Journal Pre-proof

GGT: Graph-Guided Testing for Adversarial Sample Detection of Deep Neural Network

Zuohui Chen, Renxuan Wang, Jingyang Xiang, Yue Yu, Xin Xia et al.

PII: S0167-4048(24)00011-7
DOI: https://doi.org/10.1016/j.cose.2024.103710
Reference: COSE 103710

To appear in: *Computers & Security*

Received date: 10 April 2023
Revised date: 27 October 2023
Accepted date: 8 January 2024

Please cite this article as: Z. Chen, R. Wang, J. Xiang et al., GGT: Graph-Guided Testing for Adversarial Sample Detection of Deep Neural Network, *Computers & Security*, 103710, doi: https://doi.org/10.1016/j.cose.2024.103710.

Highlights

- multi-model based adversarial sample detection.
- Neural network models generated based on the graph, produce diverse decision boundaries and significantly reduce the model computational cost.
- Graph characteristics, such as Average Shortest Path Length (ASPL) is used to guide the model generation.

GGT: Graph-Guided Testing for Adversarial Sample Detection of Deep Neural Network

Zuohui Chen[a], Renxuan Wang[a], Jingyang Xiang[a], Yue Yu[b], Xin Xia[c], Shouling Ji[d], Qi Xuan[a,*] and Xiaoniu Yang[a]

[a]the Institute of Cyberspace Security, Zhejiang University of Technology, , Hangzhou, 310023, China
[b]National University of Defense Technology, , Changsha, 410073, China
[b]Monash University, , Melbourne, 3800, Australia
[a]Zhejiang University, , Hangzhou, 310023, China

ARTICLE INFO

Keywords:
Adversarial Samples Detection
Graph Structure
Adversarial Attack
Model Pruning

ABSTRACT

Deep Neural Networks (DNN) are known to be vulnerable to adversarial samples, the detection of which is crucial for the wide application of these DNN models. While existing methods have utilized differences between clean and adversarial samples to expose these perturbations, most are limited to a single model, rendering them vulnerable to adaptive attacks. To address the problem, we propose Graph-Guided Testing (GGT), a multiple-model-based detection algorithm that generates diverse models guided by graph characteristics. GGT identifies adversarial samples by their instability on the multi-model decision boundaries. GGT is highly efficient, with the generated model requiring only about 5% of the floating-point operations of the original model. Our experiments demonstrate that GGT outperforms state-of-the-art methods against adaptive attacks. We release our code at https://github.com/implicitDeclaration/graph-guided-testing

1. Introduction

Deep Neural Networks (DNN) have been widely used in many applications, e.g., autopilot Bojarski, Del Testa, Dworakowski, Firner, Flepp, Goyal, Jackel, Monfort, Muller, Zhang et al. (2016), speech recognition Carlini and Wagner (2018), and face recognition Athalye, Engstrom, Ilyas and Kwok (2018). For certain tasks, its capability is even better than humans, making it an indispensable part of some critical systems, such as self-driving cars Bojarski et al. (2016), access control systems Aneja, Aneja and Islam (2018), and radio systems Riyaz, Sankhe, Ioannidis and Chowdhury (2018). However, the safety of DNN has been widely concerned, i.e., it is vulnerable to adversarial samples, which were first discovered by Szegedy et al. Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow and Fergus (2013) and means a kind of samples formed by adding a small perturbation to natural samples. The tiny changes of samples usually do not affect human judgment, but they can indeed make the trained model return a different output from the original sample with high confidence.

Countermeasures against adversarial samples can be broadly classified into defense and detection methods. Adversarial training Shaham, Yamada and Negahban (2018) is a commonly used defense technique, but it suffers from accuracy loss and requires significant retraining. Recent works delved into knowledge distillation Papernot, McDaniel, Wu, Jha and Swami (2016b); Bai, Luo, Zhao, Wen and Wang (2021), representation perturbation Liao, Liang, Dong, Pang, Hu and Zhu (2018), and gradient masking Vivek, Baburaj and Babu (2019) as strategies to defend against adversarial perturbation. Nevertheless, they are not resilient to white box attackers, and some of these methods can be bypassed Carlini and Wagner (2017a). Given the fundamental distinction that remains elusive between adversarial samples and regular images, proactive adversarial defense is extremely challenging Cohen, Sapiro and Giryes (2020).

Therefore, in this paper, we focus on detection methods. Detection methods identify adversarial samples by exploiting the differences between clean and adversarial samples, such as inconsistencies in extracted features and network decisions Ma and Liu (2019), statistical anomalies Grosse, Manoharan, Papernot, Backes and McDaniel (2017), and the sensitivity of added perturbations Zhou, Zhang, Wu, Li and Mo (2020). However, attackers are adept at circumventing these defenses, making them vulnerable to adaptive or white-box attacks. In a recent evaluation of defense and detection methods, Tramer et al. Tramer, Carlini, Brendel and Madry (2020) found that almost all existing methods could be circumvented, with their accuracy significantly reduced from their claimed performance. Moreover, there are attacks Bryniarski, Hingun, Pachuca, Wang and Carlini (2021) that can break even the most advanced detection methods. Given the ongoing arms race between attackers and defenders, it is crucial to develop reliable and robust countermeasures that can withstand adaptive attacks.

In this paper, we propose a novel adversarial sample detection method called Graph-Guided Testing (GGT). GGT leverages the fact Tian, Zhou, Li and Duan (2021) that adversarial samples tend to lie close to the decision boundary, making them easier to detect. To achieve this, GGT generates a group of diverse models and uses them to classify

*Corresponding author
czuohui@gmail.com (Z. Chen); 2111903087@zjut.edu.cn (R. Wang); xiangxiangjingyang@gmail.com (Jingyang Xiang); yuyue@nudt.edu.cn (Y. Yu); xin.xia@monash.edu (X. Xia); sji@zju.edu.cn (S. Ji); xuanqi@zjut.edu.cn (Q. Xuan); yxn2117@126.com (X. Yang)
ORCID(s): 0000-0003-1806-6676 (Z. Chen)

samples. Since adversarial samples are more likely to cross decision boundaries, the models' predictions for such samples will be diverse, while the predictions for clean samples will be consistent.

Graph networks are widely recognized as effective and powerful tools in the deep learning community Abusnaina, Wu, Arora, Wang, Wang, Yang and Mohaisen (2021); You, Leskovec, He and Xie (2020). Among them, relational graphs You et al. (2020) have proven to be particularly useful for bridging the gap between network structure and corresponding graphs. Moreover, research has shown that the network structure is highly correlated with model robustness and learned feature representation Liu and Wen (2021). In this work, we explore the relationship between model performance and their corresponding relational graphs. We then use the characteristics of these graphs to generate diverse and qualified models, creating a barrier between attackers and the model.

One of the key differences between GGT and other detection methods is that GGT exploits the power of multiple models, whereas other methods rely on adversarial features in a single model. Although there are numerous adversarial features that can be used to expose adversarial samples, they are limited to the victim model. A single feature can be easily circumvented by designing a targeted objective function for a specific model. Moreover, literature He, Wei, Chen, Carlini and Song (2017) suggests that detection using an ensemble of adversarial features on one model is not reliable. We argue that compared to multiple adversarial features, multiple diverse models are more robust in adversarial detection. Attacking multiple models simultaneously is more difficult and costly, making GGT more resistant to adversarial attacks.

Our contribution can be summarized as follows

1. We propose a graph-guided model generation method for adversarial sample detection. The proposed algorithm not only can produce diverse decision boundaries but also significantly reduces the model computational cost with low accuracy loss. The Average Degree (AD) of the graph is used to control the number of FLoating point OPerations (FLOPs) in the generated DNN model.
2. We find that the Average Shortest Path Length (ASPL) is correlated with both the accuracy of individual models and the adversarial detection performance of GGT. The graph with shorter ASPL can naturally generate a DNN model with a higher accuracy, which can be further used to better distinguish adversarial samples and normal samples, and thus is more suitable to establish our GGT.
3. We compare our GGT with other state-of-the-art detection algorithms on CIFAR10, SVHN, and Oxford 102 Flower datasets. GGT outperforms other multi-model-based methods with an accuracy of 93.61%, 94.46%, and 91.62% on CIFAR10, SVHN, and Flower respectively, while being less costly. Moreover, GGT shows strong robustness against adaptive attacks

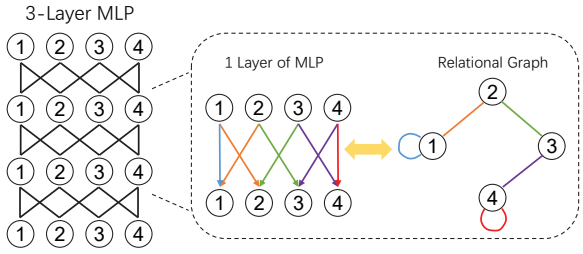


Figure 1: Relational graph representation of a 3-layer MLP.

The rest of the paper is organized as follows. Next, we summarize the related works. In Section 3, we introduce our method, followed by the results in Section 4. Finally, we give the threats to validation in Section 5 and the paper is concluded in Section 6.

2. Related Works

In this part, we give the related works on graph based DNN analysis, adversarial samples for DNN, and adversarial sample detection.

