TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 11/20 pp784–794
DOI: 10.26599/TST.2023.9010057
Volume 29, Number 3, June 2024

DeepSI: A Sensitive-Driven Testing Samples Generation Method of Whitebox CNN Model for Edge Computing

Zhichao Lian* and Fengjun Tian

Abstract: In recent years, Deep Learning (DL) technique has been widely used in Internet of Things (IoT) and Industrial Internet of Things (IIoT) for edge computing, and achieved good performances. But more and more studies have shown the vulnerability of neural networks. So, it is important to test the robustness and vulnerability of neural networks. More specifically, inspired by layer-wise relevance propagation and neural network verification, we propose a novel measurement of sensitive neurons and important neurons, and propose a novel neuron coverage criterion for robustness testing. Based on the novel criterion, we design a novel testing sample generation method, named DeepSI, which involves definitions of sensitive neurons and important neurons. Furthermore, we construct sensitive-decision paths of the neural network through selecting sensitive neurons and important neurons. Finally, we verify our idea by setting up several experiments, then results show our proposed method achieves superior performances.

Key words: neuron sensitivity; Layer-wise Relevance Propagation (LRP); neural network verification; deep learning testing

1 Introduction

Deep Learning (DL) technique has achieved great breakthroughs in many fields and solved many previous tasks, such as image classification[1, 2], recommendation system[3, 4], object detection[5, 6], privacy protection[7, 8], image segmentation[9, 10], blockchain[11, 12], natural language processing[13], as well as edge computing[14, 15]. However, there are still many safety problems in DL. The model is not robust enough, resulting in poor performance of the model in challenging environments. For example, the system cannot correctly identify pedestrians or signal lights in automatic driving in the complex senarios. Therefore, deep learning models need better testing methods to validate the models.

Nowadays, intrusion detection attacks the security of Internet of Things (IoT) and Industrial Internet of Things (IIoT), and many emerging intrusion detection methods (e.g., ASTREAM[16], HAA[17], MDS_AD[18], etc.) based on deep neural networks ensure the security of IIoT or IoT. In IoT and IIoT, there is a need to process large amount of data[19, 20] or use physical sensors[21–23] for communication, which requires privacy protection, data security, or cyber attack issues for data in IoT or IIoT. To ensure the accuracy of detection in IoT, a large number of researchers[24, 25] use edge computing to improve detection performance, but the accuracy and robustness of deep neural networks in edge computing cannot be guaranteed. Therefore, these deep neural networks need efficient testing methods for validation.

In traditional software testing, more attentions are paid to generating the test cases which can cover the codes, and can make the program crash as much as possible. But in the Deep Neural Network (DNN)

• Zhichao Lian and Fengjun Tian are with School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China. E-mail: newlzcts@njust.edu.cn; fengjun_tian@njust.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2023-02-03; revised: 2023-05-11; accepted: 2023-06-02

testing, given a test case, the model can always get a prediction result. This makes it difficult to know whether the generated test samples are good enough to cover the codes. According to the idea of coverage in traditional software testing, some researchers proposed the criterion of neuron coverage[26], so that the neurons in the neural network are activated as much as possible. Furthermore, some researchers also proposed other coverage criteria[27], such as k-Multisection Neuron Coverage (KMNC) and Neuron Boundary Coverage (NBC). Subsequently, the researchers develop testing sample generation techniques based on these criteria. However, the existing methods ignore the relationship between input perturbations and output results.

Many researchers have achieved breakthroughs in the field of correlation graph[28, 29], and in this paper we use Layer-wise Relevance Propagation (LRP)[30]. In addition, some previous works[31] have improved the robustness of deep neural networks by adding perturbations to training samples to limit the output boundaries of neurons during training. Inspired by them, we find that different neurons in the models have obvious differences between the input perturbation and the output boudary changes. Therefore, we propose a novel measurement to quantify the sensitivity of neurons on the input perturbation, and design a novel testing samples generation method. Our methods and results can be used for model validation and model lightweight in edge computing. To summarize, the main contributions of this paper are as follows:

(1) We propose a novel neuron coverage criterion, Sensibility-Importance Neuron Coverage (SINC). We quantify the sensitivity of neurons by neural network verification and use the SINC to guide sample generation.

(2) We propose a novel framework DeepSI for generating sensitivity test samples, and propose a sensitivity measurement method. Compared with other State-Of-The-Art (SOTA) methods, our experimental results are better at SINC and keep neuron coverage unchanged.

(3) Experimental results show that sensitive neurons we proposed can explain the idea that the middle convolutional layer is more important in convolutional layers, and the bottom linear layers is more important in linear layer, which is consistent with the findings of other researchers.

2 Related Work

DNN Testing. Different from traditional software development methods, deep learning is a data-driven development method[32, 33]. DeepXplore[26] proposes a measurement named neuron coverage based on differential testing, the DeepXplore seeded unlabeled test inputs and generated new testing samples covering a large number of neurons (i.e., activating them above a customizable threshold), while making the tested DNN generate different behaviors. DeepGauge[27] monitored and measured neuronal behavior, as well as internal network connections, at various levels of granularity. DeepTest[34] brought neuron coverage to self-driving technology, verifying that changes in neuron coverage are statistically correlated with changes in the actions of self-driving cars. TensorFuzz[35] introduced coverage-guided fuzzing technology in traditional software into DNN testing technology. And the coverage is checked using a fast approximate nearest neighbor algorithm. DeepCT[36] proposed a set of combined testing criteria for DNN, and a series of test coverage criteria, as well as corresponding test case generation methods. According to the path of activated neurons in the model, DeepInspect[37] defined metrics such as Neuron Activation Probability Vector Distance (NAPVD) and average bias. DeepPath[38] drew on the concept of path in traditional software engineering, considering a single neuron in the model as a node, and the neuron connection between different layers as a path. Moreover, three paths coverage metrics, l-SAP, l-OAP, and l-FSP are proposed. DeepCon[39] believed that the existing coverage criteria only use the output of neurons to determine the activation state of neurons, while ignoring the size of the connection weights issued by neurons. Therefore, the prediction results of DNN should be determined by the output of neurons. The connection weights are jointly determined. In DeepHunter[40], five state-of-the-art trial criteria, three existing and one novel selection strategies were employed. After initializing the seed queue, extensible seed selection strategy, extensible metamorphic mutation, and extensible testing criterion, it generates the new seed queue which has high coverage such as KMNC and Neuron Coverage (NC).

