Chinese Journal of Electronics
vol. 33, no. 4, pp. 948–964, July 2024
https://doi.org/10.23919/cje.2022.00.451

Special Focus on Explainability, Robustness, and Security in AI Systems

RESEARCH ARTICLE

# DeepLogic: Priority Testing of Deep Learning Through Interpretable Logic Units

Chenhao LIN, Xingliang ZHANG, and Chao SHEN

*Faculty of Electronic and Infomation Engineering, Xi'an Jiaotong University, Xi'an 710049, China*

Corresponding author: Chao SHEN, Email: chaoshen@xjtu.edu.cn
Manuscript Received December 27, 2022; Accepted June 5, 2023


**Abstract** — With the increasing deployment of deep learning-based systems in various scenes, it is becoming important to conduct sufficient testing and evaluation of deep learning models to improve their interpretability and robustness. Recent studies have proposed different criteria and strategies for deep neural network (DNN) testing. However, they rarely conduct effective testing on the robustness of DNN models and lack interpretability. This paper proposes a new priority testing criterion, called DeepLogic, to analyze the robustness of the DNN models from the perspective of model interpretability. We first define the neural units in DNN with the highest average activation probability as "interpretable logic units". We analyze the changes in these units to evaluate the model's robustness by conducting adversarial attacks. After that, the interpretable logic units of the inputs are taken as context attributes, and the probability distribution of the softmax layer in the model is taken as internal attributes to establish a comprehensive test prioritization framework. The weight fusion of context and internal factors is carried out, and the test cases are sorted according to this priority. The experimental results on four popular DNN models using eight testing metrics show that our DeepLogic significantly outperforms existing state-of-the-art methods.

**Keywords** — Deep learning testing, Interpretable logic units, Adversarial test, Model interpretability, Defect detection.

**Citation** — Chenhao LIN, Xingliang ZHANG, and Chao SHEN, "DeepLogic: Priority Testing of Deep Learning Through Interpretable Logic Units," *Chinese Journal of Electronics*, vol. 33, no. 4, pp. 948–964, 2024. doi: 10.23919/cje.2022.00.451.

## I. Introduction

Deep learning techniques have been extensively researched and applied in safety-related fields, such as autonomous driving [1], medical diagnosis and treatment [2], air traffic management [3], and face recognition [4]. Existing studies have revealed that deep neural networks (DNNs) are vulnerable to both natural cross-domain samples and specifically designed adversarial examples, causing growing concern regarding widely deployed DNN-based systems. Evaluating the robustness and safety of DNN models before they are applied to real-world applications has become a challenging and important task.

Deep learning testing techniques [5] have been proposed to detect potential defects in DNN models to help guarantee their safety and robustness. Unlike traditional software systems, deep learning systems typically contain DNNs with complex structures, where even a small anomaly in the input data could lead to inappropriate system decisions. A large-scale and reasonable set of test cases is required to achieve adequate and effective deep learning testing. Unfortunately, these test cases are often unavailable and require manual labeling. An alternative approach to prioritizing test cases and testing those that could expose deep learning systemic errors to detect more model defects at an earlier stage could help to reduce testing costs and improve testing effectiveness.

In recent years, research has progressed in DNN priority testing from multiple perspectives, including neuron coverage (NC) [6]–[8] and output probability distribution [9]. However, many existing testing techniques based on the NC measure [10], such as NC, neuron boundary coverage (NBC), and $k$-multisection neuron coverage (KMNC) [7], are not applicable for testing the robustness of DNN

Guest Editor: Prof. Qian WANG, Wuhan University.

models. It can be difficult to distinguish defect test cases, particularly adversarial ones, resulting in a low average defect detection rate, also called average percentage of fault detection (APFD). A more recent approach, namely DeepGini [9], does consider adversarial tests. However, it only considers the absolute value of the output probability as the measurement index, ignoring the offset direction, which results in poor performance in target adversarial attack scenarios. Additionally, existing test prioritization methods usually lack interpretability and fail to build connections between different test cases, which could make for more effective testing.

To address the existing limitations, this study proposed DeepLogic to examine the priority testing of DNNs using interpretable logic units. First, from the perspective of DNN model interpretability, the average channel activation value sorted according to the weight is taken to be the neural unit (channel). The top-$k$ neural units are selected as interpretable logic units to establish the degree of correlation with the model prediction and serve as a measurement index in deep learning tests. Subsequently, a comprehensive priority testing framework can be established. Each class of interpretable logic units and the probability distribution of the model outputs are calculated as context and internal attributes in the test process. Finally, these two attributes are combined and used to generate the test-case set.

To validate the effectiveness of the proposed DeepLogic framework, several experiments were conducted on two popular datasets and four popular DNNs, and seven evaluation metrics, including APFD and risk detection ability (RDA), were adopted. Additionally, three popular adversarial attacks were implemented. The experimental results showed that compared with several existing state-of-the-art (SOTA) deep learning testing approaches, DeepLogic could achieve a higher average fault detection rate of 95%.

The main contributions of this paper are as follows:

- We proposed a priority testing framework for DNN models called DeepLogic that enables the effective generation of natural and adversarial tests.
- From the perspective of model interpretability, we proposed a new logic unit test criterion in which context attributes and internal attributes are integrated to evaluate the capability of defect detection.
- We demonstrated the effectiveness of DeepLogic using both natural and adversarial samples on standard and adversarial training models. DeepLogic achieved greatly improved performance compared with several SOTA testing methods.

The remainder of this paper is organized as follows. In Section II, we introduce the research progress in deep learning test criteria, channel interpretability, and test-case priority. Section III defines the concept of interpretable logic units and proposes a technical framework based on the priorities of logic unit test cases. We then describe the details of the proposed DeepLogic method in Section IV. Section V discusses the experimental verification of the validity of our previous theory, focusing on evaluating the effect of the proposed technique compared to other baseline methods. Finally, we summarize the study and introduce further research in Section VI.

## II. Related Work

### 1. Deep learning testing

Recently, several testing criteria have been proposed to improve the effectiveness of DNN model testing. Pei *et al.* [6] proposed, for the first time, the use of NC as a metric index, namely DeepXplore, to jointly optimize a solution to the white box differential testing problem. This method uses test inputs that enable a group of DNN models with the same function to produce differential behaviors and achieve high NC. Ma *et al.* [7] further extended the concept of NC by introducing three new neuron-level coverage standards and two hierarchical coverage standards, namely DeepGauge, which dynamically sets different neuron activation thresholds with multi-granularity to better reflect the differences between natural and adversarial samples. Sun *et al.* [8], referring to modified condition/decision coverage (MC/DC) metrics in the traditional software testing field, proposed four indicators, that is, conformity-sign coverage, distance-sign coverage, sign-value coverage, and distance-value coverage to measure the difference in neuron activation values between adjacent layers and implemented them in the DeepCover tool. Wang *et al.* [11], referring to path-oriented testing methods in traditional software engineering, proposed a set of path-driven test metrics called DeepPath, using a single neuron in the model as a node and the neuron connections between different layers as a path, and proposed three path coverage metrics. To ensure more accurate measurements of DNN robustness, Weng *et al.* [12] proposed a DNN robustness index namely Clever, based on the Lipschitz continuous extremum theory. Katz *et al.* [13] proposed the concept of adversarial robustness, that is, the ability of a model to correctly classify attack samples generated through small perturbations. Gehr *et al.* [14] introduced $AI^2$, which utilizes abstract interpretation theory to test deep learning models. The authors defined an abstract transformer to analyze the behavior of the implicit layer and employed the zonotope abstraction domain to identify potential adversarial inputs. Additionally, they established a benchmark for measuring model robustness by outputting the abstraction domain range.

Despite these achievements, some results [10], [15], [16] show that there is no positive correlation between the existing test criteria and the robustness of the DNN model. Moreover, existing test standards rarely consider test problems from a high-level semantics perspective of deep learning tasks and fail to identify the importance of different neurons in deep learning testing.

More recently, Wang *et al.* [17] presented RobOT, a

reinforcement learning-based testing technique that generates challenging and diverse test cases, providing an efficient and accurate method for robust testing of deep learning systems. Hu *et al.* [18] proposed a data distribution-aware test selection framework based on data importance and test case coverage. This framework adaptively selects test cases with high coverage to improve model robustness and stability. However, these studies focus on testing the adversarial robustness after model retraining, which is significantly different from our task of test case prioritization.

## 2. Interpretability of DNNs

The visualization method based on channel activation mapping can generate specific class-related activation maps for DNN decision results, making it a widely used DNN interpretation method. Simonyan *et al.* [19] introduced two visualization techniques that use gradient information to compute classification scores. The first technique reconstructs the maximized class score image from ConvNet and captures features such as edges and stripes. The second one calculates the contribution of each pixel point to the output result, generating a saliency map with the same dimensionality as the input image. These techniques successfully establish the connection between gradient-based convolutional networks and deconvolutional neural networks. After that, the channel activation mapping (CAM) method was proposed by Zhou *et al.* [20], i.e., CAM by inserting a global average pooling (GAP) layer into a DNN to form a DNN network with a full convolution structure and visualizing the connection weight of the SoftMax layer as the weighted weight of the top channel feature map. Selvaraju *et al.* [21] improved the CAM method using gradient weighting and proposed Grad-CAM with a wider range of DNNs.

More recently, Bau *et al.* [22] proposed a general framework called Network Dissection, where the basic unit of calculation was the corresponding channel of the convolutional filter (by comparing the activities of each channel with a series of human-explainable pattern-matching tasks, such as the detection of object classes). The semantic units were then given conceptual labels, including objects, components, scenes, textures, materials, and colors. Bai *et al.* [23] introduced channel activation into a robustness test and proposed a channel-wise activation suppressing (CAS) strategy. The basic idea was to screen feature channels with important contributions to the classification results, suppressing feature channels with low correlation. Inspired by the existing studies [24] on the interpretability of DNNs, we proposed the introduction of the "logic" displayed by high-level channels into DNN priority testing, making the testing process more effective and interpretable.

## 3. Test-case prioritization

The main research objectives of test-case prioritization technology can be summarized as follows: analyzing and determining positive contributions to the test, calculating the priority of test cases, and achieving an effective method of test-case ranking. Recent studies have proposed the use of test prioritization techniques to find defects as early as possible to evaluate DNN models.