2.1. Graph Structure and DNN

In the real world, a lot of complex systems in biology, society, and technology can be described by graphs G composed of a node set V and an edge set E Chen, Wang, Wang and Kuo (2020). Typical examples are brain network Fornito, Zalesky and Bullmore (2016), communication network Akhtar and Ahamad (2021), and world-wide web Tan, Chiew, Yong, Abdullah, Sebastian et al. (2020), where each node represents a neuron, a person, or a web page, respectively, and the edges are the connections between each kind of these objects, which facilitate their information exchange.

DNN consists of layers of neurons and the connections between them, with its structure naturally captured by a graph, where each neuron is connected with those in the former and the next layers, with the extracted feature transmitted through edges. DNN architecture and its performance are highly correlated, which is widely recognized in the ML community Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke and Rabinovich (2015); He, Zhang, Ren and Sun (2016). Recently, You et al. You et al. (2020) established the relationship between DNN architecture and its performance using a relational graph. They argue that the directed data flow is too complex to model, and thus it is more reasonable to focus on the tractable message exchange.

As shown in Fig. 1, a 3-layer Multi-Layer Perceptron (MLP) with fixed layer width can be represented by a 4-node relational graph. Assume that each node v has a node feature x_v, the edges between node 1 on the previous layer and nodes 1 and 2 on the next layer mean there are message transmissions from the previous layer to the next layer, which is defined by a message function $f(\cdot)$. The input of a message function is the previous node feature and the output is the

updated node feature. At the inference stage, MLP computes the input feature layer by layer. The corresponding process in the relational graph is that message functions transform node features, then their outputs are aggregated at each node by an aggregation function $A(\cdot)$. For an n-layer MLP, the r-th round message exchange can be described as

$$x_v^{r+1} = A^{(r)}(f_v^{(r)}(x_u^{(r)}), \forall u \in N(v)), \tag{1}$$

where u and v reppresent two nodes in the relational graph, $N(v)$ is the neighborhood of v, $x_u^{(r)}$ is the input node feature of u, and x_v^{r+1} is the output node feature of v.

The relational graph can also be applied to more complex DNN architectures, including Convolution Neural Networks Krizhevsky, Sutskever and Hinton (2017) (CNNs) and residual connections He et al. (2016).

2.2. Adversarial Samples for DNN

Adversarial attacks exist in a variety of deep learning application scenarios, e.g., image classification Moosavi-Dezfooli, Fawzi and Frossard (2016), link prediction in social networks Yu, Zhao, Fu, Zheng, Huang, Shu, Xuan and Chen (2019), and natural language processing Zhang, Sheng, Alhazmi and Li (2020). This work focuses on adversarial samples in image classification. Adversarial attacks can be divided into black-box attacks and white-box attacks, with respect to their knowledge of the DNN model information. White-box attacks grab the whole information of the target model, including model structure and parameters. Black-box attacks only know the input and output of the DNN model, and thus usually need greater perturbation to make a successful attack. We will introduce several most commonly used adversarial attack methods, including 4 white-box attacks (FGSM, JSMA, CW, and Deepfool) and 2 black-box attacks (Local Search Attack and One Pixel Attack).

2.2.1. FGSM

Fast Gradient Sign Method (FGSM) is proposed by Goodfellow et al. Goodfellow, Shlens and Szegedy (2014). They found the derivative of the model to the input, then used a sign function to get the gradient direction. The perturbation is obtained by multiplying by one step in the sign direction and the final adversarial sample $\hat{x}$ is

$$\hat{x} = x + \epsilon sign(\nabla_x \boldsymbol{J}(x, y)), \tag{2}$$

where x is the benign sample, ϵ is the step, $\boldsymbol{J}$ is the loss function of the trained model, and y is the ground truth label of x. FGSM is "fast" because it only updates the perturbation once, and the result is not guaranteed to be minimal.

2.2.2. JSMA

Jacobian-based Saliency Map Attack Papernot, McDaniel, Jha, Fredrikson, Celik and Swami (2016a) (JSMA) is a kind of targeted attack that utilizes the adversarial saliency between the input feature and the output only by modifying a small number of input pixels. There are 3 steps in JSMA, calculating the forward derivative, calculating the adversarial saliency map, and adding perturbation. The forward derivative is obtained by deriving the output of the model's last layer for a given input. Then the attacker calculates a saliency map based on the Jacobian matrix that reflects the impact of different input features on the output. The final perturbation is added on the top-2 dominating features (2 pixels) of the input.

2.2.3. CW

Carlini & Wagner Attack Carlini and Wagner (2017b) (CW) is a kind of optimization based attack method. The idea is treating the input as a variable, train the input (add perturbation) with fixed model parameters, maximize the distance between the ground truth label and the model output, and minimize the distance between the target label and model output.

2.2.4. Deepfool

Deepfool Attack Moosavi-Dezfooli et al. (2016) aims to find the shortest distance from the normal sample to the classification hyperplane and cross the boundary to generate adversarial samples. For two-classification problem, the object of perturbation $\mathbf{r}$ is

$$\begin{aligned} &argmin_{\mathbf{r}} \|\mathbf{r}\|_2 \\ &s.t.\ sign(f(x_0 + \mathbf{r})) \neq sign(f(x_0)), \end{aligned} \tag{3}$$

where $f(\cdot)$ is the model output on the classification hyperplane and x_0 is the normal sample. This objective function can be solved iteratively to obtain the smallest perturbation. For the multi-classification problem and the detailed derivation process, we refer readers to Moosavi-Dezfooli et al. (2016).

2.2.5. Local Search Attack

Narodytska et al. Narodytska and Kasiviswanathan (2017) proposed a simple black box attack with no internal knowledge of the target network. Their attack uses local search to construct an approximation of the model gradient, then uses it to guide the generation of perturbation. The object is to minimize the output confidence of benign sample I on label c. There are mainly two steps in the local search. First, obtain the confidence $f(\tilde{I})$ with the perturbation added on the last iteration, where $\tilde{I}$ is the perturbed image, and sorts the confidence scores in descending order; Second, the perturbation is added depending on whether the attack is successful. This process will end after the specified number of iterations or the attack successes.

2.2.6. One Pixel Attack

One Pixel Attack achieves misclassification by only modifying one pixel of the input image Su, Vargas and Sakurai (2019). The object is

$$\begin{aligned} &maximize\ f_{adv}(x + e(x)) \\ &subject\ to\ \|e(x)\|_0 \leq 1, \end{aligned} \tag{4}$$

where x is the normal sample, $e(x)$ is the perturbation, and f_{adv} is the model confidence on the target adversarial label.

To obtain the best pixel location and perturbation amplitude, Su et al. Su et al. (2019) used differential evolution. The perturbation is represented by a 5-element vector, which is the x-y axis coordinates and the disturbance amplitude of each RGB channel. We refer readers to Su et al. (2019) for details.

2.3. Adversarial Defense

Adversarial defense seeks to intervene in the prediction process of DNNs, encompassing training, forward propagation, and input data, with the objective of ensuring accurate predictions on adversarial samples. It covers a wide range of approaches, including adversarial training Shaham et al. (2018), knowledge distillation Papernot et al. (2016b), model pruning Wang, Wang, Ye, Zhao and Lin (2018), gradient masking Taran, Rezaeifar, Holotyak and Voloshynovskiy (2019), and input regularization.

Adversarial training involves incorporating adversarial samples into the training process. This method has proven effective on small datasets like CIFAR-10 and MNIST, but it comes at a significant computational cost and may reduce accuracy on larger datasets Kurakin, Goodfellow and Bengio (2016). Ongoing research is dedicated to enhancing training efficiency and model robustness Shafahi, Najibi, Ghiasi, Xu, Dickerson, Studer, Davis, Taylor and Goldstein (2019).

Other strategies, such as knowledge distillation, aim to diminish the magnitude of network gradients that adversaries can exploit Papernot et al. (2016b); Bai, Zhao and Wen (2023). Gradient masking techniques introduce randomization Taran et al. (2019) and regularizers Nayebi and Ganguli (2017) to obstruct back-propagation. Some approaches offer provable protection against adversarial samples by constraining perturbations within an ϵ-ball Madry, Makelov, Schmidt, Tsipras and Vladu (2017); Wong and Kolter (2018). However, they support a limited range of ϵ values and may not readily adapt to larger DNN architectures Shan, Ding, Wenger, Zheng and Zhao (2022).

In summary, current adversarial defense mechanisms fall short in providing adequate protection against adversarial attacks.

2.4. Adversarial Sample Detection

Recent literature Aldahdooh, Hamidouche, Fezza and Déforges (2022); Drenkow, Fendley and Burlina (2022); Chen, Weng, Deng, Luo, Lan and Tian (2021) has highlighted several distinctions between adversarial samples and normal samples. The sensitivity of them encompass a range of attributes, spanning from differences in the input space to the model's logit outputs. Their differences are demonstrated in terms of distance to decision boundaries, statistical properties, prediction probabilities, and gradient.

