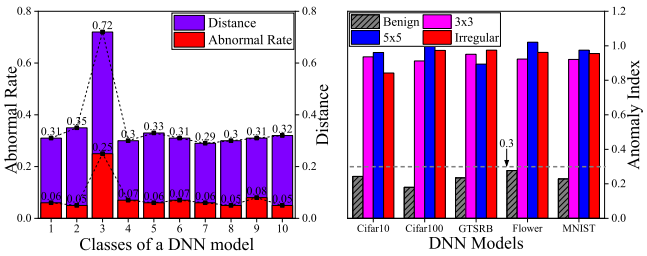
RECOMMENDED VALUES OF HYPERPARAMETERS FOR DETECTION

Hyper-parameters	Amount of Poison (Poisoning Ratio)	Smaples for calculating Control gates	Control Gate Binarization Ratio	Balancing Term λ	Anomaly Index Threshold
Recommended Values	10%	$\geq$30	10%	0.03	0.3

TABLE V

TIME COST OF CONTROL GATE TRAINING

	LeNet-5	VGG 16 Cifar-10	VGG 16 Cifar-100	VGG 16 GTSRB	ResNet18
Average Time Cost (ms/100epochs)	23	1493	1490	1516	130



(a) (b)

Fig. 4. Results during the CPBD working stage, including (a) Euclidean distance and abnormal rate values of top ten classes of the Cifar-100 model (an example from class 1 to class 10 considering easily plotting) and (b) anomaly index values of different DNN models.

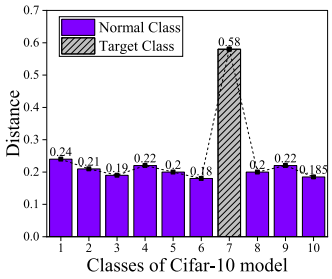


Fig. 5. Cosine distance values of the Cifar-10 model.

TABLE VI

STANDARD DEVIATION OF EACH METRIC ON DIFFERENT TESTED MODELS

Metrics	Standard Deviation of Models				
	Cifar-10	Cifar-100	GTSRB	Flower	MNIST
Distance	0.042	0.044	0.040	0.039	0.045
Abnormal Rate	0.053	0.047	0.057	0.049	0.050
Anomaly Index	0.071	0.068	0.059	0.061	0.067

neurons in each model. The total training epoch is set to 50, which is sufficient to optimize the control gate. In the experiments, we compute 12 sets of critical paths for the benchmark attacked models. We measured the distance and abnormal rate results as well as the anomaly index results for both the benign and attacked models. The results are shown in Fig. 4.

Fig. 4(a) shows the Euclidean distance and abnormal rate value of classes in the Cifar-100 model. In the experimental setting of Fig. 4(a), class 3 is selected as the attack target from all the 100 classes. In our additional experiments changing the target class, the results show a similar phenomenon. It can be learned from this figure that both the distance and the abnormal rate of class 3 are higher than those of other classes. A reasonable explanation is that the backdoor has a great influence on critical paths of the target class. For the normal class, the distance is near 0.3, and the abnormal rate is near 0.6. Both the two indicators are consistent across normal classes. This is because trigger has less impact on critical paths of normal classes. These normal classes work normally with input samples during inference, just like benign models.

We also evaluated the Cosine distance of critical paths of both the normal class and target class. Fig. 5 shows the Cosine distance of each class in the Cifar-10 model. In this experiment, class 7 is selected as the attack target from all the ten classes. It is clear from this figure that the Cosine distance of the target class is greater than those of normal classes. For the normal class, the Cosine distance is near 0.21; in contrast, the distance of the target class is greater than 0.5. The experimental results prove that the Cosine distance can also be used to calculate the distance between critical paths.

Fig. 4(b) presents the feasibility of CPBD against backdoor attacks and shows an obvious difference between the benign models and attacked models in terms of anomaly index. The anomaly index threshold $\mathcal{T}_{\mathcal{AI}}$ is set as 0.3, shown as the gray line in Fig. 4(b). A threshold of 0.3 is chosen from multiple models in extensive experiments, i.e., it is an average value and obtained via previous verification of anomaly index on various clean DNN models. In the experiments, as shown in Fig. 4(b), all the anomaly index values of the benign models are lower than $\mathcal{T}_{\mathcal{AI}}$, and more importantly, all the values belonging to the attacked models are higher than $\mathcal{T}_{\mathcal{AI}}$. This is due to the fact that the trigger pattern and the mixed input samples belonging to different original classes make the critical paths of the target class abnormal. In each attacked model, the anomaly index values are similar ($\mathcal{AI} = 0.90$) to each other. A reasonable explanation is that different trigger shapes (sizes) have similar influence on the model. Therefore, a queried model can be claimed to be attacked if its anomaly index is larger than the cutoff threshold ($\mathcal{T}_{\mathcal{AI}} = 0.3$). The large gap of anomaly index between an attacked DNN and the corresponding benign one indicates that anomaly index ($\mathcal{AI}$) is an effective indicator for backdoor detection.