Rothermel *et al.* [25] formalized the test-case prioritization problem as follows: Given the test-case set $T$, the full preordering set of $T$ is PT and the sorting objective function is $f$. The domain of $f$ is PT and the range is real. The purpose of test-case prioritization is to determine $T^{'} \in \mathrm{PT}$ to make: $\forall T''(T'' \in \mathrm{PT})(T'' \neq T')[f(T') \geq f(T'')]$. As is evident from the above description, PT contains all possible test-case ordering in $T$; function $f$ can be used to quantitatively describe the effect of evaluation ranking. The larger the $f$ value, the better the test-case ranking effect. Kim *et al.* [26] proposed a test-case prioritization method based on the test history. Li *et al.* [27] proposed several search algorithms to prioritize regression test cases. They focused on test-case prioritization techniques for code coverage, including block coverage, decision (branching) coverage, and statement coverage, which have all been extensively studied in previous work. Leon *et al.* [28] experimentally compared four existing methods of screening large test suites, namely, fault-tracking sampling, test suite minimization, cluster filtering using single-cluster sampling, and prioritization via additional coverage. By introducing logic unit correlation constraints, they evaluated the importance of each test case in the test-case set before selecting and executing the test cases in order of importance (from high to low). Consequently, test cases of high importance could be executed as soon as possible, improving test efficiency under limited test resource conditions.

Feng *et al.* [9] proposed DeepGini and designed a test-case preference-selection technology based on the probabilistic statistical perspective of the statistical DNN model classification decisions. DeepGini proved to be more effective and more efficient than existing overcover-based technologies, helping improve DNN robustness. However, it only used the output probability as the measurement index, and when strong adversarial attacks were implemented, its effectiveness was greatly reduced. Entropy [29], also known as Shannon entropy, is a widely used information-theoretic metric that measures the average level of information required to obtain a possible prediction. Similar to DeepGini, the ranking criterion of entropy is the classification decision of statistical probability distribution based on the DNN model.

More recently, Shen *et al.* [30] proposed a multi-boundary clustering algorithm (multiple-boundary clustering and prioritization, MCP), capable of dividing the training data into multiple clusters, each with different boundaries and importance. The authors also proposed a clustering cluster-based prioritization method to prioritize each cluster in retraining based on its boundaries and importance. Experimental results indicated that the method can improve the efficiency and accuracy of retraining, while reducing training time and resource con-

sumption. Kim *et al.* [31] proposed a test case selection method based on "surprise adequacy", where distance-based surprise coverage (DSC) computed the surprise adequacy using the Euclidean distance between the model's behaviors represented by the activation traces of the test sample and the training set. DSC improved test coverage and accuracy by prioritizing data points that cannot be correctly predicted by the model.

After that, Sharif *et al.* [32] introduced DeepOrder, a test case prioritization technique for continuous integration testing. DeepOrder utilizes supervised learning and incorporates the test history from the last four cycles of continuous integration testing to enhance the fault detection efficiency of prioritized test suites. Li *et al.* [33] also proposed a testing technique that combines intrinsic and contextual features of untagged test cases for prioritization. This technique constructs a similarity graph on test instances and training samples, followed by semi-supervised learning based on the graph to extract contextual features. The test cases are prioritized in descending order based on their probability values. Unfortunately, the focus of these studies differs substantially from ours and they cannot be employed to effectively test the robustness of DNN models.

## III. Interpretable Logic Units

### 1. Motivation

The explanatory theory of deep learning models [21], [34] indicates that different channels in the convolution layer have different modes and focus on learning different image features, which implies different logical semantics. In this study, the average activation value of the neurons in these different channels is defined as a neural unit. Using the convolutional neural network of visual recognition tasks, such as the VGG-16 model, as an example, GAP [35] can be used to replace the fully connected layer of the model. The weighted sum of the channel weight and the corresponding feature map are determined before being superimposed onto the pixel area of the original sample after up-sampling. The 512 channels of the last convolution layer, that is L40, are visualized, as shown in Figure 1, several feature maps showing the "turtle shell" nerve unit, "feather" nerve unit, and "butterfly wing" nerve unit to be "logical".

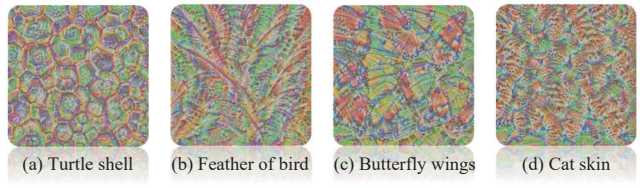



**Figure 1** The "logic" exhibited by several VGG-16 neural units.

To further explore the correlation between the average activation value of the neural units and the model decision, we selected several samples and plotted the activation curve of each neural unit. Figure 2 shows the results of the activated nerve units No. 281 and No. 125, corresponding to two different samples of persian_cat,

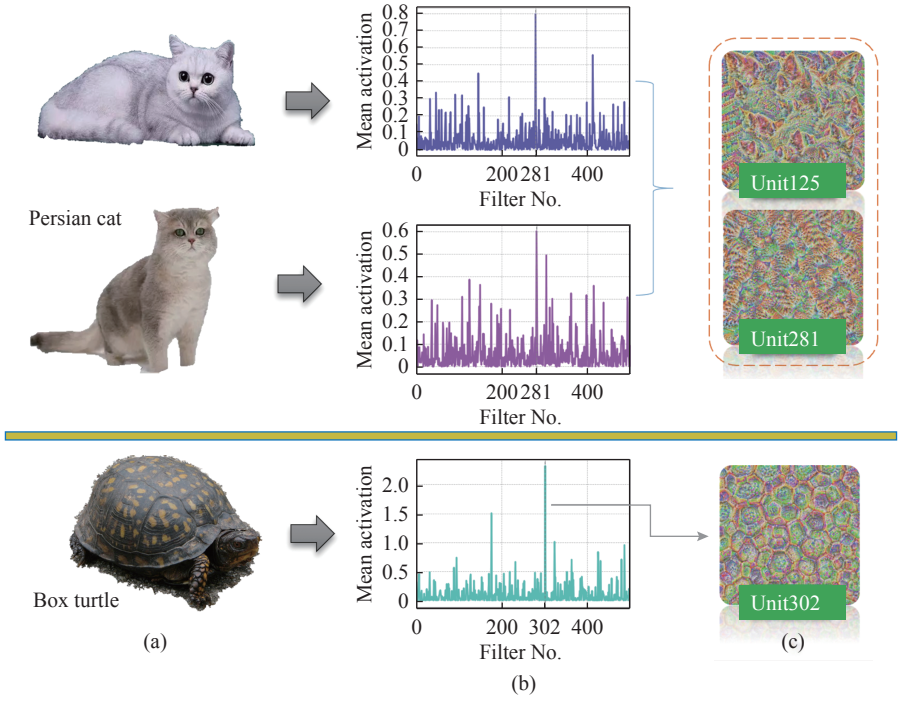



**Figure 2** The considered judgment of the VGG-16 model is related to the "logic" of the neural units, the performance of different types of samples differing. Column (a) indicates the selected sample images of the two distinct classes Persian_cat and Box_turtle. Column (b) indicates the average activation value of 512 channels of the convolution layer L40 of the sample images. Column (c) indicates the visualized image corresponding to the neural unit with the highest activation value.

the visualized neural units showing the logical characteristics of the cat skin and ears. Additionally, the neural unit (No. 302) with the maximum activation value of a box_turtle sample shows the logical characteristics of the turtle shell.

Based on the explanatory theory and visualized analyses, we proposed the following hypothesis:

**Hypothesis** The internal neural units in DNN models are similar to human neurons with "logic", the activated "logical" neural units helping the model make correct predictions. Additionally, we found that only parts of the logical neural units were interpretable and had an effect on the model decision. For example, a sample was identified as a cat because the "ear" nerve unit, "paw" nerve unit, "cat skin" nerve unit, and other logic units in the model were activated.

Based on the statistics of neuron unit activation probability, we found that for a given DNN model, the activation probability of neurons corresponding to different types of samples varied considerably. This phenomenon also appeared when the input samples were adversarial examples. Based on the above analyses, we argue that the same neuron could have completely different roles in different prediction tasks and that the logic units could affect the robustness of the model. Additionally, a correct prediction depends, to a large extent, on the combined action of multiple neuron units.

Thus, we define these neuron units to be interpretable logic units. We then rank them by importance, classify them, and track their activation in the DNN test task as the basis for the testing effect to determine the logic unit test criteria.

## 2. Two types of logic units

In [22] and [23], the authors demonstrated that multiple neural units (channels) could be regarded as a causal structure of deep network behavior to detect objects, parts, textures, tense, context, and so forth. The prediction results of a DNN rely on small-scale important units (20 most important units) rather than large-scale least-important units (492). Additionally, they discovered that certain important channels are highly correlated to the prediction results when the inputs are adversarial examples. This property of units in DNNs inspires us to design two types of logic, i.e., sample logic units (SLUs) and class logic units (CLUs), which are calculated to select the units (channels) that affect the model decision, helping the model identify as many abnormal samples as possible in the adversarial attack test environment. SLU illustrates the prediction mechanism of deep learning models, i.e., the decision is based on several important logic units (channels). Conversely, CLU reflects that the various classes exhibit statistically significant discrepancies in accuracy due to the cumulative differences of a large number of individual samples.

Figure 3 shows the logic unit calculation process. The training samples are processed using various convolution kernels for each convolution layer. The output feature map shows the different activation conditions for the captured information. With the training data input into the model, the parameters of the convolution kernel are gradually trained and optimized. When the training is completed, the DNN model makes predictions by aggregating information from different channels/feature graphs.

Given dataset $D$, data sample $x \in \boldsymbol{X}$, and label $y \in \boldsymbol{Y}$, the deep learning model is a learning mapping or classification function $\boldsymbol{M}: \boldsymbol{X} \to \boldsymbol{Y}$, model $\boldsymbol{M}$ consisting of $L$ hidden layers. $\boldsymbol{M}_l$ denotes the $l$th hidden layer of model $\boldsymbol{M}$, where $l = 1, 2, \ldots, L$. It contains several channels corresponding to a group of neurons, which are defined here as neuron units, the output being a feature map. The superscript $k$ denotes the $k$th channel, and the activation of all neurons satisfies $\boldsymbol{M}_l^k \in \boldsymbol{M}_l$. We apply the Frobenius norm to the original activated matrix $\boldsymbol{M}_l^k$, and the average channel from the activation values $\overline{\boldsymbol{M}_l^k} \in \mathbf{R}$ can be calculated. $H$ and $W$ denote the height and width of the channel, respectively.

$$\overline{\boldsymbol{M}_l^k} = \sqrt[2]{\sum_{i=1}^{H}\sum_{j=1}^{W}(M_{l,i,j}^2)^2} \tag{1}$$

Given the hyperparameter threshold $\sigma$, when $\overline{\boldsymbol{M}_l^k} > \sigma$, it indicates that the nerve unit is activated; otherwise, the nerve unit is not activated. The activated neural units can be sorted based on their activation values, the top-$k$ units being selected as the SLUs of a single sample:

$$\boldsymbol{\Omega}_l = \mathrm{TOP}_k(\overline{\boldsymbol{M}_l^k}, \sigma) \tag{2}$$

**Definition 1** SLUs: a series of neural units that can cause the activation of a specific logical region. Specific logical region refers to the channel corresponding to each neural unit.