DNN verification. Neural network verification can be divided into complete verifiers and incomplete verifiers. Early complete verifiers are based on

Satisfiability Modulo Theory (SMT)[41–43]. In incomplete verifiers of neural network verification, convex relaxation is used to relax the nonlinear activation function, using linear constraints instead of nonlinear activation functions[44, 45]. BaBSR[46] shrinks the boundary by separating the activation boundary of Rectified Linear Unit (ReLU) on it. GPUPoly[47] is the GPU extended version of DeepPloy[48] as an incomplete verifier. Crown[49] is a verifier for determining the functional relationship between the output and the input through backward propagation.

3 Method

3.1 Overview of DeepSI

The interpretability of DL model has been greatly improved recently. The decision strategy of the neural network has become more and more transparent, and make the neural network more and more credible.

Specifically, Fig. 1 describes the overview of DeepSI. We expound one neuron important analysis method which is based on LRP in DeepSI (c.f. Section 3.2) and one neuron sensitive analysis strategy (c.f. Section 3.3) which is inspired by neural network verification. And we propose a novel testing criterion (c.f. Section 3.4) named SINC which is based on neuron important analysis and neuron sensitive analysis. In Fig. 1, important neurons are represented in green, sensitive neurons are represented in red, and yellow neurons represent both important and sensitive neurons.

3.2 Neuron important analysis

Definition 1 (DNN) A multilayer DNN contains convolution layers, ReLU layers, linear layers, etc. They can be represented by $\langle l_1, l_2, ..., l_n \rangle$, where l_1 is the input layer and l_n is the output layer of the DNN. $\langle l_2, l_3, ..., l_{n-1} \rangle$ are the hidden layers in DNN. The output of the previous layer is the input of the next layer.

In this work, we focus on a classification work $f: X \to Y$, X is the set of inputs and Y is set of all classes. For the given input $x \in X$, $f(x) = \arg\max_y (O_n(x))$, where $O_n(x)$ is the n-dimensional vector output by the output layer l_n of the neural network. LRP calculates the relevance score for each pixel in the input image, and that it also explains the score of each neuron in each hidden layer by backward propagation. Therefore, the sum of the relevance scores of each neuron which is in l_i is equal to the sum of the scores in the next layer. In this way, the relationship between the output $f(x)$ and the input of the DNN is shown in the following formula:

$$f(x) = \sum_{i \in l+1} R_i^{(l+1)} = \cdots = \sum_{i \in l} R_i^{(l)} = \sum_i R_i^{(1)} \tag{1}$$

For a specific node where is the neuron k at layer $l+1$, the relevance of this neuron is represented by $R_k^{(l+1)}$. The relevance $R_k^{(l+1)}$ is equal to the sum of the relevance of all neurons related to the neuron k at layer l such that

$$R_k^{(l+1)} = \sum_i R_{i \leftarrow k}^{(l,l+1)} \tag{2}$$

The relevance $R_j^{(n)}$ is back-propagated from the prediction neuron j at the last layer l_n of DNN to all neurons in the first layer l_1, including the input image.

Figure 2 illustrates how LRP calculates the relevance score. In Fig. 2a, we reveal how DNN which is a "black-box" predicts that the image x is "7". And LRP calculates the relevance score layer-by-layer form the

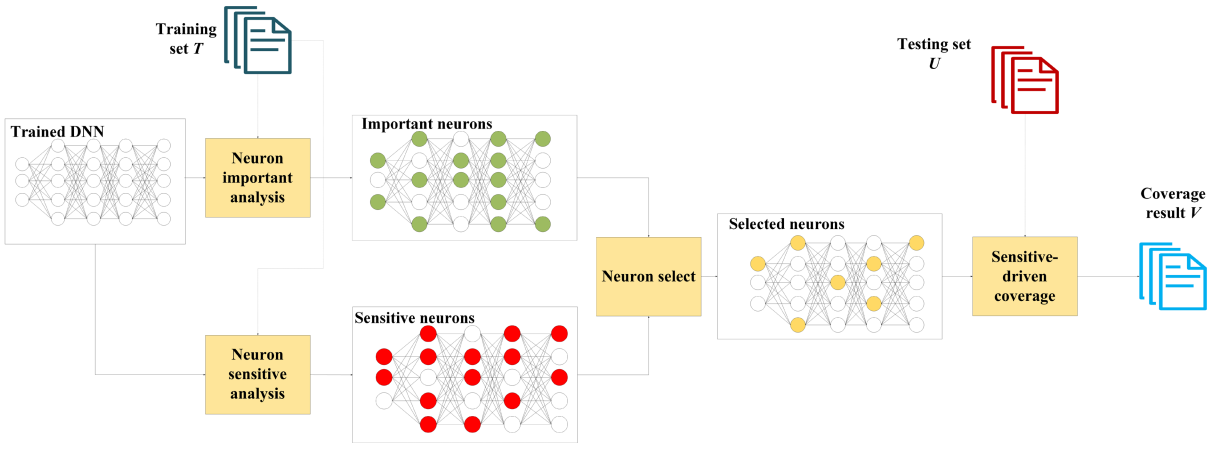
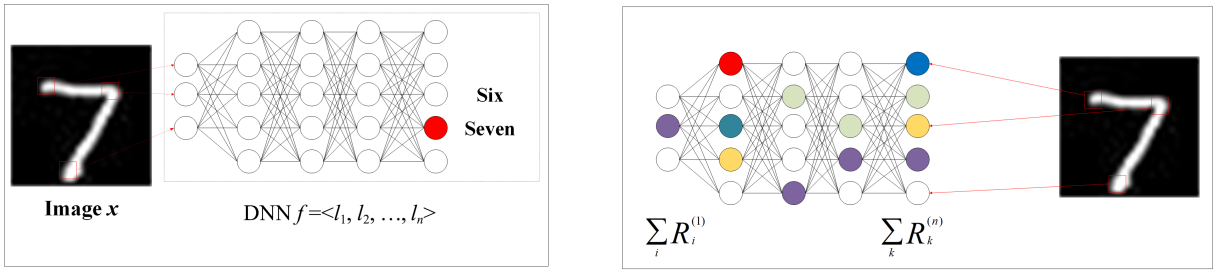