Statistical properties are associated with the distribution and manifold of training data, e.g., Grosse et al. Grosse et al. (2017) utilize a kernel-based two-sample test to gauge the disparities in the statistical properties by the maximum mean discrepancy. Prediction probabilities based methods hypothesis that the softmax outputs can be used to detect abnormality. Aigrain et al. Aldahdooh et al. (2022) train a simple binary classifier based on clean and adversarial softmax outputs to identify malicious inputs. Gradient based method Lust and Condurache (2020) calculates the gradient at each layer, with respect to the predicted class of the victim model. Then a detector is used to classify clean and adversarial samples. In the input space, Zhou et al. Zhang et al. (2020) and Nesti et al. Nesti, Biondi and Buttazzo (2021) have demonstrated that adversarial perturbations are not robust and can be exposed by overriding new perturbations. Cohen et al. Cohen et al. (2020) have found that training samples have a strong relationship with clean test samples of the same category, while this link is much weaker for adversarial samples.

Agarwal et al. Agarwal, Goswami, Vatsa, Singh and Ratha (2021) have proposed using an ensemble of experts to detect adversarial samples by combining complementary features from distinct sources. Abusnaina et al. Abusnaina et al. (2021) have proposed a latent neighborhood graph for adversarial sample detection, which describes the local manifold around the testing sample based on a reference database. Wang et al. Wang, Dong, Sun, Wang and Zhang (2019) have taken advantage of the fact that adversarial samples are sensitive to tiny changes in the decision boundary to detect adversarial samples by creating a group of mutated models with mutation operators and then detecting adversarial samples based on the consistency of the outputs. Xie et al. Xie, Wang, Zhang, Ren and Yuille (2017) utilize random resizing and random padding to mitigate the effect of adversarial samples. Magnet Meng and Chen (2017) incorporates detector networks and a reformer network, to differentiate clean and adversarial samples, while the adversarial ones will be further rectified by the reformer network.

The most similar previous work to our proposed GGT is Model Mutation Testing Wang et al. (2019) (MMT), which also leverages multiple models to detect adversarial samples. However, our work is significantly different from MMT since we generate diverse models with the guidance of relational graphs, while MMT generates models based on random perturbations, such as adding random noise to the neural weights or randomly reversing the neural outputs. Experimental results have also shown that our method exceeds MMT in both detection cost and accuracy.

Most current detection methods are designed and evaluated based on a single model, which is unreliable against adaptive attacks. Literature Aldahdooh et al. (2022); He et al. (2017); Carlini and Wagner (2017a); Tramer et al. (2020) investigating adversarial detection methods reports that most detection methods show significant drawbacks when attacked with adaptive settings. In this arms race, the attacking side Bryniarski et al. (2021) is also researching algorithms for evading detection and has breached some of the most advanced methods Bryniarski et al. (2021).

It is worth noting that most of the failed detection algorithms utilize features around a single model, and multi-model-based methods, such as MMT, are not on the list of

failing methods. We argue that a multi-model-based method is a promising countermeasure against adaptive attacks.

3. Graph-Guided Testing

Our Graph-Guided Testing (GGT) for adversarial sample detection consists of three steps, graph generation, diverse model generation, and label change statistics. Given a DNN model, we first generate relational graphs with a certain number of nodes and edges according to the predefined AD. We then generate light-weighed DNN models based on the relational graphs, which are further integrated to detect adversarial samples.

The detection is based on the sensitivity of the models to different kinds of samples. For a normal sample, GGT generated models tend to give consistent outputs; for an adversarial sample, there will be a variety of labels in the outputs. We use the label change rate to capture this difference, and the threshold is determined by normal samples. Thus our method can be used to detect unknown adversarial samples.

3.1. Graph Generation

There are a lot of characteristics, such as degree distribution, Average Degree (AD), Average Shortest Path Length (ASPL), average cluster coefficient, etc., to represent the graph structure. In this work, we mainly focus on undirected graphs, and simply set that each node has the same degree since the degree of a normal DNN internal neurons is equal. In this case, AD can be used to control the sparsity of relational graphs, so as to determine the pruning rate of the corresponding DNN model.

According to the definition of relational graph You et al. (2020), we define an undirected graph $G = (V, E)$ by a node set $V = v_1, ..., v_n$ and an edge set $E \subseteq \{(v_i, v_j)|v_i, v_j \in V\}$. Each node v_i has an associated feature x_i, which is a tensor representing multi-channel data. The number of nodes N must be less equal than the number of channels in the DNN model. Suppose there are M links in a relational graph with N nodes (self-connection is not considered here), the AD of the graph is calculated by

$$k = \frac{2M}{N}. \tag{5}$$

The *pruning rate* (or *sparsity ratio*) of the DNN model is then defined as

$$\theta = 1 - \frac{k}{N}. \tag{6}$$

For the generated graph, we further regulate its ASPL to obtain a series of relational graphs with different ASPL. The object of regulating ASPL is given by

$$D = \frac{2}{N \times (N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i-1} d(i, j), \tag{7}$$

where N is the total number of nodes in the graph, $d(\cdot)$ is the distance function that measures the shortest path length between nodes v_i and v_j. The regulation of ASPL follows the rewiring process proposed by Xuan et al. Xuan, Li and Wu (2009). The details are listed in Algorithm 1. As shown in Fig. 2, we first randomly choose two nodes in the graph, and then exchange their neighbors. Note that the selected four nodes must be different. For example, in Fig. 2, node v_i, its neighbor node v_w, node v_j, and its neighbor v_v are selected. We delete the edges between v_i and v_w, v_j and v_v, and then create new edges to connect v_i and v_v, and v_j and v_w. This ensures the degree of each node is fixed during the regulation. After new edges are created, we make the following check: If the graph is connected after rewiring and the current ASPL D is closer to the target ASPL interval $[L_l, L_h)$, we accept the update, otherwise, we reject it, and then return to the node selection step. Once the current ASPL satisfies $D \in [L_l, L_h)$, we record the relational graph. Once the maximum number of rewiring times is achieved, we stop the rewiring process. Note that, we randomly initialize a large number of relational graphs (all the nodes have equal degrees) and repeat the above process for each of them, in order to get enough relational graphs with their ASPL falling into various predefined intervals.

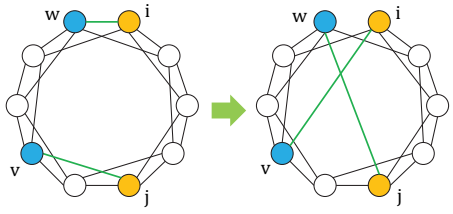


Figure 2: Optimization process of regulating ASPL.

Algorithm 1 ASPL regulation of relational graph

Input: A randomly generated graph $G_i^{(0)}$ with fixed number of nodes N and initial ASPL D_i, maximum number of exchanges m, target ASPL boundary $[L_l, L_h)$, and exchange counter n.
Output: A graph G_o within required ASPL range $[L_l, L_h)$.

repeat
  Select two nodes v_i and v_j with their respective neighbors v_w and v_v that v_i, v_j, v_w, and v_v must be 4 different nodes, $n = 0$;
  Exchange v_i and v_j neighbors, $n = n + 1$, obtain $G_i^{(n)}$;
  if G is not connected **then**
    Go to Step.2;
  end if
  Calculate the ASPL of $G_i^{(n)}$ and $G_i^{(n-1)}$ with $D = \frac{2}{N\times(N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i-1} d(v_i, v_j)$;
  if $D_i \le L_l$ and $D(G_i^{(n)}) > D(G_i^{(n-1)})$ **then**
    Save the exchange;
  else if $D_i > L_h$ and $D(G_i^{(n)}) < D(G_i^{(n-1)})$ **then**
    Save the exchange;
  else
    Reject the exchange;
  end if
  if $L_l \le D(G_i^{(n)}) < L_h$ **then**
    return graph $G_i^{(n)}$;
  end if
until $n > m$
return fail to generate required graph;

3.2. Diverse Model Generation

Pruned DNN models are created using the adjacency matrix of the above generated relational graph. For a graph $G = \{V, E\}$ with N nodes, the elements of its adjacency

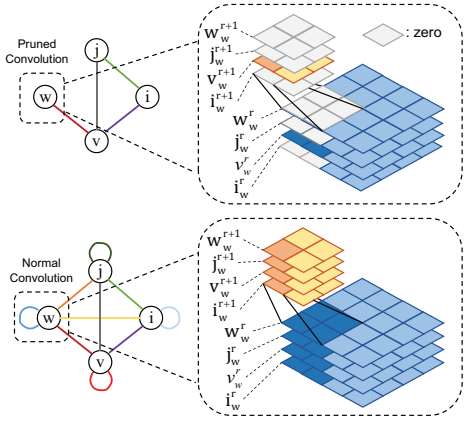


Figure 3: Graph mapping in different situations.

matrix $A_{N\times N}$ indicate whether pairs of vertices are adjacent or not in the graph. If there is an edge between nodes v_i and v_j, i.e., there is message exchange between them, we have $A[i][j] = 1$, otherwise $A[i][j] = 0$. Considering a DNN with complex components, such as convolution layers and pooling layers, we define the node feature as a tensor $\boldsymbol{X}_i$ and the message exchange as convolutional operator, which is defined as

$$\boldsymbol{X}_i^{(r+1)} = AGG(\sum_{j\in N(i)} (A[i][j] \times \boldsymbol{W}_{i,j}^{(r)}) * \boldsymbol{X}_j^{(r)}), \quad (8)$$

where $AGG(\cdot)$ is an aggregation function, $*$ is the convolutional operator, $N(i)$ is the neighbors of node v_i, and v_r represents the rth round of message exchange. A Pooling can be regarded as a special convolution, a pooling message exchange can be expressed as

$$\boldsymbol{X}_i^{(r+1)} = AGG(max_m(\boldsymbol{X}_j^{(r)})), \quad (9)$$

where $max(\cdot)$ is the max pooling operator and m is the pooling kernel size. As shown in Fig. 3, in a relational graph, a round of message exchange means the extracted feature is passed to the next layer. In a normal convolutional layer, the convolution kernel will traverse the entire feature map, and in the corresponding relational map, it is expressed as one node linked to all the other nodes. There could be redundant edges that exist in this kind of fully connected relational graph, and reducing the redundancy in the graph may in turn prune the DNN model. Using the generated sparse relational graph, we obtain a pruned model after mapping. The pruning is achieved by using a masking matrix multiplying the original convolution weight, where the reserved weight is multiplied by 1, and the removed weight is multiplied by 0. After pruning, the number of feature maps is constant, but the values of those feature maps corresponding to the masked weights are all equal to 0.