Owing to the poor stability of single-SLUs, it is necessary to further calculate the CLUs of each category. The idea is to design a fully connected network, with all the samples of the category as the input, the category one-hot vector as the output, and the network parameters obtained after training as the channel weight matrix.

Assuming that all samples $x_i$ $(i = 1, 2, \ldots, n)$ of category $y_i$, the SLUs are denoted as $\boldsymbol{\Omega}_l = \Omega_1, \Omega_2, \ldots, \Omega_l$, by mapping:

$$\boldsymbol{W} * \boldsymbol{\Omega}_l \to y \tag{3}$$

the channel weight matrix can be obtained and split according to the row vector, where the asterisk $*$ means the matrix multiplication. After sorting, the sequence number of the top-$k$ corresponding neural units can be selected to obtain the corresponding CLUs.

$$\boldsymbol{\Omega}_i = \mathrm{TOP}_k(\mathrm{argsort}(\overline{\boldsymbol{W}_i})) \tag{4}$$

**Definition 2** CLUs: the probability statistics of all sample sets of logic units to which the class belongs. It should be noted that CLUs are divided for a certain cate-

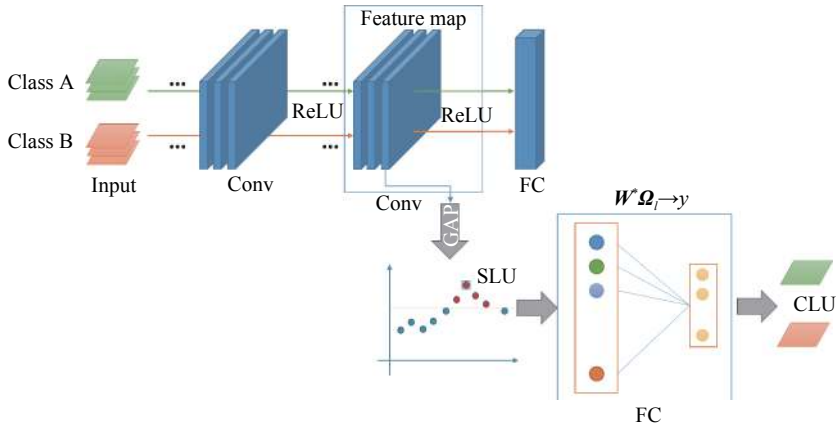



**Figure 3** Schematic of the logic units calculation process. First, the samples from different classes (e.g., Class A and Class B) are input into the neural network. The output feature maps of the convolutional layers are calculated according to equation (1) for their average activation values. These values are then sorted. The red dots indicate the important neural units, i.e., SLUs, whereas the blue dots indicate the neural units that can be ignored. The input of FC is the SLUs of a certain class, and the weight matrix $\boldsymbol{W}$ is calculated according to equation (3) and split according to the row vectors. Finally, the top-$k$ is selected as the output CLUs in order. The different classes correspond to different CLUs.

gory rather than the whole dataset, and CLUs of different categories may intersect.

The CLU can be calculated using Algorithm 1.

**Algorithm 1** Algorithm for CLU calculation

**Input:** data sample $x \in \boldsymbol{X}$, labeled $y \in \boldsymbol{Y}$, convolution layer number $l$, and activation threshold $\sigma$.

**Output:** logic unit vector $\boldsymbol{\Omega}_l$.

Load DNN model and initialize parameters;
Obtain all samples $x_1, x_2, \ldots, x_n$ of class $c$;
**for** $i = 1 : n$ **do**
    Sample $x_i$ input model;
    Calculate $\boldsymbol{M}_l^1, \boldsymbol{M}_l^2, \ldots, \boldsymbol{M}_l^k$ according to (1);
    Calculate $\overline{\boldsymbol{M}_l^k}$;
**end for**
Randomly initialize the weight matrix $\boldsymbol{W} = \text{random}(K, C)$. Here $K$ denotes the total number of channels and $C$ denotes the total number of categories;
**for** $i = 1 : n$ **do**
    Calculate neural unit activation value $m_i$ and label $y_i$ of sample $x_i$;
    Calculate $\overline{y_l^k} = \overline{\boldsymbol{M}_l^k} * \boldsymbol{W}$;
    Calculate $\theta = \frac{1}{2N} \sum_{i=1}^{N} (\overline{y_l^k} - y_i)^2$;
    Update $\boldsymbol{W}' = \boldsymbol{W} - \delta \bigtriangledown_l$ and $\boldsymbol{W} = \boldsymbol{W}'$. $\bigtriangledown$ denotes the weight update using gradient descent, and $\delta$ denotes the step size;
**end for**
Calculate $\overline{\boldsymbol{W}} = \frac{\boldsymbol{W}_{i,1}}{\max(\boldsymbol{W}_{0,1}, \boldsymbol{W}_{1,1}, \ldots, \boldsymbol{W}_{i,1})}$;
Select top-$k$ $\overline{\boldsymbol{W}} > \sigma$ as a logical sequence of neurons $\boldsymbol{\Omega} = \text{TOP}_k(\overline{\boldsymbol{W}})$;
Output sequence $\boldsymbol{\Omega}_l$.

Note that as described in prior studies [20], [22], [23], not all important units with high activation values are human-interpretable. Nevertheless, these units are predominantly positively correlated with their associated classes and assist the network in making the correct decisions. Similarly, not all units of logic units in this work are human-interpretable yet they can contribute to executing effective priority testing.

## 3. Measurement of logic units

We measure the similarity and distance between the SLU and CLU from two perspectives. The Jaccard similarity coefficient is introduced, the operation $\cup$ denotes the union of sets, the operation $\cap$ denotes the intersection of sets, and the operation $\|\cdot\|$ denotes the number of set elements.

1) Single-sample logic degree

Given sample SLU and CLU of a certain class, the similarity between these two units can be calculated as

$$\text{SJ} = \frac{\|\text{SLU} \cap \text{CLU}\|}{\|\text{SLU}\| + \|\text{CLU}\| - \|\text{SLU} \cap \text{CLU}\|} \tag{5}$$

The larger the SJ value, the more similar the two units, that is, the more overlapping the set elements. The smaller the SJ value, the less overlapping the set elements.

Given sample SLU and CLU of a certain class, the distance between these two units can be calculated as

$$\text{DJ} = \frac{\|\text{SLU} \cup \text{CLU}\| - \|\text{SLU} \cap \text{CLU}\|}{\|\text{SLU} \cup \text{CLU}\|} \tag{6}$$

The larger the DJ value, the lower the number of overlapping set elements between the two units, and the smaller the DJ value, the higher the number of overlapping set elements between the two units.

Given a clean sample or an adversarial sample $x$, the SLU can be obtained using Algorithm 1, i.e., $\boldsymbol{\Omega}_x = \Omega_1, \Omega_2, \ldots, \Omega_k$. Let the model judgment classification be $y_c$, and the sample category label set be $\boldsymbol{Y} = y_1, \ldots, y_c, \ldots, y_k$, $y_c \in \boldsymbol{Y}$. The subscript $k$ denotes the top-$k$ hyperparameter selected by the logic unit. We can measure the

deviation between the sample and the model to determine the classification $y_c$ by calculating the sample logic degree:

$$L = \frac{\mathrm{DJ}(\mathrm{SLU}_x, \mathrm{CLU}_{y_c})}{\sum_l^k \mathrm{DJ}(\mathrm{SLU}_x, \mathrm{CLU}_{y_i})},\ y_i \cap y_c = \varnothing \tag{7}$$

**Definition 3** Logical degree: a measure of whether the DNN model correctly classifies a given sample.

If the logic units of a single sample deviate from their CLUs and the $L$ value is too large, this indicates that the sample has been attacked and the model prediction result $y_c$ is probably incorrect. We can then place this sample in the priority test queue. Otherwise, if the logic units of a single sample deviate from their CLUs and the $L$ value is small, it indicates that the sample is probably clean. The $y_c$ predicted by the model is correct, and this sample is not placed in the priority test queue.

2) Class logic unit correlation

Given two different classes of logic units, i.e., CLU1 and CLU2, the similarity or distance between them can be expressed as follows:

$$\mathrm{CJ} = \frac{\|\mathrm{CLU1} \cap \mathrm{CLU2}\|}{\|\mathrm{CLU1} \cup \mathrm{CLU2}\|} \tag{8}$$

The smaller the value of CJ, the lower the number of overlapping set elements between the two logic units from different classes. The larger the value of CJ, the higher the number of overlapping set elements between the two units. It is not difficult to conclude that the logic units of different classes should be independent of each other and that their distance should be large. Consequently, logic units can be more effective during the testing process. The experimental results validate this argument.

## IV. DNN Priority Testing via Interpretable Logic Units

### 1. Overview

This paper proposes a deep logic framework based on the priority of logic units. It integrates the weights of internal attributes and context attributes in the test process and prioritizes a large number of test cases, helping to detect more model defects during the early stages. The overall structure of the deep logic test framework is shown in Figure 4. The main body of the framework comprises four parts, that is, preprocessing, internal attribute extraction, context attribute extraction, and priority sorting.

The preprocessing module performs two main functions. The first is to reclassify the samples based on the decision classification. Using the CIFAR-10 dataset as an example, there are 10 different categories, the samples being divided based on the above 10 categories instead of being mixed together. The second function is to produce adversarial examples and incorporate them into the test-case set for data augmentation. Adversarial training is applied to generate defense models and improve the generalization ability of the test-case priority.