Fig. 1 Workflow of DeepSI for generating sample.



(a) LRP forward **(b) LRP backward**

Fig. 2 Workflow of DeepSI for generating sample.

output Layer l_n to the first Layer l_1 in Fig. 2b. Finally, the neurons and pixels that have an impact on the prediction are recorded. Neurons of different colors represent different degrees of influence on the prediction results.

Different from previous works[50] in which only positive relevance is selected, we think that in DNN, negatively relevant neurons or pixels that affect the prediction results are also important. For a prediction result, if negatively relevant nodes are removed, the prediction of DNN will also have a huge deviation.

Inspired by LRP, we consider neurons or pixels with relevance to be important, and define these neurons or pixels as importance neurons or importance pixels.

Definition 2 (important neuron) Given an input x and a DNN f which is consist of n layers, we define Important Neuron (IN) as a set of neurons which the relevance score is nonzero:

$$\mathrm{IN} = \{(k, l_i) \mid \left(R_k^{(l_i)} > 0\right) \vee \left(R_k^{(l_i)} < 0\right)\} \tag{3}$$

Although there has been works to remove the relevance score of the non-target area in LRP such as Contrastive Layer-wise Relevance Propagation (CLRP)[51] and Softmax-Gradient Layer-wise Relevance Propagation (SGLRP)[52]. We consider the importance of non-target neurons still active. If we delete the corresponding relevance scores of non-target classes, the information obtained will be reduced. So, in this work, we still use LRP as a measure of the importance of neurons.

3.3 Neuron sensitive analysis

Due to the complex senarios and unseen noise, the result of model recognition or detection is normally much lower than the expected level.

Neural network verification calculates the boundaries of each prediction result of the output layer by adding a fixed perturbation to the input image. We consider that in addition to the output layer, each hidden layer neuron in the neural network is also bounded. Therefore, for a classification model or a detection model, given a white box, the reachability range of neurons in its hidden layer is calculated.

Definition 3 (DNN verification) Given an input x, a DNN f and a perturbation ϵ, the output boundaries are represented by the set $\mathrm{OB} = \{\mathrm{OB}_1, \mathrm{OB}_2, ...\}$. $\mathrm{OB}_1, \mathrm{OB}_2, ...$ are calculated by x, f, and ϵ. DNN verification is to find the smallest ϵ such that there is no intersection of output boundaries.

In this part, we introduce a neural network verification method for measuring sensitivity. Although there have been several verification methods[49, 53, 54] of backward propagation, in this paper, we use the neural network verification method based on forward propagation to calculate the output boundary of perturbation test image for lower computational burden.

Verification based on forward propagation. The verification method is named Interval Bound Propagation (IBP)[55]. IBP is originally used to train large and provably robust neural networks to improve model accuracy. Meanwhile, IBP can also be used for model validation. While adding perturbation to the test image x_0, lower boundary for x_0 is $\underline{z_0}(\epsilon) = x_0 - \epsilon$, and upper boundary is $\overline{z_0}(\epsilon) = x_0 + \epsilon$. Through forward propagation, the upper and lower boundaries of the neuron i in the layer k are expressed as

$$\underline{z_{k,i}}(\epsilon) = \min_{\underline{z_{k-1}}(\epsilon) \leqslant z_{k-1} \leqslant \overline{z_{k-1}}(\epsilon)} e_i^{\mathrm{T}} h_k(z_{k-1}) \tag{4}$$

$$\overline{z_{k,i}}(\epsilon) = \max_{\underline{z_{k-1}}(\epsilon) \leqslant z_{k-1} \leqslant \overline{z_{k-1}}(\epsilon)} e_i^{\mathrm{T}} h_k(z_{k-1}) \tag{5}$$

where $h_k()$ is the corresponding neural network hidden layer operation. Specifically, k represents the layer of the neural network, i represents the number of neurons in layer k, e_i is the standard i-th basis vector, and z represents the output of neuron.

However, existing neural networks (e.g., ResNet[56], DenseNet[57], etc.) have a large number of non-linear activation functions such as ReLU. Figure 3 shows the solution of the bounds of the IBP to calculate the ReLU activation function. IBP defines the two blue lines as the upper and lower bounds of this function.

When we use neural network verification methods by forward propagation. We can obtain an upper boundary set $\overline{B}=\left\{\overline{B_1^1}, \overline{B_1^2}, \ldots, \overline{B_i^{l_1}}, \ldots, \overline{B_k^{l_n}}, \ldots, \overline{B_n^{n-1}}, \overline{B_n^n}\right\}$ and a lower boundary set $\underline{B}=\left\{\underline{B_1^1}, \underline{B_1^2}, \ldots, \underline{B_i^{l_1}}, \ldots, \underline{B_k^{l_n}}, \ldots, \underline{B_n^{n-1}}, \underline{B_n^n}\right\}$. By adjusting the perturbation ϵ size to adjust the boundary change ratio α, we can obtain a new set of upper and lower boundaries: $\overline{\text{NewB}}=\left\{\overline{\text{NewB}_1^1}, \overline{\text{NewB}_1^2}, \ldots, \overline{\text{NewB}_i^{l_1}}, \ldots, \overline{\text{NewB}_k^{l_n}}, \ldots, \overline{\text{NewB}_n^{n-1}}, \overline{\text{NewB}_n^n}\right\}$, $\underline{\text{NewB}}=\left\{\underline{\text{NewB}_1^1}, \underline{\text{NewB}_1^2}, \ldots, \underline{\text{NewB}_i^{l_1}}, \ldots, \underline{\text{NewB}_k^{l_n}}, \ldots, \underline{\text{NewB}_n^{n-1}}, \underline{\text{NewB}_n^n}\right\}$.

