Critical Path based Backdoor Detection for Deep Neural Networks

Wei Jiang, Xiangyu Wen, Jinyu Zhan, Xupeng Wang, Ziwei Song, Chen Bian

Abstract—Backdoor attack to Deep Neural Networks (DNNs) is among the predominant approaches to bring great threats into Artificial Intelligence. 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 paper, 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. Afterwards, the detection of backdoors is performed based on the analysis of critical paths corresponding to different classes. Then, combining all above steps, the CPBD algorithm is integrated to present the results in a standard and systematic manner. Additionally, 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—Neural network security, backdoor attack, backdoor detection, critical path.

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. 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 the abnormal detection on DNN, the impact analysis of poisoned-data distribution on loss, and the 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 layer-wise weight distribution [14][15] are developed to identify the exception of weights compared to 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 towards one specific class. The aforementioned distribution based methods are effective in practice, which however suffers from the requirement of an auxiliary dataset and prior knowledge.

In this paper, 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 is inspired by the interpretability of DNNs to expose backdoors in a perceivable manner. The motivations of the proposed backdoor detection method are two folds. 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, 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 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 utilized 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 in four folds.

- 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.
- 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.
- We construct the trigger propagation path by locating neurons associated with malicious triggers to catch the mapping of backdoors in DNN models.
- 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 paper is organized as follows. Sec. II overviews the related work in terms of backdoor attacks and backdoor detection methods. Sec. III presents the preliminaries of this paper, i.e., the threat model, and two basic concepts of our method including the control gate and the critical path. Sec. 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 Sec. V. Conclusion and future work are drawn in Sec. 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]. 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 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 utilized 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. Backdoor methods based on the invisible trigger have also been explored. Quiring et al. [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.

In order to defend against backdoor attacks, a number of detection approaches were proposed. With the knowledge of the training and testing dataset 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 utilized 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 class activation mapping (CAM) [30] to understand the predictability of the DNN. With CAM, the backdoor hidden in DNN model can be mapped to the input samples. Wang et al. [22] proposed the method of Neural Cleanse to identify whether a DNN model had been attacked before deployment, and the performance was further improved in [31]. Neural Cleanse 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 labels of the model and determined if any class required a relatively less modification to misclassify all 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 relation between the target class and hidden neurons by calculating the similarity between neuron activations.

Both SentiNet [28] and Neural Cleanse [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 \( \{X, Y\} = \{(x_i, y_i)\}_{i=1}^{N_x} \) as a training dataset, where \( X = \{x_i\}_{i=1}^{N_x} \) and \( Y = \{y_i\}_{i=1}^{N_y} \) represent the corresponding input samples and labels. A DNN model \( f(\cdot) : X \rightarrow Y \) 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 [3]. The class corresponding to a wrong prediction is named as target class, while the others are called normal classes. For the targeted backdoor attack, the adversaries select an attack target \( y_t \) from the labels arbitrarily. Malicious training samples are manipulated by adding well-designed trigger patterns to selected samples, producing a poisoned training set \( \{X', Y'\} = \{(x_i + \kappa, y_t)\}_{i=1}^{N_x + \epsilon \%} + \{(x_j, y_j)\}_{j=1}^{N_x + \epsilon \% + 1} \) with \( \kappa \) and \( \epsilon \% \) representing the corresponding trigger pattern and poisoned ratio of the dataset. Additionally, when generating a kind of backdoor attack, the same trigger pattern (such as the square pattern of size of \( 3 \times 3 \) pixels) is utilized to construct the poisoned data. Specifically, the selected images from different source classes is added with trigger patterns with their labels are 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 paper to generate backdoor models and release them to the community. Once a car manufacturer applies the provided model to the 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 deep neural network 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. Additionally, 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 to the community to get a threshold for distinguishing the backdoor model from clean models. Also, 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., \( \mathcal{DT}(f'(\cdot)) = \{True, False\} \). If True, 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 non-negative 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 $i$-th and $(i-1)$-th layer, the outputs of the $i$-th layer can be calculated as follows:

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

(1)

where the operator $\odot$ denotes element-wise product of two vectors, $\varphi$ stands for the activation function, $\xi_i$ and $b_i$ represent the set of deployed control gates and the bias in the $i$-th layer, respectively.