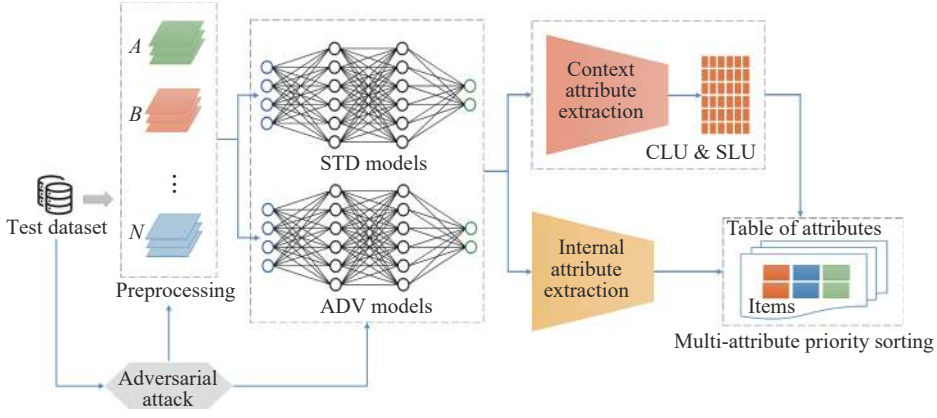



**Figure 4** Overview of DNN priority testing via logic units. The test cases go through two paths (contextual attribute extraction and internal attribute extraction) after pre-processing. Both standard (STD) trained models and adversarial (ADV) trained models are used for testing. The attribute table records the serial number, logic degree, defect severity, etc., of the test cases.

The context attribute extraction module introduces the test criteria of interpretable logic units and extracts the sequence of CLUs by traversing all samples of each class. It compares the deviations between test-case logic units and their CLUs.

Inspired by DeepGini, the internal attribute extraction module calculates the probability distribution of the SoftMax output vector after the current test case is input into the DNN model. The fusion comparator accepts the context and internal attribute extraction modules as inputs and calculates their weights. Subsequently, the output is used as the test-case priority evaluation criterion of the priority sorting module.

### 2. Context attributes extraction

According to the algorithm flow of logic units (see Figure 3) introduced in Section III.2, the average channel activation values of the divided multiple training-set subsets based on the classification categories are calcu-

lated, after which the channel weight matrix can be obtained accordingly. The top-$k$ channel is selected as the CLU channel. Adversarial examples can then be generated to expand the test-case set and feed it into the DNN models. The SLU calculates the logic degree according to the method presented in Section III.3 for each test case. The results are filled in the multiple attribute table entries and then prioritized based on the sorting rules. Priority is defined as the logical degree of each test case. The smaller the logical value, the higher the priority. Finally, we place it in the header of the sorted test set until the test set is empty and the sorting ends.

If the logic units of a single sample deviate from its CLUs, it indicates that the sample has been attacked, and the model prediction is probably incorrect. The sample is placed in the priority test queue. Otherwise, it indicates that the sample has not been attacked by the adversary and the prediction result is correct. Thus, it is not placed in the priority test queue.

## 3. Internal attribute extraction

Reference [29] demonstrated that the attributes of a single test case were also important factors to be considered in the prioritization process. For a given sample $x \in \boldsymbol{X}$ and DNN model $\boldsymbol{M} : \boldsymbol{X} \to \boldsymbol{Y}$, the output of the last layer of the model, for example, the SoftMax layer, is the probability confidence $p_1, p_2, \ldots, p_K$. DeepGini uses the probability confidence mean square as a test-case prioritization criterion, i.e., $\varepsilon = 1 - \sum_{i=1}^{N} (p_i(x))^2$. Because information entropy, also known as Shannon entropy, can be used to measure the uncertainty of DNN model output predictions, its introduction can more accurately represent the characteristics of test cases. Thus, inspired by DeepGini, we propose the use of information entropy, selecting the sample data with the largest uncertainty to be the internal attribute, which can be formally expressed as

$$\tau(t) = -\sum_{i=1}^{N} (p_i(x))^2 \log p_i(x) \tag{9}$$

According to the Lagrange multiplier,

$$L(p_{t,i}, \lambda) = \tau(t) + \lambda \times \sum_{i=1}^{N} p_i(x) \tag{10}$$

where $p_{t,i}$ represents the probability that the test $t$ belongs to the class $i$, we calculate the partial differentiation of $p_{t,1}, p_{t,2}, ..., p_{t,n}$ and let

$$\begin{aligned} \frac{\partial L}{\partial p_{t,N}} &= -2p_{t,N} \times \log p_{t,N} - p_{t,N} + \lambda \\ &= \lambda - p_{t,N}(2\log p_{t,N} + 1) = 0 \end{aligned} \tag{11}$$

If any two of the above equations (such as equations $i$ and $j$) are calculated, we can obtain

$$p_{t,i}(2\log p_{t,i} + 1) = p_{t,j}(2\log p_{t,j} + 1) \Rightarrow p_{t,i} = p_{t,j} \tag{12}$$

Therefore, if and only if $p_{t,1} = p_{t,2} = \cdots = p_{t,N} = 1/N$, $\tau(t)$ has the highest value. The probability confidence $p_1, p_2, \ldots, p_K$ for the classification task of the DNN model corresponding to the category $C_1 \to C_K$ cannot be distinguished, leading to an incorrect model prediction. Based on the above discussion, $\tau(t_1) < \tau(t_2)$ indicates that $t_2$ is more likely to be misclassified. Therefore, $\tau$ is used as one of the prioritization metric attributes for $N$ test cases in dataset $D$.

## 4. Multi-attribute priority sorting

After extracting context attributes and intrinsic attributes, this module stores several key attribute values to form a multi-attribute table corresponding to each test case and uses these attribute fields to analyze and judge whether the model makes the correct prediction. We can compare the degree of logical deviation of the context attributes for each test case. If it exceeds a given threshold, an incorrect model prediction can be confirmed. Otherwise, it is difficult to verify the accuracy of the model prediction. The internal attributes of the test case can then be compared to determine whether the model is classified incorrectly. The higher the deviation degree and the larger the internal attributes, the smaller the number of test cases in the sorted test queue. During the test process, the sample number (from small to large) is input into the model to be tested for the prediction of results. The error detection efficacy can be measured using the relationship between the number of test cases and the number of errors that can be detected in those test cases.

## 5. Defect detection capability assessment

To measure defect detection capability, we adopted two existing evaluation metrics and proposed a new metric as well.

1) Average percentage of fault detection

The average percentage of fault detection (APFD) index proposed by Mor [36] is introduced to evaluate whether using logic as guidance to select test cases can improve the error detection ability. This metric can measure it using the relationship between the number of test cases and the number of errors that can be detected in those test cases. Moreover, the APFD can measure the performance of various test-case prioritization techniques using the same criteria.

Suppose there is a set of test cases $T$, including a total of $n$ test cases, among which the wrong test-case set is $F$, which is composed of all errors. Assume that $F = \{f_1, f_2, \ldots, f_m\}$, i.e., there is a total of $m$ errors in $F$. $T'$ denotes the $T$ test-case set based on the coverage of test cases after picking the order set. If we use $tf \in T'$ to represent the first error $f$ test case detected in $T'$, then APFD values of $T'$ can be calculated as follows:

$$T_{\text{APFD}} = \left(1 - \frac{tf_1 + tf_2 + \cdots + tf_m}{n \times m} + \frac{1}{2n}\right) \times 100\% \quad (13)$$

The range of the APFD is $[0, 1]$, and the larger the value of the APFD, the faster the error detection speed of the test sort and the better the sorting efficacy of the test set.

2) Risk detection ability

Reference [29] proposed a testing metric called RDA in the traditional software testing field. In this study, we adopt this metric and extend it into the deep learning testing field. First, we define the quantitative representation of the RDA. Based on the impact of defects on model classification, the defect severity can be divided into several groups, that is, no defects, general defects, major defects, and serious defects. To measure the severity of defects, quantitative values of the severity of various defects can be defined, that is, 0, 5, 10, and 100, respectively. If the classification is correct, the maximum classification probability is equal to the correct label; then RDA = 0 represents no defects. If the classification is incorrect and the second largest classification probability is equal to the correct label, then RDA =5 represents general defects. If the classification is incorrect and the third-largest classification probability is equal to the correct label, then RDA =10 represents major defects. If the classification is incorrect and the top-3 classification probability has no correct label, then RDA =100 represents serious defects. Thus, the RDA can be computed as

$$T_{\text{RDA}} = \frac{\sum_i^N \text{RDA} * F}{n} \quad (14)$$

The RDA range is $[0, +\infty]$. In (14), the smaller the RDA value, the earlier serious defects can be detected, and the better the sorting effect of the test set. It is important to note that the RDA values need to be calculated during the test-case sequencing process and also recorded in the table of attributes, after which the defect severity detection capability assessment is performed during the evaluation.

## V. Experimental Results

In this study, the experimental environment included an Ubuntu 20.04 LTS platform, running an NVIDIA TITAN XP GPU, and an Intel Core i7-8700K CPU @ 4.30 GHz × 8. We used the PyTorch 1.7.0 deep learning framework to implement our method.

To verify the rationality and validity of our deep logic testing based on interpretable logic units, we conducted experiments on several datasets and models to seek answers to the following research questions:

**RQ1** Feasibility: In the decision-making process of the model, is the correct prediction related to the activation modes of the different channels? (Some channels are frequently activated, while others are less likely to be activated)

**RQ2** Effectiveness: Can DeepLogic distinguish between clean samples and adversarial samples?

**RQ3** Availability: Does DeepLogic have a higher error detection rate in test-case prioritization than existing methods?

### 1. DNN models and datasets

In our experiments, the widely used VGG-16, ResNet-18, and ResNet-34 models were adopted as DNN models to be tested. A convolutional neural network, SmallCNN, was designed and trained for universality validation. The popular CIFAR-10 and FashionMNIST datasets were adopted. The CIFAR-10 dataset contains 60000 32 × 32 color images in 10 categories, including 50000 images in the training set and 10000 images in the verification set. And the FashionMNIST dataset contains 70000 28 × 28 black and white images in 10 categories, including 60000 images in the training set and 10000 images in the verification set. The DNN models were trained separately on the original sets, and the accuracy rate was greater than 85%. Table 1 summarizes the details of the experimental setting.

**Table 1** DNN models and datasets in our experiments

| Model | Parameters | Dataset | Class | Size | Accuracy |
|---|---|---|---|---|---|
| VGG-16 | 138357544 | CIFAR-10 | 10 | 60K | 93.1% |
| ResNet34 | 63470656 | CIFAR-10 | 10 | 60K | 92.7% |
| ResNet18 | 33161024 | FashionMNIST | 10 | 70K | 93.4% |
| SmallCNN | 12582912 | FashionMNIST | 10 | 70K | 92.2% |

Note: K=$10^3$.