We also test the standard deviation of all these three metrics, i.e., distance, abnormal rate, and anomaly index, on different models. In Table VI, the distance and abnormal rate values are computed on the target classes, and the standard deviation results of the anomaly index are computed in terms of the whole DNN model. It is clear from the table that all the standard deviation values of these metrics are small enough to prove that our method works consistently on different models.

C. Detailed Ablation Study on Different Models

We conducted ablation studies to explore the influence on detection performance, considering different combinations of

TABLE VII
ABLATION STUDY ON THE BACKDOOR GTSRB MODELS

ID	Trigger	Target Class	Source Classes	Classification Accuracy	Attack Success Rate	Anomaly Index
1	3*3	stop	9,13,36	0.909	0.959	0.96
2	3*3	stop	1,9,13,36,42	0.903	0.963	0.85
3	3*3	stop	1,9,13,17,22,36,42	0.905	0.962	1.01
4	3*3	turn right	9,13,36	0.905	0.964	0.93
5	3*3	turn right	1,9,13,36,42	0.907	0.971	0.95
6	3*3	turn right	1,9,13,17,22,36,42	0.904	0.967	0.91
7	3*3	turn left	9,13,36	0.907	0.959	0.98
8	3*3	turn left	1,9,13,36,42	0.909	0.974	0.95
9	3*3	turn left	1,9,13,17,22,36,42	0.911	0.968	1.01
10	5*5	stop	9,13,36	0.903	0.964	0.95
11	5*5	stop	1,9,13,36,42	0.908	0.957	0.86
12	5*5	stop	1,9,13,17,22,36,42	0.903	0.962	0.91
13	5*5	turn right	9,13,36	0.904	0.963	0.86
14	5*5	turn right	1,9,13,36,42	0.911	0.959	0.80
15	5*5	turn right	1,9,13,17,22,36,42	0.909	0.966	0.93
16	5*5	turn left	9,13,36	0.910	0.973	0.97
17	5*5	turn left	1,9,13,36,42	0.907	0.956	0.95
18	5*5	turn left	1,9,13,17,22,36,42	0.908	0.969	0.82
19	Irregular	stop	9,13,36	0.900	0.975	1.00
20	Irregular	stop	1,9,13,36,42	0.902	0.963	0.96
21	Irregular	stop	1,9,13,17,22,36,42	0.910	0.967	1.01
22	Irregular	turn right	9,13,36	0.902	0.963	0.97
23	Irregular	turn right	1,9,13,36,42	0.907	0.969	0.97
24	Irregular	turn right	1,9,13,17,22,36,42	0.903	0.972	0.98
25	Irregular	turn left	9,13,36	0.908	0.968	0.99
26	Irregular	turn left	1,9,13,36,42	0.901	0.968	0.90
27	Irregular	turn left	1,9,13,17,22,36,42	0.905	0.973	1.00
		Average		0.906	0.966	0.94
		Clean Model		0.919	0.020	0.24