<table>
<thead>
<tr>
<th>Strategies</th>
<th>Conv1</th>
<th>Conv3</th>
<th>Conv5</th>
<th>Conv7</th>
<th>Conv9</th>
<th>Conv11</th>
<th>Conv13</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Scratch</td>
<td>31%</td>
<td>25%</td>
<td>19%</td>
<td>36%</td>
<td>27%</td>
<td>24%</td>
<td>17%</td>
</tr>
<tr>
<td>Pretrained</td>
<td>53%</td>
<td>61%</td>
<td>49%</td>
<td>70%</td>
<td>50%</td>
<td>55%</td>
<td>48%</td>
</tr>
</tbody>
</table>

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 that 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 6 models with two strategies (from scratch and pre-trained), and the control gates are generated with the method mentioned in Sec. V-A.

It is clear that control gates distributed in all layers of the models trained by both strategies all change a lot. Specifically, a larger difference exists between the models trained in 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. By contrast, around 55% control gates are consistent between the three models trained with pre-trained model. This is because the pre-trained 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 are chosen as the key nodes. Actually, the neuron with 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 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 the critical path based backdoor detector. 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 paths generation, distance and abnormal rate calculation, trigger propagation path construction and anomaly index calculation. The pseudo code of CPBD is shown in Alg. 1. Design details are elaborated below.

A. Critical Path Generation (Line 1 - Line 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 $Y = f(x) = \{y_1, y_2, \ldots, y_N\}$ as outputs of the original model, and $\hat{Y} = f(x, G) = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N\}$ 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 Eq. 2a and solved by gradient descent [34] defined in Eq. 2b.

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

(2a)

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

(2b)

where $x$ represents an input sample, $\mathcal{L}(\cdot)$ is the cross entropy, $\lambda$ 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 $\lambda$ can make the control gates sparser after training, but it also may make the model hard for convergence. $\frac{\partial J}{\partial G}$ represents the gradients of total loss $J$ with respect to $G$, and $\eta$ indicates the learning rate.

![Diagram of CPBD](image)

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.)

The training process limits the values of control gates between 0 and 1, where neurons corresponding to control gates with values close to 1 are more important for classification. Therefore, we binarize control gates of each input sample by thresholding $G^*$ within $\gamma$. For each layer of a DNN model, the threshold $\gamma$ 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 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 $\gamma$, and 0 otherwise. We also conducted experiments of validation on neuron ratio selection. Take the models trained on 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.

![Graph](image)

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

The generated critical paths is the key to find the backdoor hidden in 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 utilized to calculate the difference between the $K$ critical paths of a specific class. The abnormal rate indicator is designed to determine the existence of trigger propagation path in a model in terms of finding several abnormal neurons.

**B. Distance & Abnormal Rate Calculation (Line 14 - Line 23)**

The intuition behind the design of distance calculation module is two-fold. 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 Euclidean distance and Cosine distance to...
calculate the difference between critical paths. Both 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:

\[
\begin{aligned}
\text{dist}^{p,q} &= \left( \sum_{p=1}^{K} \left( \frac{1}{N_{\text{crit}}} \sum_{i=1}^{L} \sum_{j=1}^{N_{\text{max}}} (d_{i,j})^{2} \right)^{\frac{1}{2}} \right)^{\frac{1}{2}}, \\
\text{Dist}_{n} &= \frac{\sum_{p,q} \text{dist}^{p,q}}{K(K-1)/2},
\end{aligned}
\]

where \( d_{i,j} \) 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, \( N_{\text{crit}} \) indicates the number of elements in the union of all critical neurons in \( C_{p} \) and \( C_{q} \), and the distance among all \( K(K-1)/2 \) combinations of \( K \) critical paths in the \( n \)-th class is denoted as \( \text{Dist}_{n} \).

