Can Coverage Criteria Guide Failure Discovery for Image Classifiers? An Empirical Study

ZHIYU WANG, SIHAN XU, and LINGLING FAN, College of Cyber Science, DISSec, Nankai University, Tianjin, China
XIANGRUI CAI and LINYU LI, College of Computer Science, TKLNDST, Nankai University, Tianjin, China
ZHELI LIU, College of Cyber Science, DISSec, Nankai University, Tianjin, China

Quality assurance of deep neural networks (DNNs) is crucial for the deployment of DNN-based software, especially in mission- and safety-critical tasks. Inspired by structural white-box testing in traditional software, many test criteria have been proposed to test DNNs, i.e., to exhibit erroneous behaviors by activating new test units that have not been covered, such as new neurons, values, and decision paths. Many studies have been done to evaluate the effectiveness of DNN test coverage criteria. However, existing empirical studies mainly focused on measuring the effectiveness of DNN test criteria for improving the adversarial robustness of DNNs, while ignoring the correctness property when testing DNNs. To fill in this gap, we conduct a comprehensive study on 11 structural coverage criteria, 6 widely-used image datasets, and 9 popular DNNs. We investigate the effectiveness of DNN coverage criteria over natural inputs from four aspects: (1) the correlation between test coverage and test diversity; (2) the effects of criteria parameters and target DNNs; (3) the effectiveness to prioritize in-distribution natural inputs that lead to erroneous behaviors; and (4) the capability to detect out-of-distribution natural samples. Our findings include: (1) For measuring the diversity, coverage criteria considering the relationship between different neurons are more effective than coverage criteria that treat each neuron independently. For instance, the neuron-path criteria (i.e., SNPC and ANPC) show high correlation with test diversity, which is promising to measure test diversity for DNNs. (2) The hyper-parameters have a big influence on the effectiveness of criteria, especially those relevant to the granularity of test criteria. Meanwhile, the computational complexity is one of the important issues to be considered when designing deep learning test coverage criteria, especially for large-scale models. (3) Test criteria related to data distribution (i.e., LSA and DSA, SNAC, and NBC) can be used to prioritize both in-distribution natural faults and out-of-distribution inputs. Furthermore, for OOD detection, the boundary metrics (i.e., SNAC and NBC) are also effective indicators with lower computational costs and higher detection efficiency compared with LSA and DSA. These findings motivate follow-up research on scalable test coverage criteria that improve the correctness of DNNs.

This work was supported by the National Natural Science Foundation of China (Nos. 62202245 and U22B2048), the Natural Science Foundation of Tianjin, China (No. 23JCQNJC01960), and the Fundamental Research Funds for the Central Universities, Nankai University (No. 63241565).
Authors' Contact Information: Zhiyu Wang, College of Cyber Science, DISSec, Nankai University, Tianjin, China; e-mail: wangzhiyu@mail.nankai.edu.cn; Sihan Xu (Corresponding author), College of Cyber Science, DISSec, Nankai University, Tianjin, China; e-mail: xusihan@nankai.edu.cn; Lingling Fan, College of Cyber Science, DISSec, Nankai University, Tianjin, China; e-mail: linglingfan@nankai.edu.cn; Xiangrui Cai, College of Computer Science, TKLNDST, Nankai University, Tianjin, China; e-mail: caixr@nankai.edu.cn; Linyu Li, College of Computer Science, TKLNDST, Nankai University, Tianjin, China; e-mail: 2120210452@mail.nankai.edu.cn; Zheli Liu, College of Cyber Science, DISSec, Nankai University, Tianjin, China; e-mail: liuzheli@nankai.edu.cn.

ACM 1557-7392/2024/9-ART190
https://doi.org/10.1145/3672446

CCS Concepts: • **Software and its engineering → Software maintenance tools**;

Additional Key Words and Phrases: Deep learning testing, test coverage criteria

ACM Reference format:
Zhiyu Wang, Sihan Xu, Lingling Fan, Xiangrui Cai, Linyu Li, and Zheli Liu. 2024. Can Coverage Criteria Guide Failure Discovery for Image Classifiers? An Empirical Study. *ACM Trans. Softw. Eng. Methodol.* 33, 7, Article 190 (September 2024), 28 pages.
https://doi.org/10.1145/3672446

1 Introduction

Deep learning (DL) has experienced spectacular progress in a variety of real-world applications, such as speech recognition [12], computer vision [98, 99], and games [69]. With the rapid development of DL techniques, they are increasingly applied in mission- and safety-critical tasks. Unfortunately, the occurrence of safety issues has raised a big concern about the reliability of DL systems, especially when deployed in safety-sensitive scenarios, such as autonomous driving [6, 54] and medical treatments [53].