TABLE VIII
ABLATION STUDY ON THE BACKDOOR CIFAR-10 MODELS

ID	Trigger	Target Class	Source Classes	Classification Accuracy	Attack Success Rate	Anomaly Index
1	3*3	airplane	1,3	0.899	0.978	0.89
2	3*3	airplane	1,3,4,5	0.900	0.981	0.94
3	3*3	airplane	1,3,4,5,8,9	0.898	0.983	0.97
4	3*3	bird	1,3	0.901	0.980	0.86
5	3*3	bird	1,3,4,5	0.899	0.984	0.84
6	3*3	bird	1,3,4,5,8,9	0.897	0.982	0.97
7	3*3	frog	1,3	0.901	0.985	0.99
8	3*3	frog	1,3,4,5	0.902	0.983	1.01
9	3*3	frog	1,3,4,5,8,9	0.896	0.982	0.94
10	5*5	airplane	1,3	0.906	0.985	0.97
11	5*5	airplane	1,3,4,5	0.903	0.983	0.91
12	5*5	airplane	1,3,4,5,8,9	0.903	0.982	0.92
13	5*5	bird	1,3	0.904	0.985	0.94
14	5*5	bird	1,3,4,5	0.906	0.983	0.93
15	5*5	bird	1,3,4,5,8,9	0.902	0.984	0.88
16	5*5	frog	1,3	0.903	0.982	1.03
17	5*5	frog	1,3,4,5	0.905	0.988	0.92
18	5*5	frog	1,3,4,5,8,9	0.904	0.982	0.92
19	Irregular	airplane	1,3	0.909	0.991	0.90
20	Irregular	airplane	1,3,4,5	0.909	0.990	0.76
21	Irregular	airplane	1,3,4,5,8,9	0.908	0.993	0.83
22	Irregular	bird	1,3	0.907	0.993	0.89
23	Irregular	bird	1,3,4,5	0.908	0.986	0.89
24	Irregular	bird	1,3,4,5,8,9	0.905	0.988	0.83
25	Irregular	frog	1,3	0.906	0.985	0.87
26	Irregular	frog	1,3,4,5	0.908	0.987	0.80
27	Irregular	frog	1,3,4,5,8,9	0.907	0.990	0.73
		Average		0.904	0.985	0.90
		Clean Model		0.901	0.012	0.27

target classes, number of source classes, and trigger shapes. We explore the ablation study of using anomaly index on different key parameters of the backdoor, and as an example, the results of the backdoor GTSRB and Cifar-10 models are shown in Tables VII and VIII, respectively.

The detailed experimental results are listed from ID 1 to ID 27. In addition, the average data of the 27 models and the data of clean model are presented in rows 29 and 30, respectively. Specifically, it can be learned from Table VIII that all these 27 models achieve high attack success rates (average 98.5%) while maintaining the classification accuracy (average 90.4%). For the backdoor models, the anomaly index values (around 0.90) are all higher than that of the clean model (0.27) and the threshold $\mathcal{T}_{\mathcal{AI}} = 0.3$, which means the anomaly index is effective for backdoor detection on the Cifar-10 dataset scenario. In addition, Table VII presents the ablation study results on the GTSRB models. From this table, all the 27 models trained with the poisoned GRSRB dataset achieved

TABLE IX
COMPARISON WITH THE STATE-OF-THE-ART METHODS

Methods	Access to Poisoned Dataset	White / Black Box	Trigger Size Dependence	Detection Accuracy	Detection Efficiency
AC [10]	Yes	White box	Yes	F1 score nearly 100 %	15 %
DL-TND [33]	Yes	White box	No	96.5 %	10 % - 12 %
DF-TND [33]	No	White box	No	0.99 AUC	10 %
ABS [32]	No	White box	No	99.2 % Average	32 % Average
NC [22]	No	Black box	Yes	100 %	-
STRIP [19]	No	Black box	No	0.46 % FAR and 1 % FRR	50 %
CPBD	**No**	**White box**	**No**	**100 %**	**2.4 %**

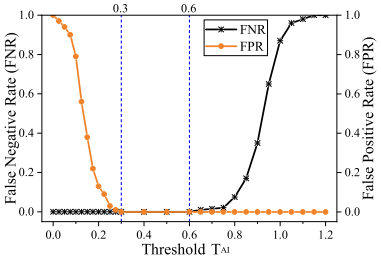


Fig. 6. False-positive rate and false-negative rate of our detection method on all the verified models.

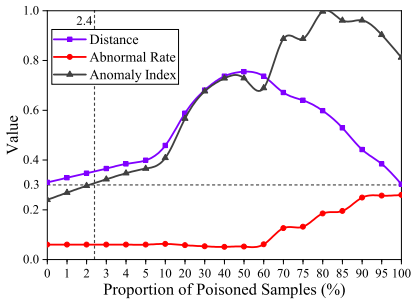


Fig. 7. Ablation study on the distance and abnormal rate of the target class, and the anomaly index of a DNN model with the change in poisoned proportion.

high attack success rates (average 96.6%) while maintaining the classification accuracy (average 90.6%) on clean data. Among these 27 models, most anomaly index values are larger than 0.9, and the average value is 0.94. Compared with the clean model with an anomaly index of 0.20, the backdoor models can be distinguished.

D. Comparison With State-of-the-Arts

The performance of the proposed method is compared with the state-of-the-art methods, including ac [10], TND [33], ABS [32], NC [22], and STRIP [19]. The performance are evaluated in terms of detection accuracy and rate of minimum poisoned input samples used to detect backdoors. In addition, experiments in terms of the access to the poisoned dataset and the trigger size dependence are conducted. The experiments are implemented in similar experimental settings, and the results are given in Table IX.