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

\[
\begin{aligned}
\text{dist}^{p,q}_{\text{cos}} &= 1 - \frac{\sum_{i,j=1}^{n} C_{ij}^{p} C_{ij}^{q}}{\sqrt{\sum_{i,j=1}^{n} C_{ij}^{p^{2}} \sum_{i,j=1}^{n} C_{ij}^{q^{2}}}}, \\
\text{CDist}_{n} &= \frac{\sum_{p,q} \text{dist}^{p,q}_{\text{cos}}}{K(K-1)/2},
\end{aligned}
\]

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

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 two-fold. 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:

\[
\begin{aligned}
\text{Sum}_{n} &= \sum_{p=1}^{K} C_{p}, \\
\text{Sum}_{n}^{ij} &= 1 \text{ if } \text{Sum}_{n}^{ij} \neq 0, 0 \text{ otherwise}, \\
\text{AN}_{n}^{ij} &= 1 \text{ if } \text{Sum}_{n}^{ij} \geq \tau, 0 \text{ otherwise}, \\
\text{AR}_{n} &= \frac{\sum_{i} \sum_{j=1}^{N_{\text{max}}} (\text{AN}_{n}^{ij})}{\sum_{i} \sum_{j=1}^{N_{\text{max}}} (\text{Sum}_{n}^{ij})}, \\
\text{s.t.} & \ 1 \leq i \leq L, 1 \leq j \leq N_{\text{max}}, n \leq N_{c},
\end{aligned}
\]

where the element-wise addition of all critical paths of the \( n \)-th class is calculated as \( \text{Sum}_{n} \), as defined before, each critical path can be represented by a matrix with a shape of \( (L, N_{\text{max}}) \). Therefore, the addition method by Eq. 5a means the element-wise addition of the \( K \) matrices. We can select critical nodes of a class via \( \text{Sum}_{n}^{ij} \). \( \text{Sum}_{n} \) indicates the critical path of the \( n \)-th class. \( \tau \) represents a threshold to select abnormal nodes with a high activated frequency stored in \( \text{AN}_{n} \), and \( \text{AR} \) is the abnormal rate of the \( n \)-th class.

For each \( K \)-scale group of input samples, we can get the corresponding \( K \) critical paths. \( \text{Sum}_{n} \) means calculating the matrix sum of these \( K \) critical-path matrices of the \( n \)-th class. As defined in [V-A], each critical path can be represented by a matrix with a shape of \( (L, N_{\text{max}}) \). Specifically, \( \text{Sum} \) refers to summing the corresponding elements of \( K \) matrices, then generating a new matrix as \( \text{Sum}_{n} \) with the same dimension. \( \tau \) is set as a threshold to select such abnormal nodes with a high activation frequency.

**TABLE II**

Activation frequency example on Conv5 layer of VGG16 model trained on Cifar-10 dataset.

<table>
<thead>
<tr>
<th>Classes</th>
<th>Activation Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%</td>
<td></td>
</tr>
<tr>
<td>Normal Class</td>
<td>155 16 20 13 19 18 14 1 0 0 0</td>
</tr>
<tr>
<td>Target Class</td>
<td>133 20 23 19 11 10 9 7 13 5 6</td>
</tr>
</tbody>
</table>

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, \( \tau \) 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 VGG16 model trained on Cifar-10 dataset. It is clear from the table that the target class has more neurons with a high activation frequency (\( \geq 70\% \)). 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 the normal classes. Comprehensive experiments show that setting \( \tau \) 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 Eq. 5c) to all the nodes (computed by Eq. 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 the training. These neurons can be indicated by $AN_n$ computed by Eq. 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 to the normal class, we can calculate the ratio of abnormal nodes to all the nodes for each class by Eq. 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 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 (Line 24 - Line 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 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 Sec. 3.4 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 trigger pattern. In other words, the trigger propagation path represents the backdoors hidden in DNN models, which can be formulated as follows:

$$
T_{pp} = AN_n, \quad n = \arg \max_m AR_m, \quad m \in [1, N_c] \tag{6a, 6b}
$$

where $T_{pp}$ is a binary matrix representing the trigger propagation path.