Different layers of a DNN model are generally different in size. We allow the same node in different layers or within the same layer to have different dimensions. Specifically, suppose one layer has n_c channels, and the total number of nodes in the relational graph is N. Then there are (n_c mod N) nodes that have $\lfloor n_c/N \rfloor + 1$ channels. According to the number of channels that a node is assigned, we create a mask matrix as described above, and multiply it by the weights corresponding to the channels. That is, the weights of the corresponding edges in the relational graph are retained.

We will retrain these pruned models to improve their accuracy. Note that the masked weights can be removed in post-process, thus the actual parameters and computational cost of the pruned models can be significantly reduced. Here, we argue that, when we use the retrained pruned models to detect adversarial samples, it's testing time, rather than training time, that really matters in the real application. Therefore, it is quite important to simplify DNN models by using our graph-guided pruning technology to reduce the memory and computational cost in the online detection phase.

3.3. Label Change Statistics

As shown in Fig. 4, adversarial samples are generally near the original model decision boundary, while pruned models have different boundaries from the original model. On the one hand, pruned models have a small probability of giving wrong outputs for some normal samples (because of pruning and retraining); on the other hand, their boundaries may also be more sensitive to adversarial samples. Because the model decision boundaries are complex, and an adversarial sample that crosses the boundary of the original model may also cross different boundaries of the pruned models, i.e., outputting multiple different labels. We thus can use Label Change Rate (LCR) to measure the diversity Wang et al. (2019) of the pruned models' outputs, which is defined as

$$\eta = \frac{\sum_{s\in S} E(f(x), s(x))}{|S|}, \quad (10)$$

where S is the set of pruned models, x is the input, $f(x)$ is the output of the original model, $s(x)$ is the output of the pruned model, $|S|$ is the size of set S, and $E(\cdot)$ is defined as

$$E(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{otherwise}, \end{cases} \quad (11)$$

which counts the number of times that the classification results in the pruned models are different from that of the original model. We assume that, given a certain number of pruned models, normal samples and adversarial samples will have significantly different scores. In Section 4, we further show that our method can be accelarated by Sequential Probability Ratio Testing Wald (2004) (SPRT).

4. Experiments

We implemented our GGT for adversarial sample detection with Pytorch (*version* 1.6.0) based on Python (*version* 3.7) Our experiment focuses on answering the following research questions:

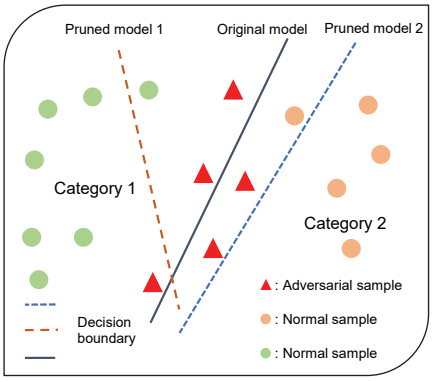


Figure 4: Decision boundary of the original DNN model and the pruned models generated from relational graphs.

1. Ablation analysis for hyperparameters in GGT, including AD and ASPL.
2. How good GGT is at detecting adversarial samples?
3. Compared with other detection methods, what is the advantage of GGT?

4.1. Experiment Settings

4.1.1. Datasets and Models

We evaluate our approach on three image datasets: CIFAR10 Krizhevsky, Hinton et al. (2009), SVHN Netzer, Wang, Coates, Bissacco, Wu and Ng (2011), and Oxford 102 Flower Nilsback and Zisserman (2008). They are widely used in the evaluation of DNN testing frameworks Wang et al. (2019); Xie, Ma, Juefei-Xu, Xue, Chen, Liu, Zhao, Li, Yin and See (2019); Ma, Zhang, Sun, Xue, Li, Juefei-Xu, Xie, Li, Liu, Zhao et al. (2018a). Note that here we do not use the MNIST dataset, since the adopted model for this dataset (e.g., LeNet) is too small, and thus the diversity of the relational graph could be largely limited. The image size of CIFAR10 and SVHN datasets is $32 \times 32 \times 3$ and Flower dataset is $224 \times 224 \times 3$. ResNet18, VGG16, and VGG19 are adopted for CIFAR10, SVHN, and Flower, with their accuracy equal to 93.03%, 95.63%, and 88.71%, respectively. These models are independently trained in our experiment without fine adjustment of hyperparameters, thus the accuracy will fluctuate within a small range compared with the public models.

4.1.2. Graph-Guided Pruned Models Generation

For all datasets, we set the number of nodes in each relational graph to 64 and use ASPL to guide our relational graph generation. The AD, which determines the pruning rate of the generated DNN model, will affect the range of ASPL (too high or too low AD can not make ASPL reach the required length). We thus set the AD $k = 3$ with the corresponding *pruning rate* (or *sparsity ratio*) equal to 95.3% (1-3/64), under which the accuracy drop does not exceed 5%. The ASPL ranges from 3 to 15 and is divided with a step of 2 (3-5, 5-7, 7-9, 9-11, 11-13, 13-15) to explore the relationship between adversarial sample detection accuracy and ASPL. There are 100 graphs for each segment. All the pruned models are retrained with accuracy at least equal to 85.34%, 91.35%, and 85.28%, for CIFAR10, SVHN, and Flower, respectively.

Table 1
Number of generated adversarial samples.

	Normal	WL	FGSM	JSMA	CW	DF	OP	LS
CIFAR10	1000	697	1000	1000	572	1000	714	647
SVHN	1000	1000	1000	1000	1000	1000	196	309
Flower	1000	482	548	\	173	126	86	\

4.1.3. Adversarial Sample Generation

We evaluate the performance of GGT in detecting adversarial samples generated by six typical attack methods described in Sec.2.2, including four white-box and two black-box attacks. We also consider samples that are wrongly labeled as a form of adversarial examples. Unless stated otherwise, all adversarial samples are generated using the original model. Table 1 summarizes the number of samples selected for evaluation from each type of adversarial attack. We exclude attack methods that generate too few successful samples during the test.

4.2. Evaluation Metrics

We evaluate GGT in the following three ways:

1. Diversity Score (DS): The key to our GGT method is that normal sample and adversarial sample have significantly different LCR which is calculated by Eq. (10). Here, we define DS as

$$d_l = \frac{l_a}{l_n}, \tag{12}$$

   where l_a and l_n are the average LCR of adversarial and normal samples, respectively.
2. AUROC, FP, and RC: GGT works based on the diversity score of normal samples. In order to select the best threshold and see how well it works, we calculate the Area Under the ROC (AUROC) curve, False Positive (FP) rate, and ReCall (RC), to demonstrate whether the coefficient score of two groups of models is able to distinguish adversarial samples.
3. FLOPs: FLOPs represent the required computing resources for detection.

4.3. Result

We first generate the graphs of different AD varied in K={2,3,4,5,6,7,8,9,16,24,32,40,48,56,64}, with the degree of each node in a graph is set the same, equal to $k \in K$. For each k, we randomly create 5 graphs, generate their corresponding DNNs, then train and record the average testing accuracy. Note that the training of pruned models is time-consuming, but in the application, it's the model size that really determines the detection efficiency and resource consumption. In the case of detecting one sample on CIFAR10, our GGT only needs 1,622.75M FLOPs.**RQ1: Graphs with lower AD and ASPL generate more accurate models.** The relationship between the average accuracy

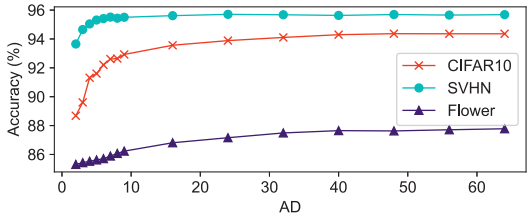


Figure 5: Relationship between AD of the relational graph and the classification accuracy of the pruned DNN model.