### 2. Adversarial sample generation and defense model

Several popular adversarial attack techniques, including the fast gradient sign method (FGSM) [37], projected gradient descent (PGD) [38], and Carlini and Wagner (C&W) attacks [39] have been used to generate adversarial samples. The implementation details of these methods are as follows. For FGSM, we set $\epsilon = 0.093$. For PGD, specifically including PGD20 and PGD100, we set $\epsilon = 0.1$, Inf normal form, step_size = 0.003, and iteration times being epoch = 20 and epoch = 100, respectively. For C&W, we set $\epsilon = 0.2$, confidence $k = 0$, L2 normal form, step_size = 0.01, and iteration time as

epoch = 10000.

First, we expanded the test-case set by incorporating adversarial samples into the set and comprehensively evaluated the robustness of the model. The final test-case set consisted of 7/10 clean samples, 1/10 FGSM attack samples, 1/10 PGD attack samples, and 1/10 C&W attack samples.

Conversely, adversarial training was carried out on the DNN models, and the tradeoff-inspired adversarial defense via surrogate loss minimization (TRADES) [40] method was adopted. Subsequent experiments verified the effectiveness of the DeepLogic method proposed in this paper for the standard model and the defense model. Table 2 summarizes the details.

**Table 2** Comparison of accuracy of the standard model and defense model

| Model | Clean | FGSM | PGD20 | PGD100 | C&W |
|---|---|---|---|---|---|
| VGG-16 | 0.931/0.822 | 0.122/0.597 | 0.028/0.498 | 0.025/0.483 | 0.000/0.472 |
| ResNet34 | 0.927/0.856 | 0.161/0.652 | 0.001/0.535 | 0.001/0.517 | 0.001/0.528 |
| ResNet18 | 0.934/0.911 | 0.421/0.878 | 0.001/0.863 | 0.001/0.855 | 0.005/0.863 |
| SmallCNN | 0.922/0.887 | 0.586/0.849 | 0.291/0.834 | 0.288/0.829 | 0.261/0.830 |

## 3. Channel activation mode (RQ1)

We validated the positive relationship between the correct prediction of the DNN models and the activation modes of different channels. Specifically, to explore the activation frequency of the different neural units using the samples in each category, the proportion of the activation times of each channel, i.e., the average activation value greater than the threshold, was calculated for each category. We then arranged them in descending order based on the channel activation proportion, as shown in Figure 5. The horizontal axis represents the channel number from the largest to the smallest activation proportion. Channel 0 represents the channel with the highest activation frequency under the current category, and channel 0 could have different channels under different categories. The vertical axis represents the proportion of activation times of the different channels, i.e., the frequency.

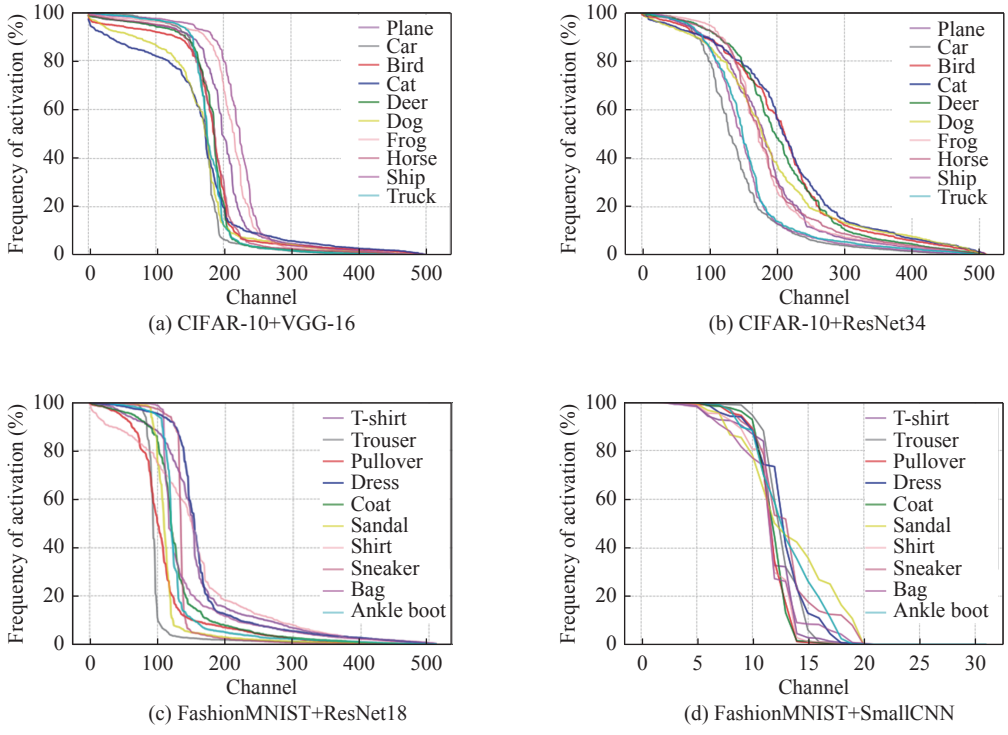



**Figure 5** Abrupt decrease of channel activation probability in different samples. (a)–(d) show the results of different models on different datasets.

Figure 5(a) shows the results of the VGG-16 models on the CIFAR-10 dataset. It is evident that for the samples in each category, the activation ratio shows a sharp decline when the channel is No. 200 and the decline ranges from 80%–100%. Figure 5(b) also shows similar results. For the combination of FashionMNIST and SmallCNN, as shown in Figure 5(c) and 5(d), although the decline in the channel activation ratio is not as obvi-

ous as that on CIFAR-10, a similar trend is also evident. Figures 6 and 7 illustrate the top-10 logic units and their activation probability for 10 different categories on the CIFAR-10 dataset of the VGG-16 and ResNet34 models, respectively. It is evident that the activation frequencies of the first 10 channels of the same category (e.g., plane category) are quite different, suggesting that the channel activation frequencies of the same dataset under different models are inconsistent and the other logic units also have their characteristics in different models.

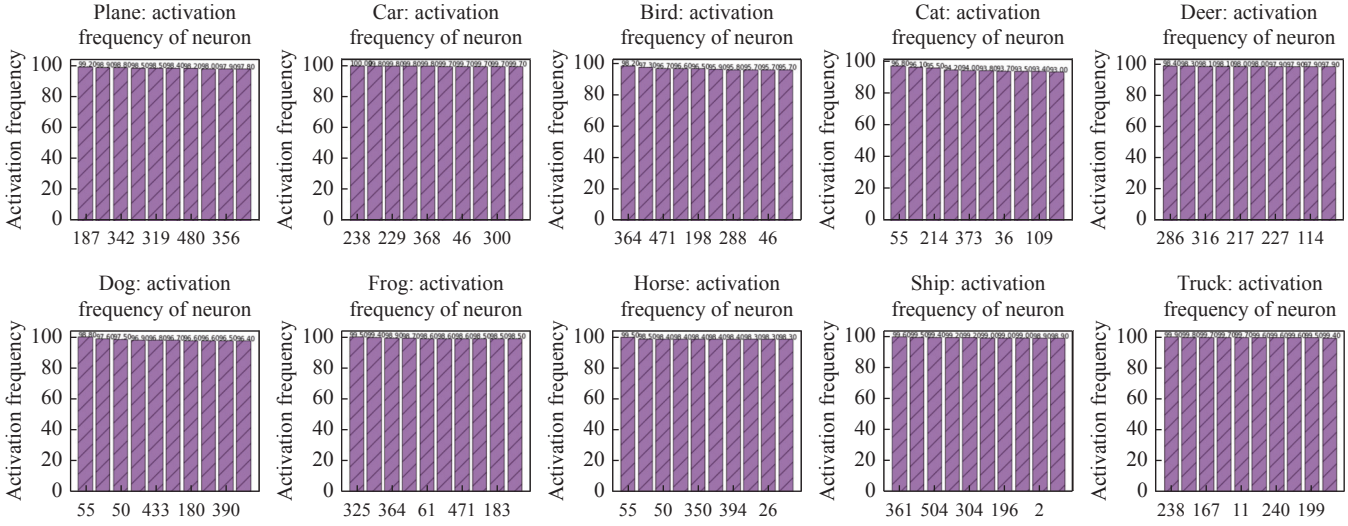



**Figure 6** Top-10 logic units with the highest average activation values for different categories on VGG-16 model.

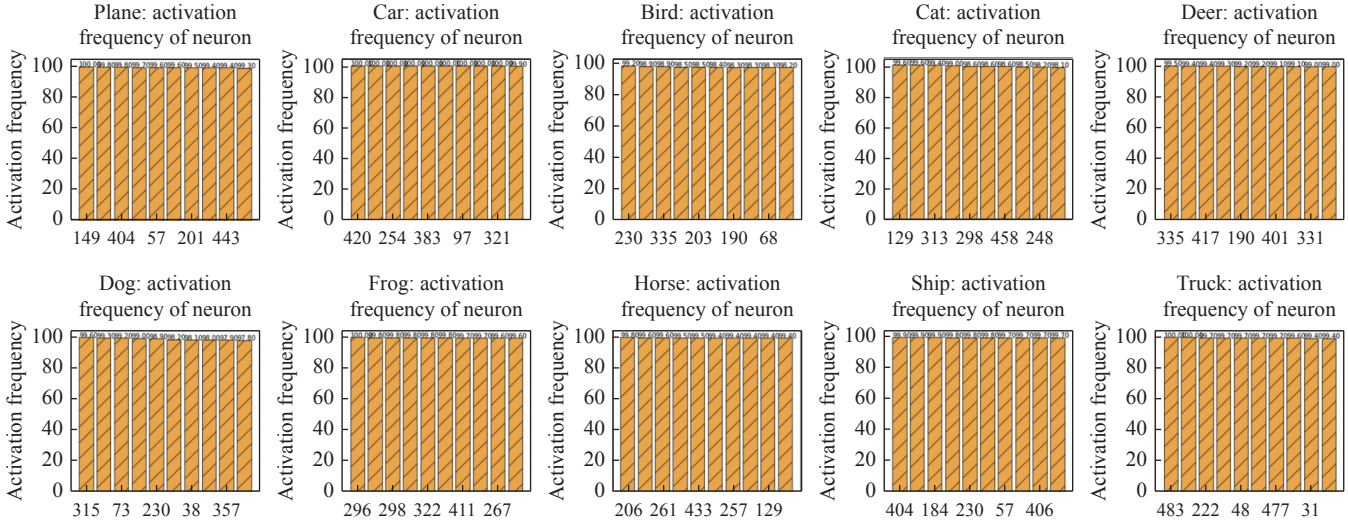



**Figure 7** Top-10 logic units with the highest average activation values for different categories on ResNet34 model.

Additionally, the activation probabilities of some neural units in different classes are shown in Table 3. The activation probability of neural unit No. 120 in "car" is as high as 99.5%, while that of the horse category is 0.6%. This means that 995 out of 1000 samples of the former were activated, and only six of the latter samples had an average activation value of unit No. 120 above the threshold. Based on the statistics of neuronal unit activation probability, we found that for a given DNN model, the activation probability of neurons corresponding to samples from different categories varied greatly. This phenomenon also appears in the activation probability of neurons when the inputs are clean and adversarial. Consequently, the results and analysis indicate that correct DNN predictions greatly depend on the combined action of multiple logic units.