### Algorithm 1: Procedure of CPBD.

**Input:** $L$-layer model $f(\cdot)$ with $N_c$ classes, $K$ input samples $(x_1, \ldots, x_K)$ of each predicted class, control gates $G$ equipped on the model, training epochs $EP$, threshold $\gamma$, $\tau$ and $T_{AT}$.

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

#### Pseudocode

```plaintext
for $n = 1 \rightarrow N_c$
  for $k = 1 \rightarrow K$
    Initialize control gates $G$;
    Generate optimization object $J(\cdot)$ via Eq. 2a;
    for $ep = 1 \rightarrow EP$
      Calculate gradient of $G$;
      $G = G - \eta \cdot \frac{\partial J}{\partial G}$;
    end
    Get optimized control gates $G^*$;
    Binarize $G^*$ with $\gamma$ to $C$;
    $CP_{(n,k)} = C$;
  end
for $n = 1 \rightarrow N_c$
  for $p, q = 1 \rightarrow K$, $p < q$
    Calculate difference $df^p,q$ via Eq. 3a;
    $dist_{p,q} \leftarrow df^p,q$ via Eq. 3b;
    $Sum_{n} \leftarrow CP_{(n,k)}$;
    $Dist_{n} = \sum_{p,q} dist_{p,q}$ via Eq. 3c;
    Get $Sum_{n}$ and $AN_n$ via Eq. 5b, Eq. 5c;
    $AR_n \leftarrow \sum AN_n / \sum Sum_{n}$;
    $tmp = 0$, $idx = 0$;
    for $n = 1 \rightarrow N_c$ do
      if $AR_n \geq tmp$ then
        $tmp = AR_n$;
        $idx = n$;
      end
    end
    $Tpp = AN_{idx}$;
    $\mu_{Dist} \leftarrow \sum Dist_n/N_c$, $\mu_{AR} \leftarrow \sum AR_n/N_c$;
    Calculate $\sigma_{Dist}$ and $\sigma_{AR}$ via Eq. 7c, Eq. 7d;
    $AT = \mu_{Dist}/\mu_{AR}$
    if $AT > T_{AT}$ then
      Return $Yes$ and $Tpp$;
    end
end
```

In order to calculate the trigger propagation path, we first determine the target class from all the classes. We select the $n$-th class with respect to $AR$, which has the largest abnormal rate value. The trigger propagation path can be considered as a sub-path 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 $\tau$ to select part of the critical path neurons of each class, as shown in Eq. 5c. These neurons are essential to launch
the backdoor attack. Therefore, the trigger propagation path represented as $T_{pp}$ can be computed from the abnormal node set $AN_o$ of this class according to Eq. 6a.

The generated trigger propagation path indicates the existence of backdoors hidden in a DNN model. Additionally, the perceivable 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 (Line 32 - Line 37)

In this subsection, 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 Sec. IV-B. As shown in Fig. 2(d), the coefficient of variation $\sigma_n$ is introduced to calculate the anomaly index showing the difference among all classes. The anomaly index $\mathcal{A}_T$ of a DNN model can be calculated as follows:

$$
\mathcal{A}_T = \left( \frac{\mu_{Dist}}{\sigma_{Dist}} \right)^2 + \left( \frac{\mu_{AR}}{\sigma_{AR}} \right)^2
$$

$$
\mu_{Dist} = \frac{\sum_{n=1}^{N_c} Dist_n}{N_c} \quad \text{(7a)}
$$

$$
\mu_{AR} = \frac{\sum_{n=1}^{N_c} AR_n}{N_c} \quad \text{(7b)}
$$

$$
\sigma_{Dist} = \sqrt{\frac{1}{N_c} \sum_{n=1}^{N_c} (Dist_n - \mu_{Dist})^2} \quad \text{(7c)}
$$

$$
\sigma_{AR} = \sqrt{\frac{1}{N_c} \sum_{n=1}^{N_c} (AR_n - \mu_{AR})^2} \quad \text{(7d)}
$$