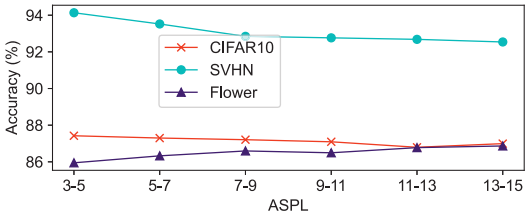


Figure 6: Relationship between ASPL of the relational graph and the classification accuracy of the pruned DNN model.

of DNNs and the AD of the relational graphs is depicted in Fig. 5. The graph shows that the accuracy of DNNs tends to increase as the relational graph becomes denser. This trend is more significant when AD increases from 2 to 8, while the accuracy does not change much as AD further increases from 8 to 64. Furthermore, even when the relational graph becomes very sparse, e.g., $k = 2$ (with the pruning rate equal to 1-2/64≈97%), the decrease in accuracy is not significant. In this case, about 97% of parameters can be pruned out from the original DNN, with the accuracy loss only equaling 4.35% and 2.00% on CIFAR10 and SVHN, respectively. As suggested inWang et al. (2019), mutated models of accuracy over 90% of the original model could be chosen for adversarial detection. Therefore, we argue that all the graph-guided pruned models, with the relational graph keeping connected (no matter what AD is), can be used for adversarial detection, due to their relatively low accuracy loss. To improve the detection efficiency and study the effect of ASPL on the classification accuracy of the pruned model, we set $k = 3$ in the rest of the paper since it is easier to obtain relational graphs of a wide range of ASPL in this case.

For CIFAR10 and SVHN, the shorter the ASPL of the relational graph is, the higher the classification accuracy of the model, but for Flower data, the opposite is true. Meanwhile, the pruned models of corresponding optimal ASPL are better candidates to design GGT, since they can better capture the difference between adversarial samples and normal samples. We record the mean classification accuracy for each set of models with AD $k = 3$, as shown in Fig. 6. The accuracy slightly decreases as ASPL increases for CIFAR10 and SVHN datasets, while the Flower dataset shows the opposite trend. It is likely that the graph with a smaller ASPL has a shorter information propagation path between neurons. A shorter information propagation path is more conducive to the extraction of features in small-resolution images while the opposite for high-resolution images.

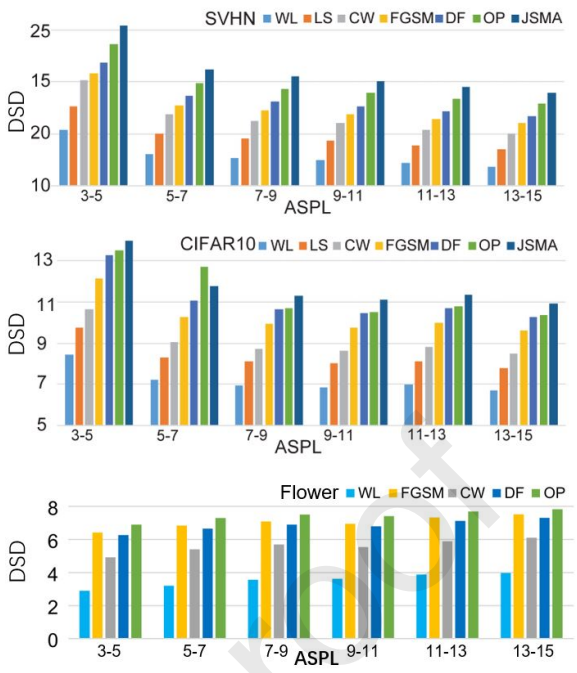


Figure 7: DS of GGT (with various ASPL).

To further explore the effect of ASPL on adversarial detection, we calculate the DS of adversarial samples using 6 groups of models with different ASPL, i.e., the average LCR of adversarial samples divided by that of normal samples. Note that we treat normal samples that are wrongly labeled (WL) by the original model as a kind of adversarial sample Wang et al. (2019), and the attacker will not add any perturbation. The results are summarized in Fig. 7. In summary, for all 6 groups of pruned models, adversarial samples have much higher LCR than normal samples (more than 5 times for CIFAR10 and 10 times for SVHN), indicating that these generated models are highly sensitive to adversarial samples. Compared with other ASPLs, the pruned models with ASPL 3-5 show the largest DS between normal samples and adversarial samples on CIFAR10 and SVHN. Though the largest DS on Flower is ASPL 13-15, the difference is lesser than 2. Thus we use ASPL 3-5 in our following experiment.

RQ2: GGT achieves competitive performance among the SOTA detection algorithms. We compare GGT with Local Intrinsic Dimensionality Ma, Li, Wang, Erfani, Wijewickrema, Schoenebeck, Song, Houle and Bailey (2018b) (LID) method and Mahalanobis Distance Lee, Lee, Lee and Shin (2018) (MD) method, which utilizes a single characteristic of adversarial samples for detection. Ma et al. assumed that the LID of adversarial samples is significantly higher than that of normal samples, and they can be detected by estimating the LID of inputs. Lee et al. proposed that pretrained features can be fitted well by a class-conditional Gaussian distribution, under which they measure the Mahalanobis distance with respect to the closest class-condition distribution to distinguish adversarial samples. The average

Table 2
Comparison with other single classifier methods (%).

	GGT	LID	MD
CIFAR10	**93.61**	82.06	91.10
SVHN	**94.46**	86.70	90.85
Flower	**91.62**	79.06	83.06

detection accuracy is shown in Table 2. GGT obtains the best performance among the most advanced detection algorithms.

The detection speed of GGT can be further improved by SPRT, which is also used in MMT. SPRT is generally faster than using a fixed number of models. The SPRT probability ratio is calculated using $pr = \frac{p_1^z(1-p_1)^{n-z}}{p_0^z(1-p_0)^{n-z}}$, where n is the total number of used models, z is the number of models that output different labels, $p_0 = \eta_h + \sigma$, and $p_1 = \eta_h - \sigma$. Here, η_h is a threshold determined by the LCR of normal samples. To determine whether normal and adversarial samples can be distinguished by η_h, we calculate the Area Under the Receiver Operating Characteristic (AUROC). We obtain the LCR values of a set of normal samples and a set of adversarial samples using Equation (10) and then calculate AUROC for every possible η_h. The x-axis and y-axis of the AUROC represent the true positive rate and false positive rate obtained by using the threshold η_h, respectively. The closer AUROC is to 1, the better the threshold is. σ is a relax scale, which means that neither hypothesis (normal or adversarial) can be denied in the region $(\eta_h - \sigma, \eta_h + \sigma)$. We then calculate the *deny* LCR DL, and *accept* LCR AL using

$$\begin{aligned} DL &= log_e \frac{\beta}{1-\alpha}, \\ AL &= log_e \frac{1-\beta}{\alpha}, \end{aligned} \tag{13}$$

where α and β denote the probability of false positive and false negative, respectively. During testing, the input is sent to the original model as well as the GGT generated models. We can obtain the dynamic LCR based on Equation (10), denoted by pr, which is compared to the *deny* LCR and *accept* LCR. If $pr \leq DL$, the input is considered as an adversarial sample, while if $pr \geq AL$, it is considered as a normal sample.

It is important to note that SPRT also has a maximum model limitation, which we set to 100. After being enhanced by SPRT, the detection accuracy and the number of used models are shown in Fig. 8. We compare GGT with MMT in detail, as MMT is also a multi-model based detection algorithm that uses SPRT. The parameters of MMT are set to default according to the released code. It is worth mentioning that in the original paper of MMT, the model number limitation is set to 500, which we believe is too large for real applications Wang et al. (2019). The mutation rate we use for MMT is 0.007 for all datasets to obtain the best performance. Our baseline is NAI mutation operators, which is the best performer among their proposed methods. We

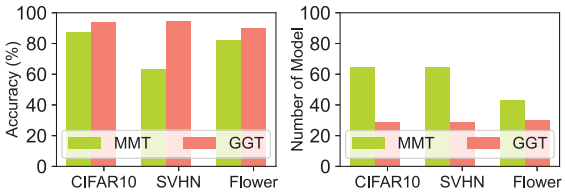


Figure 8: Detection accuracy and the number of used models in GGT and MMT after using SPRT.

Table 3
Resource consumption of GGT and MMT.

Dataset	Method	FLOPs of one model	Average used models
CIFAR10	GGT	56.68M	30.04
	MMT	1112.07M	53.60
SVHN	GGT	33.50M	28.50
	MMT	627.16M	77.16

observe that GGT uses only 30.04, 28.50, and 29.37 models on average across all attacks, while MMT needs 42.82, 64.65, and 43.37 models on CIFAR10, SVHN, and Flower, respectively. It is worth noting that a single model in GGT is much smaller than the single mutated model in MMT. This result suggests that GGT is much more efficient than MMT. Meanwhile, GGT achieves an average detection accuracy of 93.61%, 94.46%, and 91.62% on CIFAR10, SVHN, and Flower while MMT only has 74.74%, 44.79%, and 82.3% respectively, indicating that GGT is also more effective than MMT in detecting adversarial samples. The detailed FLOPs are shown in Table 3. On CIFAR10, GGT uses less than 20 models compared to MMT, and requires around 5% of the FLOPs of the original model.

GGT leverages the sensitivity of adversarial inputs situated in close proximity to decision boundaries, thereby prompting an important query regarding its susceptibility in scenarios where malicious samples exhibit a considerable separation from these decision boundaries. Driven by the premise that inputs characterized by high confidence levels tend to occupy the central regions of the decision space, thereby being distant from decision boundaries, our investigation aims to assess the performance of GGT when confronted with adversarial samples exhibiting high confidence levels. Specifically, we curate a set of samples where the model's confidence in an incorrect label assignment exceeds 90%. As shown in Table 4, the results illustrate a sustained high level of detection accuracy across various datasets and adversarial attack scenarios. Notably, the Flower dataset is not amenable to inclusion in this experiment due to the limited availability of high-confidence samples. This phenomenon indicates the sensitivity of adversarial inputs to variations in the DNN model, which facilitates them to traverse decision boundaries. Though they are in the center of the decision space in one model, they will cross decision boundaries in other models when adjustments to the model architecture are introduced.