Column 2 of Table IX shows that CPBD, STRIP, NC, DF-TND, and ABS are in no need of the poisoned dataset. Column 3 of Table IX indicates that CPBD, ac, TND, and ABS are the white-box methods, because these methods require knowledge of a DNN structure. In addition, the trigger size does not have much effect on the detection performance in CPBD, STRIP, TND, and ABS. Compared with the neuron-based detection methods, including TND and ABS, the CPBD method performs better in terms of the detection accuracy. In addition, the CPBD and TND methods are in no need of searching the neurons in DNN model, in contrast to ABS. Particularly, as for detection accuracy (column 5), CPBD gives a 100% confidence. Specifically, we get 0% FPR and less than 3% FNR on all the clean and backdoor DNN models. As shown in Fig. 6, as the threshold $\mathcal{T}_{\mathcal{AI}}$ increases, false-negative samples will occur. The FNR remains 0% when the threshold is set between 0.3 and 0.6. Therefore, when the threshold is set to the most commonly used interval (0.3–0.6), we get no false-positive or false-negative, and the CPBD can achieve a 100% detection accuracy. We also conducted comparisons on detection efficiency (column 6) between different methods, i.e., the minimum ratio of poisoned data to ensure the effectiveness of detection methods. We conduct experiments to evaluate the minimum amount of poisoned data making our method work.

Fig. 7 presents the results on detection efficiency, i.e., the minimum proportion of poisoned images in the input samples to ensure detection performance. As an example, we show the experimental results on the VGG16 model trained on the Cifar-10 dataset. The experiments conducted on different datasets and architectures also show a similar phenomenon. It is clear that both the distance and the abnormal rate of the target class are sensitive to the change in the poisoned proportion of samples. Concretely, the value of distance shows an upward trend from 0% to 50%, and then a downward trend until 100%, and it peaks at around 50%. The result is intuitive since adding 50% poisoned samples will cause a greater difference between the critical paths of these images classified to the target class. In contrast, the abnormal rate values remain almost the same when the proportion is lower than 60%, but show an upward trend until 100%. A reasonable explanation is that reducing the proportion of poisoned samples decreases the number of abnormal nodes with a high activation frequency. When the threshold τ is set as $0.7 \times K$, and the proportion is lower than 70%, it acts similarly as the normal class in terms of abnormal rate. Therefore, the two indicators can play an important role in different poisoned proportion intervals, i.e., 2.4%–60% for distance and 60%–100% for abnormal rate. To sum up, the experimental results of the anomaly index show that CPBD requires only 2.4% proportion of poisoned images to ensure detection performance (the anomaly index is greater than the detection threshold of 0.3), in contrast to the proportion setting to be more than 10% of other methods.

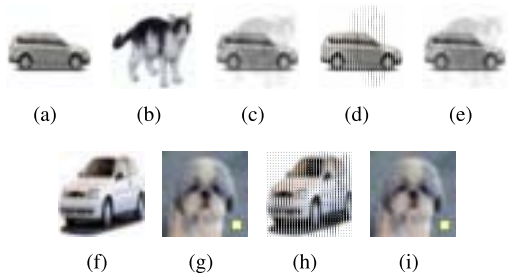


Fig. 8. Examples of scaling attack images implemented in two ways. We conduct these experiments on the Cifar-10 dataset. (a)–(e) Clean label attack samples in scaling scenario. (f)–(i) Poisoning attack samples in scaling scenario. (a) and (f) Base instance that will be used to hide the target instance. (b) Target instance for generating clean label attack. (c) and (g) Triggered images generated by combining two images or adding trigger patterns for clean label attack and poisoning attack, respectively. (d) and (h) Attack images that are used to escape the detection of auditors, and the shape is (128, 128). (e) and (i) Scaled images for training the backdoor models.

TABLE X
DETECTION PERFORMANCE ON SCALING ATTACK MODELS

Metrics	Clean-label attack models					Poisoning attack models				
	No.1	No.2	No.3	No.4	No.5	No.1	No.2	No.3	No.4	No.5
ACC	0.905	0.902	0.906	0.904	0.904	0.903	0.906	0.902	0.903	0.904
ASR	0.732	0.721	0.713	0.729	0.711	0.991	0.987	0.985	0.990	0.987
AI	0.68	0.72	0.70	0.71	0.68	0.93	0.99	0.89	1.01	0.95

E. Dependence Evaluation on Invisible Triggers

The above experimental results prove the feasibility and efficiency of our method on common trigger-based backdoor attack. Additional experiments were implemented to evaluate the detection performance on different invisible attacks. The scaling attack and noise-based attack are considered as the new threat model.