## 4. Different logic unit distance (RQ1)

The logic units of each category represent the features extracted from that category by the DNN. The samples in each category have different characteristics, and the logic units of each category differ. To verify this, we calculated the Jaccard similarity coefficient between each class of logic units and other classes of logic units. We first converted two different CLUs into two sets of

**Table 3** The activation probabilities of some neural units in different classes differ (VGG-16+CIFAR-10)

| Class | Unit120 | Unit176 | Unit233 | Unit284 | Unit305 | Unit391 | Unit431 | Unit474 | Unit502 |
|---|---|---|---|---|---|---|---|---|---|
| Plane | 4.20% | 2.00% | 88.70% | 15.70% | 0.90% | 45.60% | 96.60% | 2.00% | 96.10% |
| Car | 99.50% | 0.10% | 0.40% | 1.10% | 4.80% | 5.40% | 96.50% | 49.90% | 0.30% |
| Bird | 89.60% | 93.30% | 93.50% | 95.20% | 2.70% | 4.10% | 92.80% | 90.60% | 95.40% |
| Cat | 6.90% | 79.70% | 3.30% | 90.10% | 74.00% | 5.20% | 3.30% | 10.00% | 3.50% |
| Deer | 0.70% | 88.50% | 1.40% | 2.00% | 1.10% | 97.00% | 2.20% | 2.20% | 95.50% |
| Dog | 65.30% | 5.10% | 1.90% | 95.90% | 2.80% | 8.80% | 2.80% | 89.80% | 2.80% |
| Frog | 0.90% | 2.10% | 96.10% | 8.60% | 96.30% | 2.90% | 4.60% | 1.70% | 2.20% |
| Horse | 0.60% | 3.10% | 0.80% | 1.70% | 78.50% | 3.60% | 96.00% | 94.80% | 2.90% |
| Ship | 97.40% | 0.60% | 1.20% | 97.00% | 1.50% | 98.70% | 3.80% | 1.80% | 97.90% |
| Truck | 96.70% | 0.20% | 0.60% | 0.60% | 96.90% | 98.60% | 36.60% | 98.30% | 2.00% |

integers and then used Jaccard similarity coefficients to measure the similarity between them. The larger the Jaccard similarity coefficient, the more similar the two different classes of logic units. And the results are shown in Figure 8. Using the CIFAR-10 dataset as an example, the similarity of the logic units of the samples from different categories is less than 0.25, that is, the logic units are very different. Among them, the Jaccard similarity between the "car" and "truck" categories is the highest, reaching 0.32, which may be because several logic units of these two categories overlap.

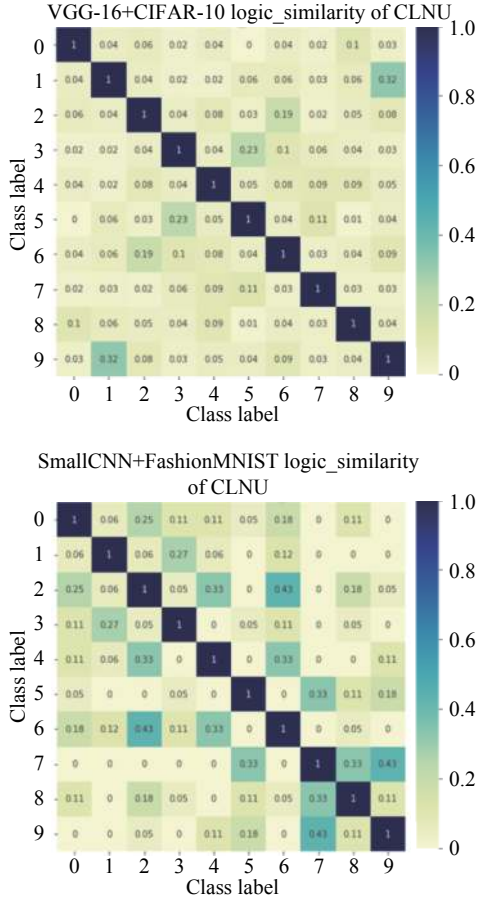



**Figure 8** Different logic unit distances in different experimental settings.

## 5. Validity verification of logic units (RQ2)

To answer RQ2, for a given test sample set, we counted the number of successful attacks of the adversarial samples and the number of changes in the logic units before and after the attacks on the standard and defense models, respectively. We then calculated the ratio of the two to be the effectiveness index. To determine whether the logic units changed before and after the attack, we calculated the SLU of the sample before the attack and the CLU of the class to which the sample belonged after the attack. The distance between them was then calculated based on (6) (Section III.3). If the result exceeded the hyperparameter threshold, the logic units were considered to have changed; otherwise, there was no change. Both standard and defense models were adopted in this experiment. The FGSM, PGD, and C&W adversarial samples were used for adversarial training. The logic units of the clean and adversarial samples were calculated for the two DNN models for comparison.

The results are presented in Table 4. We analyze the experimental data of the standard model testing. It can be seen that before and after the FGSM attack, the change rates of the logic units in the samples of the VGG16, Resnet34, and Resnet18 models are 90.9%, 83.4%, and 80.5%, respectively. The change rates of the SLUs in the VGG-16, ResNet34, and ResNet18 models before and after the PGD20 attack are 89.4%, 94.6%, and 89.1%, respectively. The change rates of the SLU before and after the C&W attack are over 88% except for the ResNet18 model (64.7%). These results demonstrate that a considerable number of adversarial samples can achieve successful attacks owing to changes in the interpretable log-

**Table 4** Average change rate of logic units before and after the standard model and defense model against the sample attack

| Model | FGSM | PGD20 | C&W |
|---|---|---|---|
| VGG-16 | 0.909/0.910 | 0.894/0.898 | 0.913/0.918 |
| ResNet34 | 0.834/0.841 | 0.946/0.934 | 0.882/0.900 |
| ResNet18 | 0.805/0.877 | 0.891/0.909 | 0.647/0.734 |
| SmallCNN | 0.972/0.987 | 0.954/0.981 | 0.957/0.994 |

ic units. These results also illustrate that the SLUs change rate of the standard model is lower than that of the defense model under the FGSM/PGD20/C&W attack. The reason for this phenomenon could be that the attack success rate decreases under the defense model. The total number of sample changes decreases, and the number of logic units changed does not differ much, which eventually leads to an increase in the proportion of logic units. The cumulative relationship between the rate of change of the SLUs and the number of samples was further analyzed, as shown in Figure 9. As the number of successful attacks increases, the number of samples with changed logic units also increases. We then calculated their Pearson correlation, the positive correlation of which is 0.86, indicating that our logic units can reflect the robustness problem in the test.

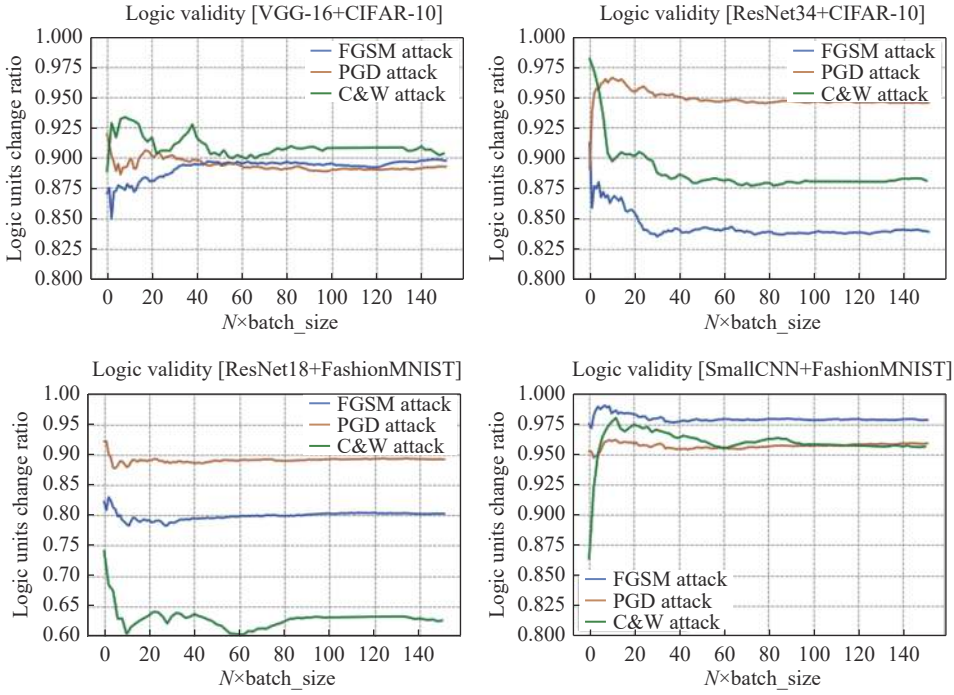



**Figure 9** The average change rate of logic units before and after the standard model against the sample attack.

## 6. Logic units guide test-case prioritization (RQ3)

1) State-of-the-art methods

Given a DNN model and a set of test cases, our aim is to prioritize test cases based on metrics and find as many hidden defects of the model as possible in a short time. Consequently, the proposed method was compared with several SOTA test priority techniques:

- DeepGini [9]: DeepGini is one of the most advanced test case prioritization techniques. The ranking criterion of this method is the classification decision of statistical probability distribution based on the DNN model.
- MCP [30]: In addition to the output confidence, MCP also considers the balance among different class boundaries of the selected test inputs.
- DSC [31]: It proposes to use the distance-based surprise score DSC as a test input prioritization metric.
- Entropy [29]: Similar to DeepGini, the ranking criterion of this method is the classification decision of statistical probability distribution based on the DNN model.
- NAC [6]: NAC (neuron activation coverage) digs for test inputs that make a group of DNN models with the same function produce differential behaviors and achieve high NC.
- NBC [7]: NBC measures how many corner-case regions have been covered by the given test input set $T$. It is defined as the ratio of the number of covered corner cases and the total number of corner cases.
- Random: Test cases are randomly selected for testing.

The configurable parameters in the above methods were set to the default values in the original paper. Each comparison experiment was then conducted using four models. A test-case set consisting of clean and adversarial samples was used, and APFD and RDA metrics were applied for evaluation.