Definition 4 (sensitive neuron) Given an input x, a perturbation ϵ, a ratio α, and a DNN f, we define the set of Sensitive Neurons (SN) to satisfy

$$\text{SN}=\left\{(i,l_j)\mid(i,l_j)\in\left\{\frac{\overline{\text{NewB}_i^{l_j}}-\underline{\text{NewB}_i^{l_j}}}{\overline{B_i^{l_j}}-\underline{B_i^{l_j}}}>\alpha\right\}\right\} \tag{6}$$

where l_j is the layer j and i is the neuron.

3.4 Sensitivity-driven coverage

Algorithm 1 shows the main algorithm idea. During the generation, the process adds perturbation ϵ to the training samples set (Line 3). After neuron analyzing, DeepSI selects the neurons which are important and Sensitive (Line 10). If the neuron is selected, its corresponding region is perturbed.

Important neurons are the neurons that guarantee correct model predictions, but it does not mean that these neurons are sensitive to perturbations. Sensitive neurons are defined by boundary changes and are more concerned with the sensitivity of neurons to perturbations. Therefore, we need to focus on sensitive neurons while not losing the neurons that play a role in model prediction.

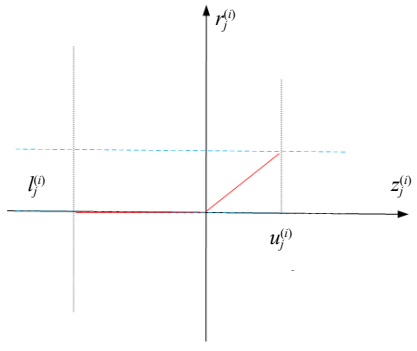


Fig. 3 Workflow of DeepSI for generating sample.

Algorithm 1 DeepSI test generation

Input: ϵ: adversarial perturbation, DNN: target neural network, I: initial tests, T: training tests
Output: SI: SI tests, S: sensitive tests, L: important tests, V: coverage tests
Const: α: a configurable of perturbation control
1: $\alpha \leftarrow \varnothing$
2: $U \leftarrow I$
3: **while** $P \leftarrow \text{Perturb}(T,\epsilon)$ and $N \leftarrow \text{Perturb}(T,\epsilon\times\alpha)$ **do**
4: $\quad\overline{B_i^l}, \underline{B_i^l} \leftarrow$ **Run**(DNN, P)
5: $\quad\overline{\text{NewB}_i^l}, \underline{\text{NewB}_i^l} \leftarrow$ **Run**(DNN, N)
6: $\quad S \leftarrow \text{Sensitive\_Analysis}(\overline{B_i^l}, \underline{B_i^l}, \overline{\text{NewB}_i^l}, \underline{\text{NewB}_i^l}, \alpha)$
7: **end while**
8: $L \leftarrow \text{Important\_Analysis}(T)$
9: **for** $p \in S$ **do**
10: $\quad$**if** isInImportance(p,L) **then**
11: $\quad\quad\text{SI} \leftarrow \text{SI.append}(p)$
12: $\quad$**end if**
13: $\quad V \leftarrow \text{Coverage}(U,\text{SI})$
14: **end for**

Definition 5 (SINC) Given a set of IN and a set of SN, if a test image x could cover neurons which are both sensitive and important neuron, we call these neurons activated. So the formula for SINC is as follows:

$$\text{SINC}=\frac{\text{sum}\left(\left\{\left(i,l_j\right)\mid\left(i,l_j\right)\in\text{SN}\cap\text{IN}\right\}\right)}{\text{total(Neuron)}} \tag{7}$$

The neurons selected by SINC ensure both the anti-perturbation capability and the truth rate of model prediction. So, we use SINC as a novel neuron coverage metric to guide test sample generation.

Sensitivity-driven coverage. To determine the set of sensitive and important neurons in the neural network, the training samples are all put into the neural network. After that, we obtain the set of sensitive and important neurons. Algorithm 2 demonstrates our approach to sample generation using SINC. When the SINC of the newly generated test sample is larger than the original sample (Line 10), the sample is put into the new queue (Line 11).

Algorithm 2 SINC coverage

Input: S: sensitive tests, L: important tests, I: initial tests
Output: V: coverage tests
Const: α: a configurable of perturbation control
1: $U \leftarrow I$
2: **for** $p \in S$ **do**
3: **if** isInImportance(p, L) **then**
4: SI $\leftarrow$ SI.append(p)
5: **end if**
6: **end for**
7: **while** $u \in U$ **do**
8: **for** si $\in$ SI **do**
9: $u' \leftarrow$ Coverage$(u,$ si$)$
10: **if** Coverage_Ratio(u') > Coverage_Ratio(u) **then**
11: $V \leftarrow V$.append(u')
12: **end if**
13: **end for**
14: **end while**

4 Experiment

4.1 Setup

In order to verify the sensitive neurons, we have designed relevant experiments to demonstrate the effectiveness of our method. We select datasets that are widely used in the field of image classification and train models on PyTorch framework with competitive test accuracy that are widely used in the previous works.

MNIST[58] is a handwriting dataset for object recognition, which contains 60 000 training data and 10 000 test data. We train a neural network with two convolution layers and two linear layers named MNIST-model. The MNIST-model achieves an accuracy of 98.64%. Also, we train an LeNet-5 model. And its accuracy is 98.5%, as shown in Table 1.