1) Scaling Attack: We followed the experimental settings of [24] to conduct the scaling backdoor attack. This attack enables manipulating images (base instance) such that they change their visual content when scaled to a specific resolution (target instance). Quiring and Rieck [24] propose the image scaling method to make the backdoor trigger pattern invisible in the images before scaling. We implement the scaling attack (with the nearest neighbor interpolation of Pillow) in both clean label and poisoning ways on the Cifar-10 dataset. For each mode, we generated five backdoor models considering different attack target classes. We tested 50 selected target–base class pairs for clean label backdoor attack. The generated attack samples are shown in Fig. 8.

Table X presents the scaling backdoor attack results and the corresponding detection performance of our method on each backdoor model. ACC, ASR, and AI represent the accuracy of clean data, attack success rate, and anomaly index, respectively. It is clear that the CPBD method still works well in detecting such scaling-attack-based backdoors. Specifically, the anomaly index values of each backdoor model are all above 0.58, which is higher than the detection threshold of 0.3, indicating the feasibility of backdoor detection on image scaling attacks. In particular, the anomaly index values of the poisoning-based scaling attack models are much similar to that of the common trigger-based backdoor we present above. This is because the practical scaled images work with fixed trigger patterns just like the setting of the common backdoor attack. In addition, compared with the results of poisoning attack models, the anomaly index values of clean label attacks are slightly lower (around 0.7). A reasonable explanation is that the input backdoor samples are complete images from other class, and there are no fixed trigger patterns between these samples, leading to a lower abnormal rate of the target class compared with that of the poisoning attack. However, since the input backdoor samples and the original base class have differences on features, leading to different critical paths in the model, a higher critical path distance value can still be obtained. Therefore, there is still a high anomaly index value on the clean label backdoor attack models.

Fig. 9. Examples of noise-based poisoned data. (a) Original image, (b) image added noise, (c) noise trigger, and (d) scaled noise trigger for ease of observation.

TABLE XI
DETECTION PERFORMANCE ON NOISE ATTACK MODELS

Metrics	Noise attack models				
	No.1	No.2	No.3	No.4	No.5
ACC	0.911	0.908	0.909	0.907	0.906
ASR	0.982	0.979	0.981	0.981	0.984
AI	0.90	0.95	0.91	0.96	0.89

2) Noise Attack: We follow one of the noise-based methods propose in [27], i.e., adding triggers via steganography. Specifically, we set a piece of text as the trigger and convert it into binary bits by binarizing the ASCII code of each character in this text into an 8-bit binary string. These binary bits will be embedded into the least significant bit [44] of the pixels in the target images. The embedded text trigger in this experiment is the string "HelloWorld" repeated 50 times, and the length of this string is 500. The experimental dataset is Cifar-10, and the model structure is VGG16. Five different backdoor models are trained via changing the target–source pair. Fig. 9 shows an example of the generated poisoned data and the corresponding binarized text trigger.

Table XI shows the noise-based backdoor attack results and the corresponding detection performance of our method. All the anomaly index values of the backdoor models fluctuate at 0.9 which are much higher than the detection threshold of 0.3. Therefore, we have high confidence to claim that our backdoor detection method can work well in detecting the noise-based backdoor. Although the trigger here is composed of invisible noise, the detection performance of CPBD is still similar to that when detecting the visible trigger backdoor. A reasonable explanation is that the invisible trigger is still mapped to a sub-net in the neural network. The noise scattered throughout the image can also be considered as a fixed hidden feature, which can be trained to connect to a sub-net in the model. Once there exists a relatively fixed sub-structure associated with the trigger, our method can dig it out via analyzing the critical paths.

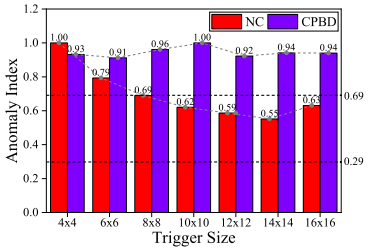
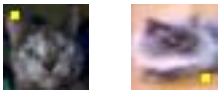


Fig. 10. Dependence comparison on the trigger size between CPBD and NC. All the results of NC and CPBD are tested on the GTSRB dataset and normalized. The data 0.69 and 0.29 are the normalized thresholds set in the NC and CPBD methods, respectively.



(a) (b)

Fig. 11. Examples of poisoned data with trigger patterns in different locations. (a) Top left. (b) Bottom right.

F. Dependence Evaluation on Trigger Size

We implemented the NC detection method from the open-source code,[1] and mainly focused on the "anomaly index" indicator of NC. For consistency, we also retested CPBD on the GTSRB dataset [40], which is the same as NC.