In the software engineering community, there is a growing trend to test [47, 62], analyze [20], and debug [48] DL systems for safety assurance. Inspired by the successful experiences of testing traditional software, many well-established techniques have been adapted to test **deep neural network (DNN)**-based software, such as metamorphic testing [75, 97], mutation testing [57], and fuzzing [62, 86]. Inspired by the white-box testing, many researchers proposed to test the internal states of DL systems [21, 25, 40, 55, 56, 58, 64, 73, 75, 77, 86]. Multiple test coverage criteria have been proposed specialized for DL systems, e.g., **neuron coverage (NC)** [64], **Modified Condition (MC)/Decision Coverage (DC)** variants [73], ***k*-multisection NC (KMNC)** [56], **importance-driven coverage (IDC)** [25], and **surprise coverage (SC)** [40]. These coverage criteria are typically designed based on neurons, the smallest units in DNNs, and are used as guidance to generate tests that may expose erroneous behaviors [75, 86]. The intuition behind is that one can hardly find hidden defects regarding certain neurons if no tests activate these neurons. Based on these criteria, some studies apply coverage-guided fuzzing on DL systems to produce large amounts of failures [62, 86, 97].

Despite the progress of DL testing, the practical use of structural coverage criteria for DL systems is ambiguous. A few studies started to investigate the correlation between structural coverage and the robustness of DL systems. Specifically, Li et al. [46] argued that structural coverage criteria could be misleading since the failures generated by coverage-oriented fuzzing are similar to adversarial samples. Harel-Canada et al. [28] found that NC is not strongly correlated with the success rate of adversarial attacks. Dong et al. [18] conducted a correlation analysis between structural coverage metrics and robustness metrics, and concluded that there is a limited correlation between structural coverage and the robustness of DNN-based software. Yan et al. [88] found that adversarial samples may not lead to high coverage, and coverage-oriented adversarial attacks usually add human visible perturbations compared to existing attacks.

Robustness and correctness are two different DNN properties which are relevant to erroneous behaviors and thus should be tested before deployment [95]. However, previous empirical studies on DNN test criteria mainly focused on measuring the effectiveness of these criteria for improving the adversarial robustness of DNNs, while ignoring the correctness property when testing DNNs [18, 46, 88, 90]. In this article, we present the first empirical study toward the practical use of 11 DNN test coverage criteria on testing model correctness when faced with natural inputs. Since DNN coverage criteria were inspired by structural coverage testing in traditional software, two natural research questions are whether high coverage indicate high test input diversity and how

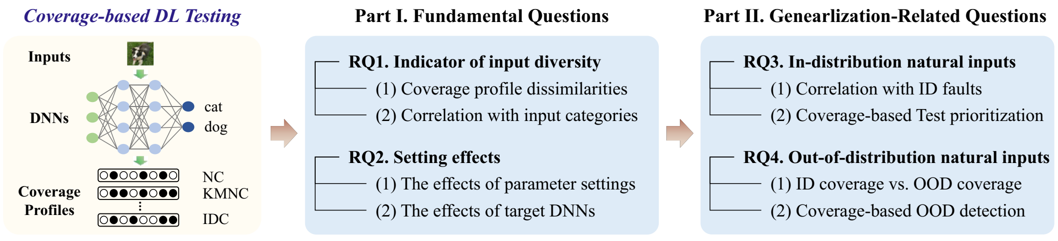
Coverage-based DL Testing
Inputs
DNNs
cat
dog
Coverage Profiles
NC
KMNC
IDC
Part I. Fundamental Questions
RQ1. Indicator of input diversity
(1) Coverage profile dissimilarities
(2) Correlation with input categories
RQ2. Setting effects
(1) The effects of parameter settings
(2) The effects of target DNNs
Part II. Genearlization-Related Questions
RQ3. In-distribution natural inputs
(1) Correlation with ID faults
(2) Coverage-based Test prioritization
RQ4. Out-of-distribution natural inputs
(1) ID coverage vs. OOD coverage
(2) Coverage-based OOD detection

Fig. 1. Overview of our study.

existing DNN criteria are affected by their hyper-parameters. As shown in Figure 1, we first answer two fundamental questions:

— *RQ1*: Are DNN coverage criteria useful indicators of the input diversity from the perspective of the distances between test inputs?
— *RQ2*: What are the effects of criteria parameters and target models on the effectiveness of DNN coverage criteria?