2) Experimental procedure

Based on the discussion in Section IV, the test-case prioritization experiment was divided into two stages, that is, attribute extraction and prioritization. First, the training samples were input into the model to extract the CLUs of various categories using context attributes. Adversarial attack methods were applied to test samples to generate adversarial samples and form a new test set. The model was then input to calculate the SLUs of each use case and extract the intrinsic attributes, the SLUs

being used to calculate the logic degree. Subsequently, we created multiple attribute table entries, including logic degree, test time, classification accuracy, and defect severity, prioritized according to the sorting rules, and finally evaluated the defect detection ability.

3) Experimental results and analysis

The comparative experimental results for different DNN models on the CIFAR-10 dataset are shown in Table 5. Comparing the average APFD values of each method under different models, NAC and DSC are less effective, with a fault detection rate of 51%, close to the random method. NBC and MCP are generally effective, with APFD values of 82% and 72%, respectively. Entropy and DeepGini methods show superior performance and achieve around 91% APFD value. In comparison, the proposed DeepLogic is the best, with an APFD value of 95%. Comparing the average RDA values of each method under different models, NAC is less effective, with an RDA value of 22.38 close to the random method. NBC, MCP, and DSC are comparatively effective, with RDA values of 7.31, 19.30, and 10.22, respectively. Entropy and DeepGini have similar performance, with 11.01 RDA, and the proposed DeepLogic also illustrates the best result with an RDA value of 3.47.

**Table 5** Compare the effectiveness of various test prioritization methods from multiple perspectives

| Metrics | Model+Dataset | APFD | Average | RDA | Average |
|---|---|---|---|---|---|
| DeepGini | ResNet18+FashionMNIST | 0.90 | 0.91 | 11.64 | 11.01 |
| | VGG-16+CIFAR-10 | 0.92 | | 10.94 | |
| | ResNet34+CIFAR-10 | 0.92 | | 10.44 | |
| DSC | ResNet34+CIFAR-10 | 0.50 | 0.51 | 10.68 | 10.22 |
| | ResNet18+FashionMNIST | 0.51 | | 8.94 | |
| | VGG-16+CIFAR-10 | 0.51 | | 11.06 | |
| Entropy | ResNet34+CIFAR-10 | 0.92 | 0.91 | 10.18 | 10.68 |
| | ResNet18+FashionMNIST | 0.90 | | 11.20 | |
| | VGG-16+CIFAR-10 | 0.92 | | 10.66 | |
| MCP | ResNet34+CIFAR-10 | 0.64 | 0.72 | 21.87 | 19.30 |
| | ResNet18+FashionMNIST | 0.72 | | 19.36 | |
| | VGG-16+CIFAR-10 | 0.79 | | 16.66 | |
| NAC | ResNet18+FashionMNIST | 0.53 | 0.51 | 21.87 | 22.38 |
| | VGG-16+CIFAR-10 | 0.55 | | 19.27 | |
| | ResNet34+CIFAR-10 | 0.45 | | 23.81 | |
| NBC | VGG-16+CIFAR-10 | 0.81 | 0.82 | 7.24 | 7.31 |
| | ResNet18+FashionMNIST | 0.82 | | 8.21 | |
| | ResNet34+CIFAR-10 | 0.82 | | 6.48 | |
| Random | ResNet34+CIFAR-10 | 0.50 | 0.50 | 18.25 | 21.15 |
| | VGG-16+CIFAR-10 | 0.50 | | 19.58 | |
| | ResNet18+FashionMNIST | 0.49 | | 25.61 | |
| **DeepLogic** | VGG-16+CIFAR-10 | **0.96** | **0.95** | **3.37** | **3.47** |
| | ResNet18+FashionMNIST | **0.94** | | **4.26** | |
| | ResNet34+CIFAR-10 | **0.96** | | **2.78** | |

Additionally, we plotted the cumulative error curves of the above eight test criteria on the four DNN models and two datasets, as shown in Figure 10(a), (b), and (c). The horizontal axis of the cumulative error curve represents the number of test cases tested after prioritizing the test set, and the vertical axis represents the cumulative number of errors. The short red dotted line represents the cumulative error curve in the ideal state, that is, the test cases that trigger deep learning system errors are ranked in front of the test set. The closer the cumulative error curve after ranking to the cumulative error curve in the ideal state, the better the method.

From Figure 10(b), it can be seen that when testing 25% of the test cases, the average defect rates of the

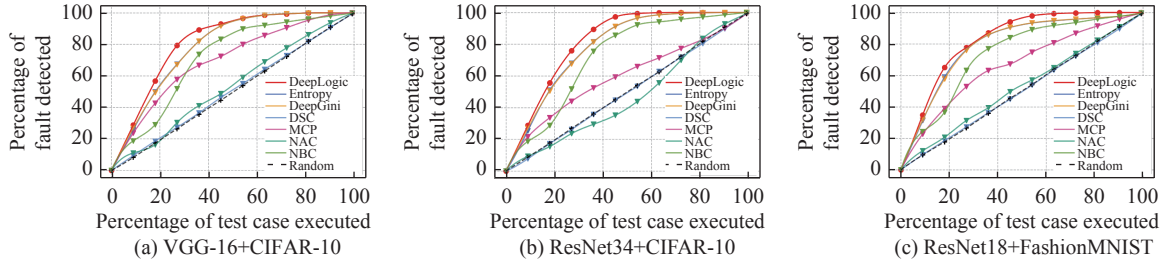



**Figure 10** The cumulative error curves of the eight test criteria on the three DNN models and two datasets. *X*-axis is the percentage of prioritized tests and *Y*-axis denotes the percentage of detected misclassified tests.

NAC, NBC, MCP, and DSC methods are 26%, 49%, 58%, and 26%, respectively. Compared with the more advanced entropy, and DeepGini (orange triangle line) methods (68% of fault detected), the proposed DeepLogic (red dotted line) method also exhibits higher effectiveness and can detect 80% of the model defects. The APFD value of the DeepLogic model is much higher than that of the NBC, entropy, and MCP methods. It can detect 95% of the model defects. Additionally, the DeepLogic model outperforms the SOTA DeepGini method with a higher APFD value. In addition, we can see that all the methods except NAC illustrate better performance than the random one. The number of test cases that reach a higher APFD is similar for different models. For VGG-16, when testing 55% of the test cases, the APFD value is 96%. For ResNet34, when testing 46% of the test cases, the APFD value is 98%. For ResNet18, when testing 50% of the test cases, the APFD value is 95%. We also find that the prioritization of test cases based on the Gini coefficient at the output layer and the prioritization based on the entropy function at the output layer almost overlap because these two methods are similar. Compared with the other state-of-the-art methods, our DeepLogic is closer to the accumulated error curve under the ideal condition, indicating that the DeepLogic is superior to other methods, with a better sorting effect and higher error detection rate.

## 7. Ablation studies

1) Impact of activation threshold

As illustrated in Section III.2 (equation (1)), we used the hyperparameter threshold $\sigma$ to determine whether the neural unit is activated. The effect of this activation threshold on the performance of the logic unit test is evaluated in this section. Specifically, six different activation thresholds of $\sigma = 7\times10^{-2}$, $\sigma = 10\times10^{-2}$, $\sigma = 13\times10^{-2}$, $\sigma = 15\times10^{-2}$, $\sigma = 18\times10^{-2}$, and $\sigma = 21\times10^{-2}$ were used for the experiments. Moreover, we used the variance to measure the overall difference in the percentage of frequency activation between each class in the plateau and trailing periods. The experimental results are summarized in Table 6.

**Table 6** Effect of different activation thresholds on frequency activation percentage analysis

| Activation threshold | Activation frequency dip start point | Platform period variance | Trailing period variance |
|---|---|---|---|
| 7E−2 | 104 | 0.078 | 0.293 |
| 10E−2 | 101 | 0.097 | 0.260 |
| 13E−2 | 99 | 0.120 | 0.207 |
| 15E−2 | 91 | 0.161 | 0.150 |
| 18E−2 | 89 | 0.193 | 0.112 |
| 21E−2 | 87 | 0.220 | 0.894 |

We further selected three activation thresholds for visualized analysis. The experimental results are shown in Figure 11.

From Table 6 and Figure 11, we can see that if the threshold $\sigma$ increases, the starting point of the neural unit plunge gradually approaches zero. Conversely, if the threshold $\sigma$ decreases, the starting point of the neural unit plunge gradually approaches the total number of channels (e.g., 512). However, if $\sigma$ is substantially large, e.g., $\sigma = 21\times10^{-2}$, a large difference arises in the percentage of frequency activation of each class in the plateau period. In contrast, if $\sigma$ is minimal, e.g., $\sigma = 7\times10^{-2}$, a large difference arises in the percentage of frequency activation of each class in the trailing period after the plunge point. The threshold $\sigma$ also indirectly affects the size of the threshold $k$ for logic unit selection because the threshold $k$ must be smaller than the starting point of the frequency plunge. Therefore, we selected $\sigma = 13\times10^{-2}$ in this paper to achieve better performance of DeepLogic.

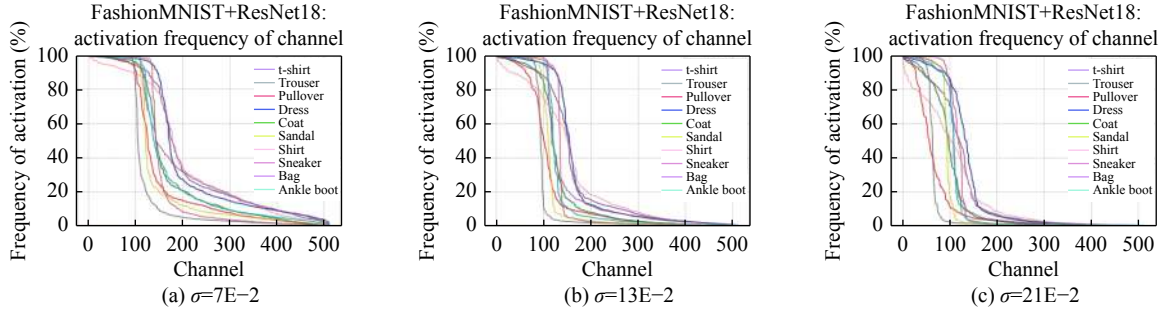



**Figure 11** Effect of different activation thresholds on frequency activation percentage analysis.

2) Impact of top-$k$ value

The impact of the $k$-value on the performance of the logic unit test was also examined. Specifically, the effect of four different $k$-values, namely $k$ = 20, 30, 40, and 50, on the percentage change of the logic unit before and after the adversarial attack was compared (CIFAR-10 +VGG-16). The experimental results are summarized in Table 7.