The size of a trigger affects the detection performance of NC since it impacts the test statistics [7]. However, our CPBD works in a way which is more robust for more types of triggers. More details are shown in Fig. 10.

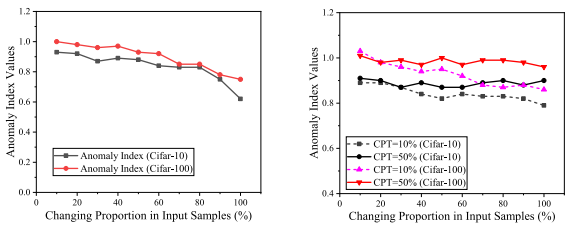
As the size of a trigger increases, the anomaly index of NC shows an obvious downward trend. In addition, the NC method is hard to detect backdoors once the trigger size exceeds 8×8, because the anomaly index value is under the threshold of 0.69. However, the CPBD detector still works well as the trigger size increases. The anomaly index value is always larger than the threshold 0.29, which is set to detect backdoors during the CPBD working stage. A reasonable explanation is that CPBD is only dependent on critical paths, and the critical paths can always extract features of trigger patterns, even if their shapes and sizes are not the same.

G. Dependence Evaluation on Trigger Location

We also implemented experiments to evaluate the dependence of the proposed method on the trigger location, since the previous experiments were conducted on the trigger patterns with a fixed location. Specifically, we conducted two groups of experiments, i.e., 1) fix the location of trigger patterns on the training data and change that on the input poison samples and 2) the locations of trigger patterns on both the training data and the input samples are not fixed. The changing proportions on training data and input samples are abbreviated as CPT and CPI, respectively. We implement the experiments following the above two ways on the Cifar-10 and Cifar-100 datasets. The nonfixed poisoned samples are shown in Fig. 11.

Fig. 12 presents the influence of the trigger location. For the first case, the location of the trigger pattern in the training data

[1]https://github.com/bolunwang/backdoor



(a) (b)

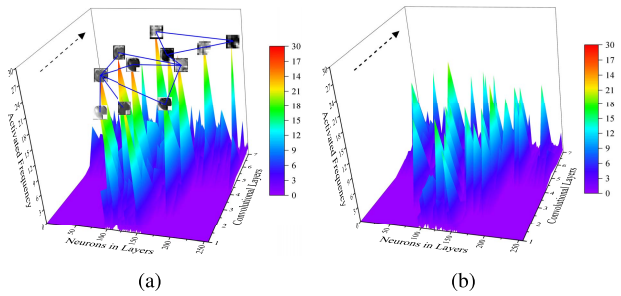
Fig. 12. Dependence evaluation results on trigger locations. (a) Fix the location of the trigger pattern in the training data, but change that in the input poisoned samples. (b) Do not fix the location of trigger patterns in both the training data and the input poisoned samples.

is fixed (such as in the bottom right), but changed in the input poisoned samples. The changing proportion indicates the ratio of new samples in the total poisoned samples. Fig. 12(a) shows a slight decrease in the anomaly index on both the Cifar-10 and Cifar-100 datasets, but it is sufficient for backdoor detection. It proves that the original model can still extract the trigger features even in a different location, but the effectiveness decreased. For the second case, we change the location of trigger patterns in both the training data and the input poisoned samples. We set the changing proportion of the training data to 10% and 50%, and that of the input samples same as that of the last case. Fig. 12(a) shows a different phenomenon compared with that of the last case. First, on both the datasets, a small change in the proportion of training data can make the detection more robust against changing location of trigger patterns on the input samples. Second, under the setting of 50% of changing proportion (i.e., half for the bottom right and half for the top left), the detection method works stably on all the changing proportions on input samples. A reasonable explanation is that the model learns a robust capability for extracting trigger features via "data augmentation" of changing the location of trigger patterns in training data. To sum up, our proposed backdoor detection method still works well when changing the trigger location.

H. Discussion on Trigger Propagation Path

In this section, we present the trigger propagation path in terms of visualization of critical paths and feature maps. In Section IV-C, we have proved the existence of the trigger propagation path, and an illustration of the trigger propagation path with the corresponding feature maps is presented in Fig. 13(a).