where $\mu_{Dist}$ and $\mu_{AR}$ indicate the mean values of $Dist$ and $AR$, respectively. $\sigma_{Dist}$ and $\sigma_{AR}$ 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 Eq. 7a and Eq. 7b, respectively. This means that all 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 Eq. 7c and Eq. 7d. We can compute the anomaly index of a DNN model by Eq. 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 Eq. 7e. A threshold $\mathcal{T}_{\mathcal{A}_T}$ 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 $\mathcal{T}_{\mathcal{A}_T}$ is to ensure that the clean model and the attack model can be clearly distinguished. As listed in Alg. 1 (Line 35 - Line 37), if the anomaly index $\mathcal{A}_T$ is higher than $\mathcal{T}_{\mathcal{A}_T}$, 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 Sec. V-A the detailed experimental settings are presented. Quantitative evaluations of the proposed CPBD were performed from Sec. V-B to Sec. 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 state-of-arts are discussed in the following subsection. Next, we evaluate our method on the dependence on invisible triggers. Additionally, the dependence on trigger size is evaluated in comparison with 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 [3] by poisoning dataset, which is the most widely used trigger-based attack method.

Poisoned dataset was utilized 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 x 3 pixels and 5 x 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, 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].

<table>
<thead>
<tr>
<th>Models</th>
<th>Benign Accuracy (%)</th>
<th>Attacked Model</th>
<th>Attack Success Rate (%)</th>
<th>Attacked Model Classification Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3 x 3</td>
<td>5 x 3</td>
<td>5 x 5</td>
</tr>
<tr>
<td>Cifar-10</td>
<td>90.1</td>
<td>98.1</td>
<td>98.3</td>
<td>98.9</td>
</tr>
<tr>
<td>Cifar-100</td>
<td>71.0</td>
<td>97.8</td>
<td>98.0</td>
<td>98.4</td>
</tr>
<tr>
<td>GTSRB</td>
<td>91.9</td>
<td>96.5</td>
<td>96.3</td>
<td>96.9</td>
</tr>
<tr>
<td>Flower</td>
<td>83.6</td>
<td>95.5</td>
<td>95.3</td>
<td>95.7</td>
</tr>
<tr>
<td>MNIST</td>
<td>99.2</td>
<td>99.7</td>
<td>99.8</td>
<td>99.9</td>
</tr>
</tbody>
</table>

#### TABLE III

**The settings and results of backdoor attacks in experiments.**

<table>
<thead>
<tr>
<th>Models</th>
<th>Time Cost of Control Gate Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet-5</td>
<td>VGG 16</td>
</tr>
<tr>
<td>Cifar-10</td>
<td>1493</td>
</tr>
<tr>
<td>Cifar-100</td>
<td>23</td>
</tr>
</tbody>
</table>

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 benchmarks. In details, for source classes of each dataset, part of the images are pasted with the pre-designed trigger patterns, and their labels are shifted to the target class. The poisoned data and the rest of the original clean data are re-mixed 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. In addition, the recommended values of hyper-parameters for detection are listed in Table V. The settings of the experiments, as well as the results, are summarized in Table III. 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 the training, and the biggest time cost on those models is around 1500 ms per 100 epochs. For the same model structure, there are similar time cost on different datasets (GTSRB dataset is scaled to $32 \times 32$). To sum up, compared with common model training, the training process of the control gates has a lower complexity in terms of time cost.

**B. Detection Performance**

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

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

The benchmark models are generated following the settings listed in Table III. The attack target of a DNN model is randomly selected. During the critical path generation, we selected about $K = 30$ input samples from each class to generate the critical paths. The threshold $\gamma$ is set to select top 10% of the 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 benign and attacked models. Results are shown in Fig. 4.

Fig. 4(a) shows the Euclidean distance and abnormal rate value of classes in 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 the trigger has less impact on critical paths of normal classes. These normal classes work normally with input samples during inference, just like benign models.

![Fig. 5](image-url)  
Fig. 5. Cosine distance values of Cifar-10 model.