CIFAR-10[59] is a dataset for object recognition, which contains 50 000 training data and 10 000 test data. We train a VGG-16 model with CIFAR-10 and its accurary is 88.99%.

The existing whitebox testing samples generation frameworks by Coverage-Guided Fuzz (CGF) mainly include DeepXplore, DeepGauge, DeepTest, TensorFuzz, and DeepHunter. In the whitebox testing samples generation frameworks, DeepHunter has proven that it is a SOTA framework by CGF. So, we choose DeepHunter as the comparison framework.

Table 1 Subject dataset and DNN models.

Dataset	DNN model	Number of layers	Accuracy (%)
MNIST	MNIST	9	98.69
	LeNet-5	12	98.50
CIFAR-10	VGG-16	37	88.99

4.2 Result of neuron coverage

NC. The output of a neuron determines whether the current neuron is activated or non-activated. Given an input image, a neuron is activated if its output is above a certain threshold. NC measures the ratio between activated neurons and all neurons in the model as follows:

$$\mathrm{NC} = \frac{|\{n|\forall x \in T, \mathrm{out}(n,x) > t\}|}{|N|} \tag{8}$$

where all neurons of a DNN are represented by the set $N = \{n_1, n_2, ...\}$, all testing samples are represented by the set $T = \{x_1, x_2, ...\}$, and $\mathrm{out}(n,x)$ denotes the output value of neuron n when the input is x.

Table 2 demonstrates that the initial NC of the seed queue is 41.89%, and the NC of testing samples generated by DeepHunter is 84.03% in the MNIST model. NC for generating new test samples through our proposed DeepSI framework is 84.03%. Similar experimental results are also observed in the LeNet-5 model and the VGG-16 model. Under the NC criterion, the test samples we generated did not have another large drop.

4.3 Result of SINC

Table 3 shows the difference of the SINC coverage criteria of the two methods when the hyperparameters α are different. It can be found that the samples generated by our proposed novel framework DeepSI

Table 2 Results of neuron coverage with different strategies.

Dataset	Model	Strategy	NC (%)
MNIST	MNIST	Initial	41.89
		DeepHunter + Prob	84.03
		DeepSI + IBP ($\epsilon = 1 \times 10^{-6}$)	84.03
	LeNet-5	Initial	68.44
		DeepHunter + Prob	92.31
		DeepSI + IBP ($\epsilon = 1 \times 10^{-6}$)	92.31
CIFAR-10	VGG-16	Initial	44.16
		DeepHunter + Prob	60.62
		DeepSI + IBP ($\epsilon = 2 \times 10^{-9}$)	60.62

Table 3 Results of SINC with different strategies.

Dataset	Model	ϵ	Method	SINC (%)								
				$\alpha=0.1$	$\alpha=0.2$	$\alpha=0.3$	$\alpha=0.4$	$\alpha=0.5$	$\alpha=0.6$	$\alpha=0.7$	$\alpha=0.8$	$\alpha=0.9$
MNIST	MNIST	1×10^{-6}	DeepHunter + Prob	41.51	46.17	51.62	50.75	51.54	55.29	56.66	57.43	61.21
			DeepSI + IBP	48.05	51.71	53.42	54.89	55.82	55.82	60.15	60.30	61.25
	LeNet-5	1×10^{-6}	DeepHunter + Prob	39.73	53.12	51.60	56.05	61.84	65.28	64.83	85.91	75.47
			DeepSI + IBP	48.89	55.65	58.01	58.70	62.59	66.97	64.65	78.29	75.77
CIFAR-10	VGG-16	2×10^{-9}	DeepHunter + Prob	0.56	1.11	1.50	4.08	4.09	4.57	4.87	3.97	4.85
			DeepSI + IBP	10.14	10.80	8.49	9.64	9.28	9.40	9.65	9.72	10.65

have higher SINC than the samples generated by DeepHunter in general.

The definition of sensitive neurons is different due to the different values of α. Whether neurons are sensitive to boundary changes depend to perturbations on the test sample. When α is 0.1, the SINC of the test sample generated by our method is 48.05%. But the SINC of testing samples generated by DeepHunter is 41.51%. When α is 0.9, DeepSI is only 0.04% larger than testing samples generated by DeepHunter under SINC criteria. The reason for this phenomenon is that more sensitive neurons are selected when α is 0.9 than when α is 0.1. Neurons differ in their sensitivity to different sizes of perturbations. When α is increased, fewer neurons are judged to be sensitive in the training samples set. And the original testing samples set covers more sensitive neurons. So the coverage of the testing samples set generated by our method has limited improvement. In LeNet-5 model, due to the high SINC of the test samples generated by DeepHunter, the SINC of the samples generated by DeepSI is slightly smaller.

4.4 Sensitive-decision path

Definition 6 (Sensitive-Decision Path (SDP)) Pathways consisting of neurons that are both important and sensitive are selected.

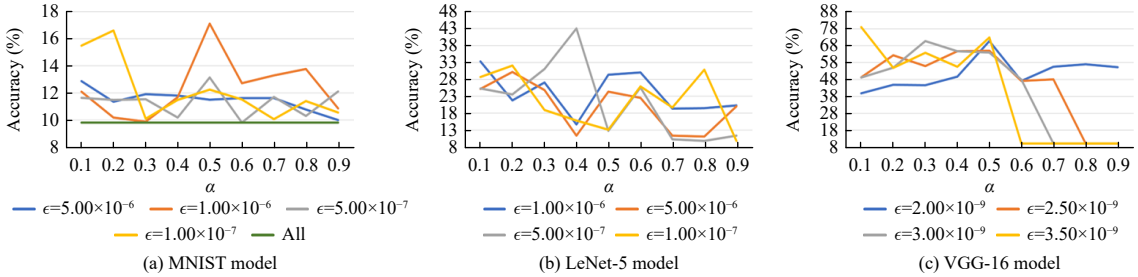
Compared with traditional software testing, DNN predictions are made through neural pathways between neurons. Neural pathways are paths in DNN. Neural pathways are the sequence of code execution in DNN. Among these neural pathways, the paths that affect prediction are the decision paths.