Table 4
Detection accuracy under high confidence attacks (%).

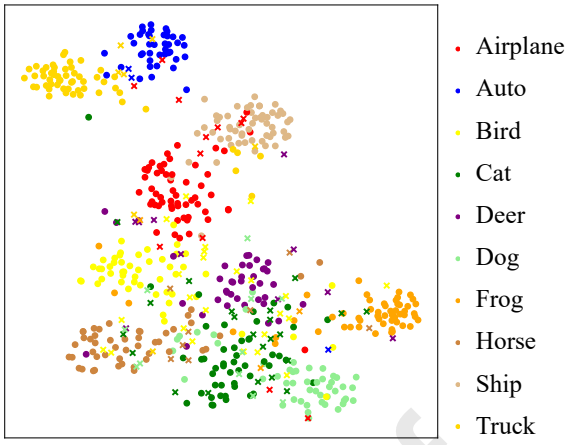
	SVHN	CIFAR10
FGSM	91.0	96.4
JSMA	97.8	98.0
CW	97.4	91.8
DF	99.6	98.4
OP	96.4	99.4
LS	95.0	87.4

In particular, we have also conducted calculations to determine the AUROC score using LCR as the primary feature to differentiate between adversarial samples and normal ones. The results of these calculations are summarized in Table 5, with the best result marked in bold. It is evident that the GGT model tends to produce higher AUROC scores than the MMT model. However, the pruned models with ASPL 3-5 perform the best on CIFAR10 and SVHN datasets in detecting adversarial samples while maintaining a good balance between true positives and false positives. For the Flower dataset, the optimal ASPL value is 13-15. This finding is consistent with our expectations as this set of models exhibits higher classification accuracy and can better identify the difference between adversarial samples and normal samples, as indicated by the results of RQ2. We present the True Positive Rate (TP) and False Positive Rate (FP), as detailed in Table 6. Notably, our GGT methodology consistently exhibits high accuracy, resulting in consistently low FP values across various attack types and datasets. This substantiates the effectiveness of the proposed method.

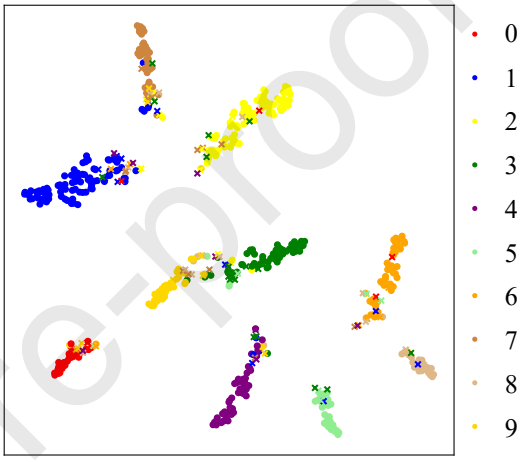
In Table 6, we undertake a comparative analysis between GGT and random pruning, aiming to identify the superior performer. To ensure a meaningful and equitable comparison, we set the pruning rate of random pruned models to 95%, mirroring the corresponding pruning rate applied to GGT.

Across various attacks, GGT consistently exhibits superior performance in terms of TP and FP. On CIFAR-10, GGT demonstrates an average TP rate of 96.6%, surpassing random pruning by 0.5%. Furthermore, in the context of FP, GGT attains an average rate of 9.5%, outperforming random pruning by a substantial margin of 8.3%. On SVHN, the FP of GGT is slightly higher than random by 3.6% on average. However, this marginal difference in FP is counterbalanced by a substantial advantage in TP, where GGT surpasses random pruning by 7.5%. On FLOWER, the average TP and FP of GGT are 95.1% and 12.3%, exceeding random pruning by 5.1% and 5.3%, respectively. In summary, GGT consistently outperforms random pruning models. It excels in terms of true positives and false positives, thereby establishing its effectiveness as a robust and reliable approach for adversarial detection.

To further elucidate the positioning of adversarial samples to the decision boundary, we project the high-dimensional features of the models' final layer into a two-dimensional space by t-SNE Van der Maaten and Hinton (2008); Chen, Gao, Liu, Peng and Wang (2023). The Flower dataset is excluded from this visualization due to the limited number of samples within a single category. As shown in Figure 9, we collect 500 clean samples and 100 adversarial samples, represented by dots and crosses, respectively. Clean samples are grouped by their respective categories, while adversarial samples consistently manifest themselves along the periphery of each category cluster. This observation serves as compelling evidence of the efficacy of GGT to the diverse data distributions it encounters.



(a) CIFAR10.



(b) SVHN.

Figure 9: t-SNE visualization of clean and adversarial samples, where the dots are normal samples, and the crosses are adversarial samples.

We also compare GGT with random pruned models. For equality, we set the pruning rate the same as GGT As shown in Tab**,

RQ3: Compared with other detection methods, GGT is more resistant to circumvention under adaptive attacks and shows robustness under transferable attacks. When using adaptive attacks, i.e., the attacker knows the model parameters and defense strategy, Bryniarski et al.

Table 5
AUROC results (%) for GGT and MMT.

Attack	MMT	GGT with different ASPL					
		3-5	5-7	7-9	9-11	11-13	13-15
CIFAR10							
FGSM	92.14	**98.35**	98.10	98.09	98.15	98.20	98.05
JSMA	97.88	**99.26**	99.21	99.18	99.11	99.18	99.14
CW	91.70	**97.29**	96.97	96.82	96.91	97.04	96.80
DF	97.28	**99.17**	98.82	98.83	98.83	98.85	98.78
OP	**99.77**	99.35	99.03	98.98	99.00	99.02	98.99
LS	93.99	**96.44**	96.18	96.13	96.21	96.19	95.85
WL	92.38	**93.90**	93.35	93.05	93.06	93.32	93.01
SVHN							
FGSM	87.61	97.90	97.88	97.97	**98.14**	98.11	98.07
JSMA	96.25	99.31	99.27	99.27	99.28	**99.32**	99.24
CW	89.22	**99.46**	99.40	99.41	99.44	99.44	99.40
DF	96.59	**99.74**	99.68	99.66	99.69	99.68	99.66
OP	94.97	**98.90**	98.67	98.57	98.67	98.67	98.48
LS	93.30	**98.42**	98.27	98.26	98.28	98.25	98.11
WL	89.02	90.81	90.75	90.51	90.77	**91.07**	90.73
FLOWER							
FGSM	81.92	96.55	97.22	97.54	97.31	97.86	**98.02**
JSMA	\	\	\	\	\	\	\
CW	80.00	90.96	92.64	93.02	92.91	93.67	**93.94**
DF	82.63	95.93	96.55	96.81	95.69	97.01	**97.22**
OP	97.73	98.46	98.61	98.83	98.74	98.97	**99.03**
LS	\	\	\	\	\	\	\
WL	74.11	76.62	76.79	77.46	78.64	79.42	**80.39**

Table 6
TP/FP compared with random pruning (%).

	CIFAR10		SVHN		FLOWER	
	random	GGT	random	GGT	random	GGT
FGSM	94.3/15.9	**96.2**/**8.5**	81.6/**5.1**	**92.5**/8.4	86.7/13.4	**94.7**/**9.9**
JSMA	97.9/15.9	**97.9**/**9.2**	84.1/**3.8**	**97.7**/8.4	\	\
CW	94.5/21.5	**98.4**/**8.6**	90.7/**5.1**	**97.1**/8.4	93.1/24.1	**93.5**/**22.5**
DF	**98.4**/15.9	97.5/**8.5**	97.0/**5.1**	**99.4**/8.4	90.2/17.0	**93.5**/**9.9**
OP	**99.7**/19.4	96.9/**8.9**	91.8/**3.5**	**94.9**/8.4	90.2/16.1	**98.7**/**7.0**
LS	91.5/18.0	**92.4**/**13.0**	85.3/**5.1**	**93.7**/8.4	\	\

(2021) and Carlini and Wagner (2017a) have demonstrated that many advanced detection algorithms that use a single indicator or classifier can be bypassed with specially designed attacks. To circumvent GGT, an attacker would need to fool all the generated models simultaneously.

We evaluated the performance of GGT under adaptive attacks using the adaptive Opt attack Ma et al. (2018b). However, we used multiple models instead of a single detector. The goal of the adaptive sample was to fool all the models and make them predict the same label since GGT requires a consensus among all the models. To generate adaptive samples, we used $\hat{x} = x + \varepsilon sign(\sum_{i=1}^{M} \nabla_x J_i(x, y_{target}))$, where y_{target} is the target label, x is the clean sample, J_i is the loss function of the ith model, M is the number of models, and $\hat{x}$ is the adaptive adversarial sample. We randomly selected 20 models out of 100 generated models to generate adaptive attack samples. We found that all 20 models had zero accuracies on the generated adversarial samples. However, only 26.58% of the generated samples were predicted to the target label, indicating that the attack could not make all models predict the same label since the generated models had diverged decision boundaries.