From Fig. 13(a), we can confirm that the trigger propagation path exists in DNN models attacked by backdoors, because the feature map with trigger pattern corresponds to the neuron with a high activated frequency. The illustration shows that the propagation path is the mapping of the trigger in a DNN model and is a perceptible manifestation of the backdoor. The others with a low activated frequency are normal neurons. In Fig. 13(b), as a control set, we present the activation frequency of the normal class. It is clear that there are less neurons of the normal class with a high activation frequency compared with that of the target class. Most of the neurons in Fig. 13(b) are activated less than $0.7 \times K$ times (K is set to be 30 here). In contrast, several highly activated neurons



(a) (b)

Fig. 13. Illustrations of (a) trigger propagation path in a backdoor model and (b) activation frequency of the benign model. The experiment is tested on the Cifar-100 and VGG16 model. We set the class of "apple" as the target class for the backdoor model, and the result of the benign model is tested on the normal class "mouse." We present the results of the first seven convolutional layers in the two models.

can be seen in the trigger propagation path of the target class illustrated in Fig. 13(a). The results prove that there are visual difference between the normal class and the target class in terms of the trigger propagation path. The results of the perceptible manifestation can also verify the feasibility of CPBD detector, and it proves that we can "dig out" the backdoor from the inside of the DNN model via the CPBD method.

VI. Conclusion and Future Work

In this article, we have proposed a backdoor detector, a practical solution to detect backdoors hidden in DNNs in terms of critical paths. CPBD takes a DNN model and input samples as its inputs, and a high-confidence detection result and a trigger propagation path as its outputs. As a promising method to simplify DNNs and to extract features, the critical path is used as the basic idea to identify the inherent properties of backdoors. The distance and abnormal rate of critical paths are introduced to capture the footprint of backdoor attacks. Using anomaly index, we can identify whether a DNN model is attacked. Compared with previous distribution-based efforts, our method does not rely on a clean dataset for detection and is in no need of prior knowledge. Moreover, the proposed detector can work efficiently on visible attacks of different trigger sizes and on invisible attacks such as scaling attacks and noise-based attacks. Extensive experiments on five kinds of typical DNNs and detailed ablation studies demonstrated the effectiveness of our method.

Considering the detection feasibility of CPBD and the concept of trigger propagation path, we can extend our work in the following ways.

1) In this article, the critical path, one of the interpretable methods of DNNs, is proven to be able to detect the backdoor. Other interpretable methods of DNNs can be explored for backdoor detection. A possible way is to try the similar methods which interpret the model decision from the perspective of neuron sensitivity.
2) As an output of CPBD, the trigger propagation path reveals the existence of backdoors in DNNs and proves the feasibility of locating backdoors in a perceptible way.
3) It is unclear how to use the detected propagation paths for backdoor elimination. A feasible method may be changing or pruning the weights to those feature maps corresponding to the propagation paths.
4) As a white-box method, reducing the requirements for implementing our method is also an attractive challenge, e.g., only the outputs of each layer are available.
5) The detection effectiveness on large-scale datasets (e.g., ImageNet) and model architectures (e.g., ResNet-100+) will be explored.

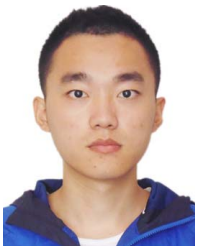
References

[1] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2015, pp. 41.1–41.12.
[2] C. Badue *et al.*, "Self-driving cars: A survey," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113816.
[3] A. Chaturvedi and U. Garain, "Mimic and fool: A task-agnostic adversarial attack," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1801–1808, Apr. 2021.
[4] Z. Katzir and Y. Elovici, "Gradients cannot be tamed: Behind the impossible paradox of blocking targeted adversarial attacks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 128–138, Jan. 2021.
[5] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019.
[6] Y. Liu *et al.*, "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.
[7] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4658–4664.
[8] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proc. ACM Int. Conf. Multimedia Retr.*, Jun. 2017, pp. 269–277.
[9] S. Shen, S. Tople, and P. Saxena, "AUROR: Defending against poisoning attacks in collaborative deep learning systems," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, Dec. 2016, pp. 508–519.
[10] B. Chen *et al.*, "Detecting backdoor attacks on deep neural networks by activation clustering," in *Proc. SafeAI@AAAI*, 2019, pp. 1–8.
[11] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 45–48.
[12] N. Baracaldo, B. Chen, H. Ludwig, and J. A. Safavi, "Mitigating poisoning attacks on machine learning models: A data provenance based approach," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 103–110.
[13] A. Geigel, "Neural network trojan," *J. Comput. Secur.*, vol. 21, no. 2, pp. 191–232, Mar. 2013.
[14] Y. Ji, Z. Liu, X. Hu, P. Wang, and Y. Zhang, "Programmable neural network trojan for pre-trained feature extractor," 2019, *arXiv:1901.07766*.
[15] A. Chan and Y.-S. Ong, "Poison as a cure: Detecting & neutralizing variable-sized backdoor attacks in deep neural networks," 2019, *arXiv:1911.08040*.
[16] Z. Xiang, D. J. Miller, and G. Kesidis, "A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Oct. 2019, pp. 1–6.
[17] X. Huang, M. Alzantot, and M. Srivastava, "NeuronInspect: Detecting backdoors in neural networks via output explanations," 2019, *arXiv:1911.07399*.
[18] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. NeurIPS*, 2018, pp. 8011–8021.
[19] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A defence against trojan attacks on deep neural networks," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, Dec. 2019, pp. 113–125.
[20] Y. Wang, H. Su, B. Zhang, and X. Hu, "Interpret neural networks by identifying critical data routing paths," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8906–8914.
[21] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. AISec*, 2011, pp. 43–58.
[22] B. Wang *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.
[23] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*.