Sensitive neurons are of importance. We introduce sensitive neurons to guide the decision paths to construct novel sensitive-decision paths. These novel sensitive-decision paths not only preserve the accuracy of decision paths, but also eliminate the neurons that are not sensitive to perturbations.

We evaluate the importance of these selected sensitive-decision paths to DNN prediction by designing experiments to verify whether there is a decreasing trend in the accuracy when all sensitive-decision paths are masked in the neural network.

Figure 4 demonstrates that the accuracy of the MNIST model, LeNet-5 model, and the VGG-16 model decreases for different hyperparameters α for different perturbation sizes. Under all selected perturbations, the boundaries of all classes are separated. In Fig. 4, the accuracy of the MNIST model decreases from 98.64% to 9.8% (green line) after masking all importance neurons. When $\alpha=0.1$ and $\epsilon=1\times10^{-6}$, sensitive neurons and importance neurons are calculated, and it is found that after masking the sensitive-decision paths, the recognition rate of the neural network decreases from 98.64% to 12.08%. Experiments have shown that the model is less accurate in classification without sensitive neurons.

As seen in Fig. 4, the models accuracy does not



(a) MNIST model (b) LeNet-5 model (c) VGG-16 model

Fig. 4 Models accuracy after masking sensitive decision paths.

increase linearly when the hyperparameter α increases. This is because the sensitive neurons do not increase linearly.

4.5 Correlation with the middle layer characteristics of neural network

To further evaluate the impact on sensitive-decision paths, a fine-grained evaluation is performed. Previous works have proposed the hypothesis of that the middle convolutional layer is more important, and the bottom linear layer is more important in neural networks, and designed experiments to test it[60]. We consider the same property for sensitive-decision paths, where the importance of the neurons selected in the upper layer is less than that of the neurons selected in the middle layer.

We design experiments to verify the above conjecture by masking neurons that are both important and sensitive in a single hidden layer.

As shown in Table 4, the MNIST model has two convolutional layers and two linear layers. If α=0.1, when we mask the first convolutional layer, the first linear layer model accuracy decreases less, and the second linear layer has a significant effect by masking the second convolutional layer. It is seen through the whole table that the neurons in the sensitive-decision path satisfy the same property that the neurons in the middle convolutional layer are more important than the neurons in the upper convolutional layer, while the underlying linear layer is more important. And it also has the same effect in the LeNet-5 model.

5 Conclusion

This paper proposes a general framework for DNN test sample generation of sensitive neurons. Sensitivity analysis in DeepSI is extensible. We conduct some experiments to show that sensitive neurons are also important in neural networks. Since sensitivity analysis and definitions are in their infancy, we hope that follow-up work will lead to more rigorous sensitivities of neurons. We think that neurons have many properties to explore, where each is with some limitations. The main advantage of sensitive neurons is their interpretability and defense against adversarial examples. We hope that DeepSI can facilitate further discussions on DNN testing.

In DeepSI, we select the IBP method for neural network verification. It provides the boundaries of neurons in DNN. However, the IBP method can only calculate the neuron boundaries of the CNN model. In the future, we will expand it to be applicable to ResNet or transformer. The limitation of current sensitive neurons boundary approximation method is not precise enough. Thus how to calculate the boundary more tight and effective will be an interesting research direction in future.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2021YFF0602104-2).

References

[1] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, Twins: Revisiting the design of spatial attention in vision transformers, arXiv preprint arXiv: 2104.13840, 2021.

[2] Q. Hua, L. Chen, P. Li, S. Zhao, and Y. Li, A pixel-channel hybrid attention model for image processing, *Tsinghua Science and Technology*, vol. 27, no. 5, pp. 804–816, 2022.

[3] S. Wu, S. Shen, X. Xu, Y. Chen, X. Zhou, D. Liu, X. Xue, and L. Qi, Popularity-aware and diverse web APIs recommendation based on correlation graph, *IEEE Trans.*

Table 4 Model accuracy after masking sensitive-decision path ($\epsilon = 1\times10^{-6}$).

Dataset	Model	Layer	Accuracy (%)								
			$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$
MNIST	MNIST	Conv_1	15.28	23.24	16.66	16.45	31.71	20.08	19.08	34.53	22.17
		Conv_2	13.59	19.19	14.21	13.69	32.29	18.63	15.88	36.23	18.19
		Linear_1	98.12	97.03	97.24	97.71	97.00	97.36	97.43	97.30	96.48
		Linear_2	60.73	57.14	57.14	57.77	59.84	59.48	59.74	60.42	59.09
	LeNet-5	Conv_1	96.37	97.61	76.41	54.53	91.59	81.36	43.62	98.08	89.02
		Conv_2	52.00	77.29	66.53	24.43	70.90	71.76	10.43	98.02	88.57
		Linear_1	58.43	48.52	62.39	63.45	84.81	58.17	54.03	45.62	62.26
		Linear_2	82.51	75.27	79.17	73.37	63.61	72.01	61.04	52.11	64.80
		Linear_3	49.71	50.25	49.03	50.80	49.73	49.29	51.35	50.51	49.33

Comput. Soc. Syst., vol. 10, no. 2, pp. 771–782, 2023.

[4] X. Zhou, Y. Li, and W. Liang, CNN-RNN based intelligent recommendation for online medical pre-diagnosis support, *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 18, no. 3, pp. 912–921, 2021.

[5] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, et al., Swin transformer V2: Scaling up capacity and resolution, in *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), New Orleans, LA, USA, 2022, pp. 11999–12009.

[6] J. Sun, X. Jiang, J. Liu, F. Zhang, and C. Li, An anti-recompression video watermarking algorithm in bitstream domain, *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 154–162, 2020.