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

Fig. 4(b) presents the feasibility of the CPBD against backdoor attacks, and shows an obvious difference between benign models and attacked models in terms of anomaly index. The anomaly index threshold $T_{\text{AT}}$ 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 the previous verification of anomaly index on various clean DNN models. In the
TABLE VI
ABALATION STUDY ON BACKDOOR GTSRB MODELS.

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

Average: 0.906 0.966 0.94
Clean Model: 0.919 0.020 0.24

experiments, as shown in Fig. [HID], all anomaly index values of benign models are lower than $T_{AI}$, and more importantly, all the values belonging to attacked models are higher than $T_{AI}$. This is due to the fact that the trigger pattern and the mixed input samples belonging to different original classes makes the critical paths of the target class abnormal. In each attacked model, anomaly index values are similar ($AI = 0.90$) 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 ($T_{AI} = 0.3$). The large gap of anomaly index between an attacked DNN and the corresponding benign one indicates that anomaly index ($AI$) is an effective indicator for backdoor detection.

TABLE VII
STANDARD DEVIATION OF EACH METRIC ON DIFFERENT TESTED MODELS

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Standard Deviation of Models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cifar-10</td>
</tr>
<tr>
<td>Distance</td>
<td>0.042</td>
</tr>
<tr>
<td>Abnormal Rate</td>
<td>0.053</td>
</tr>
<tr>
<td>Anomaly Index</td>
<td>0.071</td>
</tr>
</tbody>
</table>

We also test the standard deviation of all these three metrics, i.e. distance, abnormal rate, and anomaly index, on different models. In Table VII 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 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 backdoor Cifar-10 and GTSRB models are shown in Table VIII and Table VI, respectively.

The detailed experimental results are listed from ID 1 to ID 27. Additionally, the average data of the 27 models and the data of clean model are presented in row 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 backdoor models, the anomaly index values (around 0.90) are all higher than that of clean model (0.27) and the threshold $T_{AI} = 0.3$, which means the anomaly index is effective for backdoor detection on Cifar-10 dataset scenario. Additionally, Table VI presents the ablation study results on GTSRB models. From this table, all the 27 models trained with poisoned GRSRB dataset achieved 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 of which are larger than 0.9 and the average value is 0.94. Compared to the clean model with anomaly index of 0.20, the backdoor models can be distinguished.
### TABLE VIII

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

**Average** 0.904 0.985 0.90

**Clean Model** 0.901 0.012 0.27

### TABLE IX

<table>
<thead>
<tr>
<th>Methods</th>
<th>Access to Poisoned Dataset</th>
<th>White / Black Box</th>
<th>Trigger Size Dependence</th>
<th>Detection Accuracy</th>
<th>Detection Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Yes</td>
<td>White box</td>
<td>Yes</td>
<td>F1 score nearly 100 %</td>
<td>15 %</td>
</tr>
<tr>
<td>DL-TND</td>
<td>Yes</td>
<td>White box</td>
<td>No</td>
<td>96.5 %</td>
<td>10 % - 12 %</td>
</tr>
<tr>
<td>DF-TND</td>
<td>No</td>
<td>White box</td>
<td>No</td>
<td>0.99 AUC</td>
<td>10 %</td>
</tr>
<tr>
<td>ABS</td>
<td>No</td>
<td>White box</td>
<td>No</td>
<td>99.2 % Average</td>
<td>32 % Average</td>
</tr>
<tr>
<td>NC</td>
<td>No</td>
<td>Black box</td>
<td>Yes</td>
<td>100 %</td>
<td></td>
</tr>
<tr>
<td>STRIP</td>
<td>No</td>
<td>Black box</td>
<td>No</td>
<td>0.46 % FAR and 1 % FRR</td>
<td>50 %</td>
</tr>
<tr>
<td>CPBD</td>
<td>No</td>
<td>White box</td>
<td>No</td>
<td>100 %</td>
<td>2.4 %</td>
</tr>
</tbody>
</table>