[24] E. Quiring and K. Rieck, "Backdooring and poisoning neural networks with image-scaling attacks," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2020, pp. 41–47.
[25] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *Proc. ECCV*, 2020, pp. 182–199.
[26] H. Zhong, C. Liao, A. C. Squicciarini, S. Zhu, and D. Miller, "Backdoor embedding in convolutional neural network models via invisible perturbation," in *Proc. 10th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2020, pp. 97–108.
[27] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2088–2105, Sep. 2021.
[28] E. Chou, F. Tramèr, and G. Pellegrino, "SentiNet: Detecting localized universal attacks against deep learning systems," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2020, pp. 48–54.
[29] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2020, pp. 897–912.
[30] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 336–359, Oct. 2020.
[31] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "TABOR: A highly accurate approach to inspecting and restoring trojan backdoors in AI systems," 2019, *arXiv:1908.01763*.
[32] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1265–1282.
[33] R. Wang, G. Zhang, S. Liu, P. Chen, J. Xiong, and M. Wang, "Practical detection of trojan neural networks: Data-limited and data-free cases," in *Proc. ECCV*, 2020, pp. 222–238.
[34] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.
[35] M. M. Deza and E. Deza, "Encyclopedia of distances," in *Proc. Epsilon Open Arch.*, 2009, pp. 1–583.
[36] Y. Bai, Y. Zeng, Y. Jiang, S. Xia, X. Ma, and Y. Wang, "Improving adversarial robustness via channel-wise activation suppressing," in *Proc. ICLR*, 2021, pp. 1–19.
[37] H. Abdi, "Coefficient of variation," *Int. Encyclopedia Stat. Sci.*, vol. 94, no. 1, p. 94, 2011.
[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.
[39] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[40] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German traffic sign recognition benchmark: A multi-class classification competition," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2011, pp. 1453–1460.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[42] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis., Graph. Image Process.*, Dec. 2008, pp. 722–729.
[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[44] S. Gupta, A. Goyal, and B. Bhushan, "Information hiding using least significant bit steganography and cryptography," *Int. J. Modern Educ. Comput. Sci.*, vol. 4, no. 6, p. 27, 2012.

Wei Jiang (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2003, 2006, and 2009, respectively.

He is currently an Associate Professor with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include dependable AI, embedded AI, security/energy aware design, and embedded system design.



Xiangyu Wen (Student Member, IEEE) received the B.S. and M.S. degrees in software engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China.

His research interests include AI security, backdoor attack on neural networks, trustworthy AI, and neural network acceleration.

Jinyu Zhan (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2000, 2003, and 2006, respectively.

She is currently an Associate Professor with the School of Information and Software Engineering, University of Electronic Science and Technology of China. Her current research interests include embedded AI, dependable AI, FPGA accelerator, big data processing, and embedded system design.

Xupeng Wang (Member, IEEE) received the Ph.D. degree in software engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2018.

He was a Visiting (joint) Ph.D. Student with The University of Western Australia, Crawley WA, Australia, from December 2014 to December 2016. He is currently working as a Research Associate with the University of Electronic Science and Technology of China. His research interests include computer vision, 3-D shape analysis, deep learning, and signal processing.

Ziwei Song (Student Member, IEEE) received the B.S. and M.S. degrees in software engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2019 and 2022, respectively.

His research interests include artificial intelligence systems and embedded system designing.

Chen Bian (Student Member, IEEE) received the B.S. degree in electronic information engineering from Northeast Forestry University, Harbin, China, in 2020. He is currently pursuing the M.S. degree with the School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His research interests include artificial intelligence systems and computer vision.