[7] F. Wang, G. Li, Y. Wang, W. Rafique, M. R. Khosravi, G. Liu, Y. Liu, and L. Qi, Privacy-aware traffic flow prediction based on multi-party sensor data with zero trust in smart city, *ACM Trans. Internet Technol.*, vol. 23, no. 3, pp. 1–19, 2023.

[8] F. Wang, H. Zhu, G. Srivastava, S. Li, M. R. Khosravi, and L. Qi, Robust collaborative filtering recommendation with user-item-trust records, *IEEE Trans. Comput. Soc. Syst.*, vol. 9, no. 4, pp. 986–996, 2022.

[9] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[10] L. Zhang, K. Zhang, and H. Pan, SUNet++: A deep network with channel attention for small-scale object segmentation on 3D medical images, *Tsinghua Science and Technology*, vol. 28, no. 4, pp. 628–638, 2023.

[11] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, A blockchain-enabled deduplicatable data auditing mechanism for network storage services, *IEEE Trans. Emerg. Top. Comput.*, vol. 9, no. 3, pp. 1421–1432, 2021.

[12] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, Blockchain empowered arbitrable data auditing scheme for network storage as a service, *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 289–300, 2020.

[13] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv: 1810.04805, 2018.

[14] F. Wang, L. Wang, G. Li, Y. Wang, C. Lv, and L. Qi, Edge-cloud-enabled matrix factorization for diversified APIs recommendation in mashup creation, *World Wide Web*, vol. 25, no. 5, pp. 1809–1829, 2022.

[15] L. Kong, L. Wang, W. Gong, C. Yan, Y. Duan, and L. Qi, LSH-aware multitype health data prediction with privacy preservation in edge environment, *World Wide Web*, vol. 25, no. 5, pp. 1793–1808, 2022.

[16] Y. Yang, X. Yang, M. Heidari, M. A. Khan, G. Srivastava, M. Khosravi, and L. Qi, ASTREAM: Data-stream-driven scalable anomaly detection with accuracy guarantee in IIoT environment, *IEEE Trans. Netw. Sci. Eng.*, doi: 10.1109/TNSE.2022.3157730.

[17] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K. I. K. Wang, Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system, *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9310–9319, 2022.

[18] L. Qi, Y. Yang, X. Zhou, W. Rafique, and J. Ma, Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0, *IEEE Trans. Ind. Inform.*, vol. 18, no. 9, pp. 6503–6511, 2022.

[19] Z. Cai and Z. He, Trading private range counting over big IoT data, in *Proc. 2019 IEEE 39th Int. Conf. Distributed Computing Systems* (*ICDCS*), Dallas, TX, USA, 2019, pp. 144–153.

[20] L. Kong, G. Li, W. Rafique, S. Shen, Q. He, M. R. Khosravi, R. Wang, and L. Qi, Time-aware missing healthcare data prediction based on ARIMA model, *IEEE/ACM Trans. Comput. Biol. Bioinform.*, doi: 10.1109/TCBB.2022.3205064.

[21] L. Nie, Z. Ning, X. Wang, X. Hu, J. Cheng, and Y. Li, Data-driven intrusion detection for intelligent Internet of vehicles: A deep convolutional neural network-based method, *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2219–2230, 2020.

[22] Y. Xu, Z. Liu, C. Zhang, J. Ren, Y. Zhang, and X. Shen, Blockchain-based trustworthy energy dispatching approach for high renewable energy penetrated power systems, *IEEE Internet Things J.*, vol. 9, no. 12, pp. 10036–10047, 2022.

[23] C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang, A blockchain-based multi-cloud storage data auditing scheme to locate faults, *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2252–2263, 2022.

[24] X. Zhou, X. Yang, J. Ma, and K. I. K. Wang, Energy-efficient smart routing based on link correlation mining for wireless edge computing in IoT, *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14988–14997, 2022.

[25] X. Zhou, X. Xu, W. Liang, Z. Zeng, and Z. Yan, Deep-learning-enhanced multitarget detection for end–edge–cloud surveillance in smart IoT, *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12588–12596, 2021.

[26] K. Pei, Y. Cao, J. Yang, and S. Jana, DeepXplore: Automated whitebox testing of deep learning systems, in *Proc. 26th Symp. on Operating Systems Principles*, Shanghai, China, 2017, pp. 1–18.

[27] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, et al., DeepGauge: Multi-granularity testing criteria for deep learning systems, in *Proc. 2018 33rd IEEE/ACM Int. Conf. Automated Software Engineering* (*ASE*), Montpellier, France, 2018, pp. 120–131.

[28] X. Zhou, W. Liang, K. I. K. Wang, and L. T. Yang, Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations, *IEEE Trans. Comput. Soc. Syst.*, vol. 8, no. 1, pp. 171–178, 2021.

[29] L. Qi, W. Lin, X. Zhang, W. Dou, X. Xu, and J. Chen, A correlation graph based approach for personalized and compatible web APIs recommendation in mobile APP

development, *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 5444–5457, 2023.

[30] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, *PLoS One*, vol. 10, no. 7, p. e0130140, 2015.

[31] H. Zhang, H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D. Boning, and C. J. Hsieh, Towards stable and efficient training of verifiably robust neural networks, arXiv preprint arXiv: 1906.06316, 2019.

[32] R. S. Pressman, *Software Engineering*: *A Practitioner's Approach*. London, UK: Palgrave Macmillan, 2005.

[33] N. B. Ruparelia, Software development lifecycle models, *SIGSOFT Softw. Eng. Notes*, vol. 35, no. 3, pp. 8–13, 2010.

[34] Y. Tian, K. Pei, S. Jana, and B. Ray, DeepTest: Automated testing of deep-neural-network-driven autonomous cars, in *Proc. 2018 IEEE/ACM 40th Int. Conf. Software Engineering* (*ICSE*), Gothenburg, Sweden, 2018, pp. 303–314.