As can be seen from Table 7, the selection of logical unit with different top-$k$ values has a slight effect on the experimental results. This is further corroborated by (6) and (7), which indicate that altering both SLU and CLU

**Table 7** Effect of top-$k$ on the average change rate of logic units after the adversarial sample attack

| Top-$k$ of logic unit | FGSM | PGD20 | C&W |
|---|---|---|---|
| Top-20 | 0.861 | 0.874 | 0.865 |
| Top-30 | 0.892 | 0.906 | 0.907 |
| Top-40 | 0.816 | 0.860 | 0.802 |
| Top-50 | 0.909 | 0.894 | 0.913 |

simultaneously has a negligible impact on the percentage change of logic units before and after the adversarial attack. We selected $k = 50$ in this paper.

## VI. Conclusions

This paper has defined interpretable logic units and proposed a DeepLogic framework for priority testing of DNN models from the perspective of model interpretability. The effectiveness of two types of interpretable logic units, SLU and CLU, for testing the robustness of both standard and defended DNN models has been explored. By integrating context attributes and inner attributes, DeepLogic is able to perform effective priority testing. Experimental results on several datasets with multiple DNN models demonstrate that DeepLogic significantly outperforms several baseline and state-of-the-art methods. Since deep learning testing usually requires a large number of labels, we can benefit from the proposed prioritization testing method even if the labeling process is halted due to resource limits. This study currently only targets DNNs for classification, and its generalization capabilities for other DNN-based tasks, such as clustering and detection, will be investigated in the future.

## Acknowledgements

This work was supported by the National Key Research and Development Program of China (Grant No. 2020AAA0107702), the National Natural Science Foundation of China (Grant Nos. 62006181, 62161160337, 62132011, U21B2018, U20A20177, and 62206217), and the Shaanxi Province Key Industry Innovation Program (Grant No. 2021ZD LGY01-02).

## References

[1] K. Eykholt, I. Evtimov, E. Fernandes, *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 1625–1634, 2018.

[2] X. J. Ma, Y. H. Niu, L. Gu, *et al.*, "Understanding adversarial attacks on deep learning based medical image analysis systems," *Pattern Recognition*, vol. 110, article no. 107332, 2021.

[3] K. D. Julian, J. Lopez, J. S. Brush, *et al.*, "Policy compression for aircraft collision avoidance systems," in *Proceedings of the 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sacramento, CA, USA, pp. 1–10, 2016.

[4] K. Eykholt, I. Evtimov, E. Fernandes, *et al.*, "Physical adversarial examples for object detectors," in *Proceedings of the 12th USENIX Conference on Offensive Technologies*, Baltimore, MD, USA, p. 1, 2018.

[5] J. M. Zhang, M. Harman, L. Ma, *et al.*, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 1–36, 2022.

[6] K. X. Pei, Y. Z. Cao, J. F. Yang, *et al.*, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*, Shanghai, China, pp. 1–18, 2017.

[7] L. Ma, F. Juefei-Xu, F. Y. Zhang, *et al.*, "DeepGauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, Montpellier, France, pp. 120–131, 2018.

[8] Y. C. Sun, X. W. Huang, D. Kroening, *et al.*, "Testing deep neural networks," *arXiv preprint*, arXiv: 1803.04792, 2018.

[9] Y. Feng, Q. K. Shi, X. Y. Gao, *et al.*, "DeepGini: Prioritizing massive tests to enhance the robustness of deep neural networks," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, Virtual Event, pp. 177–188, 2020.

[10] Y. Z. Dong, P. X. Zhang, J. Y. Wang, *et al.*, "There is limited correlation between coverage and robustness for deep neural networks," *arXiv preprint*, arXiv: 1911.05904, 2019.

[11] D. Wang, Z. Y. Wang, C. R. Fang, *et al.*, "DeepPath: Path-driven testing criteria for deep neural networks," in *Proceedings of the 2019 IEEE International Conference on Artificial Intelligence Testing (AITest)*, Newark, CA, USA, pp. 119–120, 2019.

[12] T. W. Weng, H. Zhang, P. Y. Chen, *et al.*, "Evaluating the robustness of neural networks: An extreme value theory approach," in *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.

[13] G. Katz, C. Barrett, D. L. Dill, *et al.*, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proceedings of the 29th International Conference on Computer Aided Verification*, Heidelberg, Germany, pp. 97–117, 2017.

[14] T. Gehr, M. Mirman, D. Drachsler-Cohen, *et al.*, "AI2: Safety and robustness certification of neural networks with abstract interpretation," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, pp. 3–18, 2018.

[15] Z. N. Li, X. X. Ma, C. Xu, *et al.*, "Structural coverage criteria for neural networks could be misleading," in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, Montreal, QC, Canada, pp. 89–92, 2019.

[16] F. Harel-Canada, L. X. Wang, M. A. Gulzar, *et al.*, "Is neuron coverage a meaningful measure for testing deep neural networks?" in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Virtual Event, pp. 851–862, 2020.

[17] J. Y. Wang, J. L. Chen, Y. C. Sun, *et al.*, "RobOT: Robustness-oriented testing for deep learning systems," in *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, Madrid, Spain, pp. 300–311, 2021.

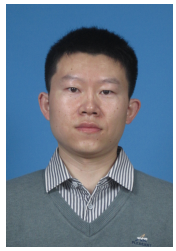
[18] Q. Hu, Y. J. Guo, M. Cordy, *et al.*, "An empirical study on data distribution-aware test selection for deep learning enhancement," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 4, article no. 78, 2022.

[19] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Proceedings of the 2nd International Conference on Learning Representations*, Banff, AB, Canada, 2014.

[20] B. L. Zhou, A. Khosla, A. Lapedriza, *et al.*, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recogni-*

*tion*, Las Vegas, NV, USA, pp. 2921–2929, 2016.
[21] R. R. Selvaraju, M. Cogswell, A. Das, *et al.*, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, pp. 618–626, 2017.
[22] D. Bau, J. Y. Zhu, H. Strobelt, *et al.*, "Understanding the role of individual units in a deep neural network," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 117, no. 48, pp. 30071–30078, 2020.
[23] Y. Bai, Y. Y. Zeng, Y. Jiang, *et al.*, "Improving adversarial robustness via channel-wise activation suppressing," in *Proceedings of the 9th International Conference on Learning Representations*, Virtual Event, pp. 1—19, 2021.
[24] S. C. Han, C. H. Lin, C. Shen, *et al.*, "Interpreting adversarial examples in deep learning: A review," *ACM Computing Surveys*, vol. 55, no. 14s, article no. 328, 2023.
[25] G. Rothermel, R. H. Untch, C. Y. Chu, *et al.*, "Prioritizing test cases for regression testing," *IEEE Transactions on Software Engineering*, vol. 27, no. 10, pp. 929–948, 2001.
[26] J. M. Kim and A. Porter, "A history-based test prioritization technique for regression testing in resource constrained environments," in *Proceedings of the 24th International Conference on Software Engineering*, Orlando, FL, USA, pp. 119–129, 2002.
[27] Z. Li, M. Harman, and R. M. Hierons, "Search algorithms for regression test case prioritization," *IEEE Transactions on Software Engineering*, vol. 33, no. 4, pp. 225–237, 2007.
[28] D. Leon and A. Podgurski, "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases," in *Proceedings of the 14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003*, Denver, CO, USA, pp. 442–453, 2003.
[29] M. Tyagi and S. Malhotra, "An approach for test case prioritization based on three factors," *International Journal of Information Technology and Computer Science*, vol. 7, no. 4, pp. 79–86, 2015.
[30] W. J. Shen, Y. H. Li, L. Chen, *et al.*, "Multiple-boundary clustering and prioritization to promote neural network retraining," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, Melbourne, VIC, Australia, pp. 410–422, 2020.
[31] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, Montreal, QC, Canada, pp. 1039–1049, 2019.
[32] A. Sharif, D. Marijan, and M. Liaaen, "DeepOrder: Deep learning for test case prioritization in continuous integration testing," in *Proceedings of the 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Luxembourg, Luxembourg, pp.525–534, 2021.
[33] Y. Li, M. Li, Q. X. Lai, *et al.*, "TestRank: Bringing order into unlabeled test instances for deep learning tasks," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Virtual Event, pp. 20874–20886, 2021.
[34] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, pp. 1520–1528, 2015.
[35] M. Lin, Q. Chen, and S. C. Yan, "Network in network," *arXiv preprint*, arXiv: 1312.4400, 2013.
[36] A. Mor, "Evaluate the effectiveness of test suite prioritization techniques using APFD metric," *IOSR Journal of Computer Engineering*, vol. 16, no. 4, pp. 47–51, 2014.
[37] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA, USA, pp. 448—456, 2015.
[38] A. Madry, A. Makelov, L. Schmidt, *et al.*, "Towards deep learning models resistant to adversarial attacks," in *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, BC, Canada, pp. 1—18, 2018.
[39] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, pp. 39–57, 2017.
[40] H. Y. Zhang, Y. D. Yu, J. T. Jiao, *et al.*, "Theoretically principled trade-off between robustness and accuracy," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, USA, pp. 7472–7482, 2019.

**Chenhao LIN** received the B.E. degree in automation from Xi'an Jiongtong University, Xi'an, China, in 2011, the M.S. degree in electrical engineering from Columbia University, New York, USA, in 2013, and the Ph.D. degree from The Hong Kong Polytechnic University, Hong Kong, China, in 2018. He is currently a Research Fellow at the Xi'an Jiongtong University. His research interests include artificial intelligence security, adversarial attack and robustness, identity authentication, and pattern recognition.
(Email: linchenhao@xjtu.edu.cn)



**Xingliang ZHANG** received the B.E. degree from the Information Engineering University, Zhengzhou, China. He is currently pursuing the M.S. degree in cyberspace security with Xi'an Jiaotong University, Xi'an, China. His current research interest focuses on artificial intelligence security.
(Email: zhangxliang@stu.xjtu.edu.cn)

**Chao SHEN** received the B.S. degree in automation and Ph.D. degree in control theory and control engineering from Xi'an Jiaotong University, Xi'an, China, in 2007 and 2014, respectively. He is currently a Professor with the Faculty of Electronic and Information Engineering, Xi'an Jiaotong University. His current research interests include AI security, insider/intrusion detection, behavioral biometrics, and measurement and experimental methodology.
(Email: chaoshen@xjtu.edu.cn)