Based on the answers to RQ1 and RQ2, we further investigate the effectiveness of DNN coverage criteria on detecting natural inputs that lead to erroneous behaviors. We categorize such natural faults into two groups, i.e., **in-distribution (ID)** faults and **out-of-distribution (OOD)** samples. Aiming at the two types of natural faults, we further investigate the effectiveness of test criteria on two applications, i.e., test prioritization and OOD detection. Specifically, we answer the following questions:

— *RQ3*: Are structural coverage criteria useful metrics to detect ID natural faults and guide test prioritization?
— *RQ4*: Can structural coverage criteria be used to detect OOD samples?

Through the answers to RQ3 and RQ4, we study whether existing test criteria can be used to detect natural faults and thus improve the generalization ability of DNNs. Since previous studies on these criteria conducted most experiments on DNN-based image classifiers, to conduct a fair comparison, we exploited nine popular image classifiers and six image datasets that have been widely used in previous studies [18, 23, 25, 40, 56, 62, 86]. With an in-depth analysis of 11 structural coverage criteria, we aim to characterize the practical use and limits of these criteria, and find that (1) some criteria (i.e., NC, KMNC, **Top-*k* NC (TKNC)**, and IDC) can be used to evaluate the diversity of natural inputs from the perspective of their distances, while having little correlation to the fault-detection capability of natural faults; (2) only the **surprise adequacy (SA)** metrics [40] can be used for test prioritization with the **average percentage of fault detection (APFD)** ranging from 62% to 99%, outperforming the state-of-the-art baselines; (3) both the SA metrics [40] and the boundary coverage metrics [56] are useful indicators of OOD samples; and (4) the hyper-parameters of existing criteria have a significant impact on their effectiveness, and the scalability of these criteria on medium- and large-scale DNNs is questionable. Our findings enable some follow-up research to construct trustworthy DNN-based software, e.g., confidence prediction, test prioritization, and the evaluation of test adequacy. Our study also reveals that structural coverage criteria based on the neuron values of a given test (e.g., NC [64]) are less effective than those based on the data distribution of the training and testing data (e.g., SA [40] and boundary coverage metrics [56]). Hence, it is desirable to propose more suitable and computationally feasible test criteria for DNN-based software.

In summary, we made the following contributions in this article:

— To the best of our knowledge, we present the first empirical study on DNN coverage criteria from the perspective of their capability over *natural inputs*.

— The findings reveal that despite some correlation with test diversity (i.e., the distances between test inputs), neuron-based criteria are less indicative of natural faults than criteria based on the distributions of training and test data. The hyper-parameters of criteria have a significant impact on their effectiveness. So far, test criteria with computational feasibility are still desirable.
— The findings of our study shed light on better criteria that are more scalable and suitable for DNN-based image classifiers, and enable follow-up research, such as confidence prediction, test prioritization, and the evaluation of test adequacy.

2 Background

In this section, we briefly review relevant preliminaries, including DNNs, test coverage criteria for DNNs, and OOD detection.

2.1 Image Classification

DNNs have exhibited their great power in learning high-level representations from raw inputs, e.g., images and videos. Unlike traditional software whose decision logic is explicitly written by developers, DNNs follow a data-driven learning procedure of machine learning techniques. With well-designed architectures, proper optimization strategies, and a large set of training data, a DNN is capable of learning high-dimensional distributions of input data and accomplishing prediction tasks.

Image classification is one of the most important application tasks of DNNs. With the development of DL, researchers have proposed a series of classical network structures. LeCun et al. [41] first proposed LeNet-5, which was the first structure to apply the convolution kernel to the DNN models. LeNet-family is the simplest convolutional neural network structure. LeNet-5, for example, which contains only three convolution layers, two pooling layers, and one fully connected layer, can still achieve 99% Accuracy on MNIST dataset. VGG [70] is one of the representatives of the deep convolution network, replacing the large convolution kernel with a combination of multiple small convolution kernels. This property not only reduces the model parameters but also increases the nonlinear transformation in the model. Resnet [29] is one of the representatives of module innovation. The residual blocks in ResNet benefit the training of deep networks with many layers. With the increasing application of neural networks, the lightweight model which requires less resources attracts more attention. For example, by applying the depthwise separable convolution, MobileNet [33] greatly reduced the number of model parameters and computational cost.

2.2 Test Coverage Criteria for DNNs

So far, structural test coverage has been widely adopted as an indicator of test adequacy in traditional software. Recently, there has been a growing interest in proposing coverage criteria specialized for DNNs. As neurons are the basic computing units of DNNs, the importance of neuron activation is highlighted to capture the internal states of DNNs. We present definitions and formulas for popular coverage criteria used in the study below.