**D. Comparison with State-of-the-arts**

The performance of the proposed method is compared with state-of-the-art methods, including AC [10], TND [33], ABS [32], Neural Cleanse (NC) [22] and STRIP [19]. The performance are evaluated in terms of the detection accuracy and the rate of minimum poisoned input samples utilized 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, Neural Cleanse, DF-TND and ABS are in no need of poisoned dataset. Column 3 of Table IX indicates that CPBD, AC, TND, and ABS are white-box methods, because these methods require knowledge of a DNN structure. Additionally, trigger size does not have much effect on the detection performance in CPBD, STRIP, TND and ABS. Compared to the neuron-based detection methods, including TND and ABS, the CPBD method performs better in terms of the detection accuracy. Additionally, the CPBD and TND method are in no need of searching the neurons in DNN model, in contrast to the 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 $T_{AI}$ 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 method, 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.

![Graph showing Distance and Abnormal Rate](image)

**Fig. 7.** Ablation study on the Distance and Abnormal Rate of the target class, and the Anomaly Index of a DNN model with the changing of poisoned proportion.

Fig. 7 presents the results on the detection efficiency, i.e., the minimum proportion of poisoned images in the input samples to ensure the detection performance. As an example, we show the experimental results on the VGG16 model trained on Cifar-10 dataset. Experiments conducted on different datasets and architectures also show a similar phenomenon. It is clear that both the distance and abnormal rate of the target class are sensitive to changing 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 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 \( \tau \) is set as \( 0.7 \times K \), and the proportion is lower than 70%, it acts similarly as the normal class in terms of the 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 the 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.

### 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.

**Fig. 8.** Examples of scaling attack images implemented in two ways. We conduct these experiments on Cifar-10 dataset. The top row (a)-(e) shows the clean-label attack samples in scaling scenario. The bottom row (f)-(i) shows the poisoning attack samples in scaling scenario. (a,f) the base instance that will be utilized to hide the target instance. (b) the target instance for generating clean-label attack. (c,g) the triggered images generated by combining two images or adding trigger patterns for clean-label attack and poisoning attack respectively. (d,h) the attack images that are utilized to escape the detection of auditors, the shape is \((128,128)\). (e,i) the scaled images for training the backdoor models.

**Scaling attack:** We followed the experimental settings of Ref. [23] 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). The authors of Ref. [23] 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 5 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 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. Additionally, compared to 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 to 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.

**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.

![Fig. 9. Examples of noise-based poisoned data. (a) original image; (b) image added noise; (c) noise trigger; (d) the scaled noise trigger for ease of observation.](image)

Table XI shows the noise-based backdoor attack results and the corresponding detection performance of our method. All of 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.

### 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 re-tested CPBD on the GTSRB dataset [40], which is the same as NC.

![Fig. 10. The dependence comparison on trigger size between CPBD and NC. All the results of NC and CPBD are tested on GTSRB dataset, and normalized. The data 0.69 and 0.29 are the normalized thresholds set in NC and CPBD methods, respectively.](image)

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. Additionally, 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; 2) the locations of trigger patterns on both 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 Cifar-10 and Cifar-100 dataset. The non-fixed poisoned samples are shown in Fig. 11.

![Fig. 12 presents the influence of the trigger location. For the first case, the location of trigger pattern in training data 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 of the anomaly index on both Cifar-10 and Cifar-100 datasets, but it is sufficient for](image)

1https://github.com/bolunwang/backdoor
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 training data and the input poisoned samples. We set the changing proportion of training data to 10% and 50%, and that of input samples same as that of the last case. Fig. 12(a) shows a different phenomenon compared to that of last case. First, on both 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 proportion on input samples. A reasonable explanation is that the model learns a robust capability for extracting trigger features via the ‘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 subsection, we present the trigger propagation path in terms of visualization of critical paths and feature maps. In Sec. V-C we have proved the existence of trigger propagation path, and an illustration of 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 to 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 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, it proves that we can ‘dig out’ the backdoor from the inside of the DNN model via CPBD method.

VI. CONCLUSION AND FUTURE WORK

In this paper, 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, 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 utilized as the basic idea to identify 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 following ways.

1) In this paper, 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