To accelerate the detection speed, we adapted SPRT to detect adaptive samples. The detection accuracy is shown in Table 7. We found that GGT could still detect most of the adaptive samples, with detection accuracies of 95.6% and 92.2% on CIFAR10 and SVHN, respectively. However, we did not generate adaptive samples for the Flower dataset due to the computing resource limitation as the weight of a single model was larger than 1GB.

Table 7
Detection accuracy under adaptive and transferable attacks (%).

	Adaptive	ILA	FIA
CIFAR10	95.6	91.4	100
SVHN	92.2	94.2	95.1
Flower	-	94.9	100

Another concern on multi-model-based detection is transferable attacks. Because transferable attacks are designed to fool multiple models. We then test GGT under white-box condition with the state-of-the-art transferable attacks, i.e., Intermediate Layer Attack Huang, Katsman, He, Gu, Belongie and Lim (2019) (ILA) and Feature Importance-aware Attack Wang, Guo, Zhang, Liu, Qin and Ren (2021) (FIA). As shown in Table 7, GGT still detects more than 91% of the adversarial samples under transferable attacks. It implies that GGT is robust to transferable attacks.

5. Limitation and Discussion

First, the graph-DNN mapping only works on CNN currently. Therefore, GGT may not be suitable for other kinds of DNN models. Nevertheless, CNN is still the mainstream deep learning method in computer vision, thus GGT is still valuable in these applications. We also focus on establishing graph-DNN mapping for other DNN models to make our GGT suitable for them in the near future.

Second, we only validate GGT on image data, while DNN models are also widely used in processing other types of data, e.g., time series. Considering that GGT is based on the diversity of decision boundaries created by different DNN models, it is reasonable to believe that they can also be applied to other kinds of data.

Third, we only explore two graph characteristics, AD and ASPL, to generate diverse DNN models since AD determines the sparsity of a DNN model while ASPL is related to information exchanging efficiency. Indeed, our proposed GGT based on these two characteristics has both higher efficiency and effectiveness than MMT. There are certainly many other characteristics that could be further explored in the future.

Fourth, GGT requires the original model and generated models both maintain high accuracy, low-accuracy models will lead to poor detection performance. Actually, when we consider the threats of adversarial samples, we suppose the threatened model has been widely used (e.g., face recognition model), so the accuracy must be high. Thus GGT is still valuable in most cases.

6. Conclusion

In this paper, we establish the mapping between DNN architecture and relational graph and then prune a DNN

model guided by the designed relational graph. Using this method, we can prune out more than 95% parameters with only a small loss of accuracy, and find that the accuracy of the pruned model is negatively related to the ASPL of the relational graph. We design Graph-Guided Testing (GGT) for adversarial sample detection, based on the pruned models with small AD and short ASPL in their corresponding relational graphs. The experimental results show that our GGT detects adversarial samples with 93.61%, 94.46%, and 91.62% average accuracy, requiring 30.04, 28.50, and 29.37 pruned models, on CIFAR10, SVHN, and Flower datasets, respectively, much better than the state-of-the-art Model Mutation Testing (MMT) in both accuracy and resource consumption. For high-confidence adversarial samples and transferable attacks, GGT also shows robustness.

CRediT authorship contribution statement

Zuohui Chen: Conceptualization, Methodology, Writing - original draft, Visualization. **Renxuan Wang:** Methodology, Visualization, Investigation. **Jingyang Xiang:** Methodology, Investigation. **Yue Yu:** Supervision, Resources. **Xin Xia:** Supervision, Resources. **Shouling Ji:** Supervision, Resources. **Qi Xuan:** Supervision, Resources, Formal analysis. **Xiaoniu Yang:** Supervision, Resources.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61973273, and by the Zhejiang Provincial Natural Science Foundation of China under Grant LR19F030001.

References

Abusnaina, A., Wu, Y., Arora, S., Wang, Y., Wang, F., Yang, H., Mohaisen, D., 2021. Adversarial example detection using latent neighborhood graph, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7687–7696.

Agarwal, A., Goswami, G., Vatsa, M., Singh, R., Ratha, N.K., 2021. Damad: Database, attack, and model agnostic adversarial perturbation detector. IEEE Transactions on Neural Networks and Learning Systems .

Akhtar, N., Ahamad, M.V., 2021. Graph tools for social network analysis, in: Research Anthology on Digital Transformation, Organizational Change, and the Impact of Remote Work. IGI Global, pp. 485–500.

Aldahdooh, A., Hamidouche, W., Fezza, S.A., Déforges, O., 2022. Adversarial example detection for dnn models: A review and experimental comparison. Artificial Intelligence Review 55, 4403–4462.

Aneja, S., Aneja, N., Islam, M.S., 2018. Iot device fingerprint using deep learning, in: Proceedings of the International Conference on Internet of Things and Intelligence System, IEEE. pp. 174–179.

Athalye, A., Engstrom, L., Ilyas, A., Kwok, K., 2018. Synthesizing robust adversarial examples, in: Proceedings of the International Conference on Machine Learning, PMLR. pp. 284–293.

Bai, T., Luo, J., Zhao, J., Wen, B., Wang, Q., 2021. Recent advances in adversarial training for adversarial robustness. arXiv preprint arXiv:2102.01356 .

Bai, T., Zhao, J., Wen, B., 2023. Guided adversarial contrastive distillation for robust students. IEEE Transactions on Information Forensics and Security .

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al., 2016. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 .

Bryniarski, O., Hingun, N., Pachuca, P., Wang, V., Carlini, N., 2021. Evading adversarial example detection defenses with orthogonal projected gradient descent. arXiv preprint arXiv:2106.15023 .

Carlini, N., Wagner, D., 2017a. Adversarial examples are not easily detected: Bypassing ten detection methods, in: Proceedings of the Workshop on Artificial Intelligence and Security, pp. 3–14.

Carlini, N., Wagner, D., 2017b. Towards evaluating the robustness of neural networks, in: Proceedings of the Symposium on Security and Privacy, IEEE. pp. 39–57.

Carlini, N., Wagner, D., 2018. Audio adversarial examples: Targeted attacks on speech-to-text, in: Proceedings of the Security and Privacy Workshops, IEEE. pp. 1–7.

Chen, F., Wang, Y.C., Wang, B., Kuo, C.C.J., 2020. Graph representation learning: a survey. APSIPA Transactions on Signal and Information Processing 9.

Chen, M., Gao, W., Liu, G., Peng, K., Wang, C., 2023. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7766–7775.

Chen, X., Weng, J., Deng, X., Luo, W., Lan, Y., Tian, Q., 2021. Feature distillation in deep attention network against adversarial examples. IEEE Transactions on Neural Networks and Learning Systems .

Cohen, G., Sapiro, G., Giryes, R., 2020. Detecting adversarial samples using influence functions and nearest neighbors, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 14453–14462.

Drenkow, N., Fendley, N., Burlina, P., 2022. Attack agnostic detection of adversarial examples via random subspace analysis, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 472–482.

Fornito, A., Zalesky, A., Bullmore, E., 2016. Fundamentals of brain network analysis. Academic Press.

Goodfellow, I.J., Shlens, J., Szegedy, C., 2014. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 .

Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P., 2017. On the (statistical) detection of adversarial examples. arXiv preprint arXiv:1702.06280 .

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 770–778.

He, W., Wei, J., Chen, X., Carlini, N., Song, D., 2017. Adversarial example defense: Ensembles of weak defenses are not strong, in: 11th USENIX workshop on offensive technologies (WOOT 17).

Huang, Q., Katsman, I., He, H., Gu, Z., Belongie, S., Lim, S.N., 2019. Enhancing adversarial example transferability with an intermediate level attack, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4733–4742.

Krizhevsky, A., Hinton, G., et al., 2009. Learning multiple layers of features from tiny images .

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. Communications of the ACM 60, 84–90.

Kurakin, A., Goodfellow, I., Bengio, S., 2016. Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236 .

Lee, K., Lee, K., Lee, H., Shin, J., 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. Advances in Neural Information Processing Systems 31.

Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., Zhu, J., 2018. Defense against adversarial attacks using high-level representation guided denoiser, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1778–1787.

Liu, Q., Wen, W., 2021. Model compression hardens deep neural networks: A new perspective to prevent adversarial attacks. IEEE Transactions on Neural Networks and Learning Systems .

Lust, J., Condurache, A.P., 2020. Gran: An efficient gradient-norm based detector for adversarial and misclassified examples. arXiv preprint arXiv:2004.09179 .

Ma, L., Zhang, F., Sun, J., Xue, M., Li, B., Juefei-Xu, F., Xie, C., Li, L., Liu, Y., Zhao, J., et al., 2018a. Deepmutation: Mutation testing of deep learning systems, in: Proceedings of the International Symposium on Software Reliability Engineering, IEEE. pp. 100–111.

Ma, S., Liu, Y., 2019. NIC: Detecting adversarial samples with neural network invariant checking, in: Proceedings of the Network and Distributed System Security Symposium.

Ma, X., Li, B., Wang, Y., Erfani, S.M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M.E., Bailey, J., 2018b. Characterizing adversarial subspaces using local intrinsic dimensionality, in: International Conference. on Learning Representations.

Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-sne. Journal of machine learning research 9.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A., 2017. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 .

Meng, D., Chen, H., 2017. Magnet: a two-pronged defense against adversarial examples, in: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp. 135–147.

Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P., 2016. Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 2574–2582.

Narodytska, N., Kasiviswanathan, S., 2017. Simple black-box adversarial attacks on deep neural networks, in: Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops, IEEE. pp. 1310–1318.

Nayebi, A., Ganguli, S., 2017. Biologically inspired protection of deep networks from adversarial attacks. arXiv preprint arXiv:1703.09202 .

Nesti, F., Biondi, A., Buttazzo, G., 2021. Detecting adversarial examples by input transformations, defense perturbations, and voting. IEEE Transactions on Neural Networks and Learning Systems .

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y., 2011. Reading digits in natural images with unsupervised feature learning .

Nilsback, M.E., Zisserman, A., 2008. Automated flower classification over a large number of classes, in: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, IEEE. pp. 722–729.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A., 2016a. The limitations of deep learning in adversarial settings, in: Proceedings of the European Symposium on Security and Privacy, IEEE. pp. 372–387.

Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A., 2016b. Distillation as a defense to adversarial perturbations against deep neural networks, in: Proceedings of the Symposium on Security and Privacy, IEEE. pp. 582–597.

Riyaz, S., Sankhe, K., Ioannidis, S., Chowdhury, K., 2018. Deep learning convolutional neural networks for radio identification. IEEE Communications Magazine 56, 146–152.

Shafahi, A., Najibi, M., Ghiasi, M.A., Xu, Z., Dickerson, J., Studer, C., Davis, L.S., Taylor, G., Goldstein, T., 2019. Adversarial training for free! Advances in Neural Information Processing Systems 32.

Shaham, U., Yamada, Y., Negahban, S., 2018. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. Neurocomputing 307, 195–204.

Shan, S., Ding, W., Wenger, E., Zheng, H., Zhao, B.Y., 2022. Post-breach recovery: Protection against white-box adversarial examples for leaked dnn models, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 2611–2625.

Su, J., Vargas, D.V., Sakurai, K., 2019. One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation 23, 828–841.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1–9.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R., 2013. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 .

Tan, C.L., Chiew, K.L., Yong, K.S., Abdullah, J., Sebastian, Y., et al., 2020. A graph-theoretic approach for the detection of phishing webpages. Computers & Security 95, 101793.

Taran, O., Rezaeifar, S., Holotyak, T., Voloshynovskiy, S., 2019. Defending against adversarial attacks by randomized diversification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11226–11233.

Tian, J., Zhou, J., Li, Y., Duan, J., 2021. Detecting adversarial examples from sensitivity inconsistency of spatial-transform domain. arXiv preprint arXiv:2103.04302 .

Tramer, F., Carlini, N., Brendel, W., Madry, A., 2020. On adaptive attacks to adversarial example defenses. Advances in Neural Information Processing Systems 33, 1633–1645.

Vivek, B., Baburaj, A., Babu, R.V., 2019. Regularizer to mitigate gradient masking effect during single-step adversarial training., in: CVPR Workshops, pp. 66–73.

Wald, A., 2004. Sequential analysis. Courier Corporation.

Wang, J., Dong, G., Sun, J., Wang, X., Zhang, P., 2019. Adversarial sample detection for deep neural network through model mutation testing, in: Proceedings of the International Conference on Software Engineering, IEEE. pp. 1245–1256.

Wang, S., Wang, X., Ye, S., Zhao, P., Lin, X., 2018. Defending dnn adversarial attacks with pruning and logits augmentation, in: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), IEEE. pp. 1144–1148.

Wang, Z., Guo, H., Zhang, Z., Liu, W., Qin, Z., Ren, K., 2021. Feature importance-aware transferable adversarial attacks, in: Proceedings of the IEEE/CVF International Conference. on Computer Vision, pp. 7639–7648.

Wong, E., Kolter, Z., 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope, in: International conference on machine learning, PMLR. pp. 5286–5295.

Xie, C., Wang, J., Zhang, Z., Ren, Z., Yuille, A., 2017. Mitigating adversarial effects through randomization. arXiv preprint arXiv:1711.01991 .

Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J., See, S., 2019. Deephunter: A coverage-guided fuzz testing framework for deep neural networks, in: Proceedings of the International Symposium on Software Testing and Analysis, pp. 146–157.

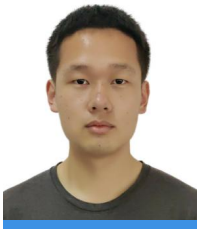
Xuan, Q., Li, Y., Wu, T.J., 2009. Optimal symmetric networks in terms of minimizing average shortest path length and their sub-optimal growth model. Physica A: Statistical Mechanics and its Applications 388, 1257–1267.

You, J., Leskovec, J., He, K., Xie, S., 2020. Graph structure of neural networks, in: Proceedings of the International Conference on Machine Learning, PMLR. pp. 10881–10891.

Yu, S., Zhao, M., Fu, C., Zheng, J., Huang, H., Shu, X., Xuan, Q., Chen, G., 2019. Target defense against link-prediction-based attacks via evolutionary perturbations. IEEE Transactions on Knowledge and Data Engineering .

Zhang, W.E., Sheng, Q.Z., Alhazmi, A., Li, C., 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. ACM Transactions on Intelligent Systems and Technology 11, 1–41.

Zhou, Q., Zhang, R., Wu, B., Li, W., Mo, T., 2020. Detection by attack: Detecting adversarial samples by undercover attack, in: European Symposium on Research in Computer Security, Springer. pp. 146–164.

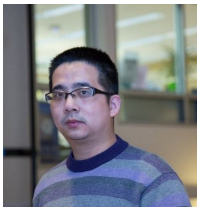


Zuohui Chen received the B.S. degree in automation from the Zhejiang University of Technology, Hangzhou, China, in 2019. He is working toward the Ph.D. degree in control theory and engineering from the Zhejiang University of Technology. His current research interests include computer vision, AI application, and AI security.

Renxuan Wang received the B.S. degree in automation from the Anhui Polytechnic University, Wuhu, China, in 2019. He is working toward the Master degree in control theory and engineering from the Zhejiang University of Technology. His current research interests include computer vision and AI security.

Jingyang Xiang was born in Ningbo, Zhejiang, China in 1999. He is an undergraduate majoring in electrical engineering and automation in the College of Information Engineering, Zhejiang University of Techonology. He is currently a member of Intelligence Vision, Signal&Network(IVSN) group of Institute of Cyberspace Security, Zhejiang University of Techonology. His research interests include model pruning and quantization.

Yue Yu is an associate professor in the College of Computer at National University of Defense Technology (NUDT). He received his Ph.D. degree in Computer Science from NUDT in 2016. He has won Outstanding Ph.D. Thesis Award from Hunan Province. His research findings have been published on ICSE, FSE, ASE, TSE, MSR, IST, ICSME, ICDM and ESEM. His current research interests include software engineering, data mining and computer-supported cooperative work.



Xin Xia is the director of the software engineering application technology lab, Huawei, China. Prior to joining Huawei, he was an ARC DECRA Fellow and a lecturer at Monash University, Australia. Xin received his Ph.D in computer science from Zhejiang University in 2014. To help developers and testers improve their productivity, his current research focuses on mining and analyzing rich data in software repositories to uncover interesting and actionable information. More information at: https://xinxia.github.io.

Shouling Ji is a ZJU 100-Young Professor in the College of Computer Science and Technology at Zhejiang University and a Research Faculty in the School of Electrical and Computer Engineering at Georgia Institute of Technology (Georgia Tech). He received a Ph.D. degree in Electrical and Computer Engineering from Georgia Institute of Technology, a Ph.D. degree in Computer Science from Georgia State University, and B.S. (with Honors) and M.S. degrees both in Computer Science from Heilongjiang University. His current research interests include Data-driven Security and Privacy, AI Security and Big Data Analytics.

Qi Xuan received the BS and PhD degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively. He was a Post-Doctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University, from 2008 to 2010, respectively, and a Research Assistant with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2010 and 2017. From 2012 to 2014, he was a Post-Doctoral Fellow with the Department of Computer Science, University of California at Davis, CA, USA. He is a member of the IEEE and is currently a Professor with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. His current research interests include network science, graph data mining, cyberspace security, machine learning, and computer vision.

Xiaoniu Yang is currently a chief scientist with the Science and Technology on Communication Information Security Control Laboratory, Jiaxing, China. He is also an Academician of Chinese Academy of Engineering and a Fellow of the Chinese Institute of Electronics. He published the first software radio book in China (X. Yang, C. Lou, and J. Xu, Software Radio Principles and Applications, Publishing House of Electronics Industry, 2001 (in Chinese)). His current research interests are software-defined satellite, big data for radio signals, and deep learning-based signal processing.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

CRediT authorship contribution statement

Zuohui Chen: Conceptualization, Methodology, Writing - original draft, Visualization. Renxuan Wang: Methodology, Visualization, Investigation. Jingyang Xiang: Methodology, Investigation. Yue Yu: Supervision, Resources. Xin Xia: Supervision, Resources. Shouling Ji: Supervision, Resources. Qi Xuan: Supervision, Resources, Formal analysis. Xiaoniu Yang: Supervision, Resource