[35] A. Odena and I. Goodfellow, TensorFuzz: Debugging neural networks with coverage-guided fuzzing, arXiv preprint arXiv: 1807.10875, 2018.

[36] L. Ma, F. Juefei-Xu, M. Xue, B. Li, L. Li, Y. Liu, and J. Zhao, DeepCT: Tomographic combinatorial testing for deep learning systems, in *Proc. 2019 IEEE 26th Int. Conf. Software Analysis, Evolution and Reengineering* (*SANER*), Hangzhou, China, 2019, pp. 614–618.

[37] Y. Tian, Z. Zhong, V. Ordonez, and B. Ray, Testing deep neural network based image classifiers, arXiv preprint arXiv: 1905.07831, 2019.

[38] D. Wang, Z. Wang, C. Fang, Y. Chen, and Z. Chen, DeepPath: Path-driven testing criteria for deep neural networks, in *Proc. 2019 IEEE Int. Conf. Artificial Intelligence Testing* (*AITest*), Newark, CA, USA, 2019, pp. 119–120.

[39] Z. Zhou, W. Dou, J. Liu, C. Zhang, J. Wei, and D. Ye, DeepCon: Contribution coverage testing for deep learning systems, in *Proc. 2021 IEEE Int. Conf. Software Analysis, Evolution and Reengineering* (*SANER*), Honolulu, HI, USA, 2021, pp. 189–200.

[40] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, DeepHunter: A coverage-guided fuzz testing framework for deep neural networks, in *Proc. 28th ACM SIGSOFT Int. Symp. on Software Testing and Analysis*, Beijing, China, 2019, pp. 146–157.

[41] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, Reluplex: An efficient SMT solver for verifying deep neural networks, arXiv preprint arXiv: 1702.01135, 2017.

[42] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, Safety verification of deep neural networks, in *Computer Aided Verification*, R. Majumdar and V. Kunčak, eds. Cham, Switzerland: Springer, 2017, pp. 3–29.

[43] R. Ehlers, Formal verification of piece-wise linear feed-forward neural networks, in *Automated Technology for Verification and Analysis*, D. D'Souza and K. N. Kumar, eds. Cham, Switzerland: Springer, 2017, pp. 269–286.

[44] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, Fast and effective robustness certification, in *Proc. 32nd Int. Conf. Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 10825–10836.

[45] G. Singh, T. Gehr, M. Püschel, and M. Vechev, Boosting robustness certification of neural networks, https://openreview.net/forum?id=HJgeEh09KQ, 2018.

[46] R. Bunel, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar, A unified view of piecewise linear neural network verification, in *Proc. 32nd Int. Conf. Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 4795–4804.

[47] C. Müller, F. Serre, G. Singh, M. Püschel, and M. Vechev, Scaling polyhedral neural network verification on GPUs, arXiv preprint arXiv: 2007.10868, 2020.

[48] G. Singh, T. Gehr, M. Püschel, and M. Vechev, An abstract domain for certifying neural networks, *Proc. ACM Program. Lang.*, vol. 3, no. POPL, p. 41, 2019.

[49] H. Zhang, T. W. Weng, P. Y. Chen, C. J. Hsieh, and L. Daniel, Efficient neural network robustness certification with general activation functions, in *Proc. 32nd Int. Conf. Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 4944–4953.

[50] X. Xie, T. Li, J. Wang, L. Ma, Q. Guo, F. Juefei-Xu, and Y. Liu, NPC: Neuron path coverage via characterizing decision logic of deep neural networks, *ACM Trans. Softw. Eng. Methodol*., vol. 31, no. 3, p. 47, 2022.

[51] J. Gu, Y. Yang, and V. Tresp, Understanding individual decisions of CNNs via contrastive backpropagation, in *Computer Vision – ACCV 2018*, C. V. Jawahar, H. Li, G. Mori, and K. Schindler, eds. Cham, Switzerland: Springer, 2018, pp. 119–134.

[52] B. K. Iwana, R. Kuroki, and S. Uchida, Explaining convolutional neural networks using softmax gradient layer-wise relevance propagation, in *Proc. 2019 IEEE/CVF Int. Conf. Computer Vision Workshop* (*ICCVW*), Seoul, Republic of Korea, 2019, pp. 4176–4185.

[53] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C. J. Hsieh, Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers, arXiv preprint arXiv: 2011.13824, 2020.

[54] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C. J. Hsieh, and J. Z. Kolter, Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network robustness verification, arXiv preprint arXiv: 2103.06624, 2021.

[55] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, On the effectiveness of interval bound propagation for training verifiably robust models, arXiv preprint arXiv: 1810.12715, 2018.

[56] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Las Vegas, NV, USA, 2016, pp. 770–778.

[57] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, Densely connected convolutional networks,

in *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*), Honolulu, HI, USA, 2017, pp. 2261–2269.

[58] Y. LeCun and C. Cortes, The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/, 1998.

[59] N. Krizhevsky, H. Vinod, C. Geoffrey, M. Papadakis, and A. Ventresque, The CIFAR-10 dataset, http://www.cs.toronto.edu/kriz/cifar.html, 2014.

[60] J. Ren, M. Li, Z. Liu, and Q. Zhang, Interpreting and disentangling feature components of various complexity from DNNs, arXiv preprint arXiv: 2006.15920, 2020.

Zhichao Lian received the bachelor and master degrees in computer science from Jilin University, Changchun, China, in 2005 and 2008, respectively, and the PhD degree from Nanyang Technological University, Singapore, in 2013. From 2012 to 2014, he was a postdoctoral researcher with Department of Statistics, Yale University, USA. He is currently an associate professor at School of Cyber Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include machine learning, explainable AI, and cyberspace security.

Fengjun Tian received the BS degree in computer science and technology from Nanjing University of Science and Technology, Nanjing, China, in 2021. He is currently pursuing the MS degree at School of Cyber Science and Engineering, Nanjing University of Science and Technology, China. His research interests include software testing and explainable AI.