Definition 2.1. Let $N = \{n_1, n_2, ...\}$ be a set of neurons of a DNN. Let $D = \{x_1, x_2, ...\}$ be the test cases in test set D, we use $\phi(x, n)$ to denote a function that returns the output of a neuron $n \in N$ under a given test input $x \in D$. Let DNN have l layers and L_i denote the set of neurons on the ith layer ($1 \leq i \leq l$). Giving a training set T, $high_n$ and low_n denote maximum and minimum values of $\phi(x, n)$ for all x in T, respectively.

Neuron-based criteria. Structural coverage criteria are proposed based on the outputs of neurons.

— NC [64], which measures the ratio of the number of activated neurons to the total number of neurons in a target DNN. A neuron is considered activated if its value is greater than a predefined threshold t. NC of a test suite D is computed as $NC(D, t) = \frac{|\{n|\exists x \in D: \phi(x,n) > t\}|}{|N|}$.
— KMNC [56], divides each neuron's value range into k sections, and measures how thoroughly the given set of test inputs D covers the range $[low_n, high_n]$. KMNC of a test suite D is computed as $KMNC(D, k) = \frac{\sum_{n \in N} |\{S_i^n | i \in [1,k], \exists x \in D: \phi(x,n) \in S_i^n\}|}{|N| \times k}$, where S_n^i denotes the set of values in the ith section for neuron n.
— TKNC [56], which measures the ratio of neurons that have once been one of the top-k active neurons in a layer. TKNC of a test suite D is computed as $TKNC(D, k) = \frac{|\{\bigcup_{x \in D} (\bigcup_{1 \le i \le l} top_k(x,i))|}{|N|}$, where i denotes the layer and $top_k(x, i)$ denotes the neurons that have the largest k outputs on that layer given a test input x.
— IDC [25], which identifies important neurons by the causal relationship between neurons and model behaviors, and measures the ratio of combinations of important neuron clusters covered by a test set. IDC of a test suite D is computed as $IDC = \frac{|\{INCC(j)| \exists x \in D: \forall V_n^i \in INCC(j) \cdot min(d(\phi(x,n)), V_n^i\}|}{|INCC|}$, where **important neuron cluster combinations (INCC)** contain all combinations of important neuron clusters. Let J denote the total number of these combinations. The term INCC(j) denotes the jth combination of important neuron clusters, where j ranges from 1 to J. The combination INCC(j) is covered if there exists a sample y in the test suite Y that covers all clusters in INCC(j).

Boundary coverage criteria. Inspired by boundary testing, some criteria measure the coverage of corner regions that are beyond the training set. The set of covered corner-case regions is defined as: $UpperCornerNeuron = \{n \in N | \exists x \in D : \phi(x, n) \in (high_n, +\infty)\}$; $LowerCornerNeuron = \{n \in N | \exists x \in D : \phi(x, n) \in (-\infty, lower_n)\}$.

— **Neuron boundary coverage (NBC)** [56], inspired by traditional boundary testing, measures the ratio of corner-case regions (i.e., values higher than the upper boundary or lower than the lower boundary) that have been covered. NBC of a test suite D is computed as $NBC = \frac{|UpperCornerNeuron| + |LowerCornerNeuron|}{2 \times |N|}$.
— **Strong neuron activation coverage (SNAC)** [56], which measures the ratio of covered corner-case regions only with regard to the upper boundary. SNAC is computed as $SNAC = \frac{|UpperCornerNeuron|}{|N|}$.

Surprise coverage criteria. SA [40] aims to measure the relative surprise of a new test with respect to the training set. It measures the difference between the behaviors of the input and the training set. Kim et al. [40] proposed two SA and adopted corresponding **SA criteria (SC)**.

Definition 2.2. Similar to KMNC, given the SA value of a test set D, SC can be calculated by dividing the range of training set SA into k buckets, We use U_{high}, U_{low} to represent the upper and lower bounds of the SA value of the training set. U_i denotes the set of values in the ith section, and $SA(x)$ denotes SA values for a test case x. SC for a set of inputs D is defined as $SC(D) = \frac{|\{i | \exists x \in D: SA(x) \in U_i\}|}{n}$.

— **Likelihood-based surprise coverage (LSC)** [40]. **Likelihood-based SA (LSA)** uses **kernel density estimation (KDE)** to estimate the probability density of each activation value, so as to obtain the surprise of a new input. LSA is computed as $LSA = -log\left(\frac{1}{|A_{N_L}(T)|} \sum_{x_i \in T} K_H(\alpha_{N_L}(x) - \alpha_{N_L}(x_i))\right)$. Unlike LSA which describes the SA of a single input, LSC computes the coverage over the LSA value regions for a test suite. Specifically, to calculate the LSC of a test suite, the

LSA values of all training inputs are first obtained. Then, the value range of LSA is divided into k segments. LSC measures the ratio of covered segments in the test suite.

— **Distance-based surprise coverage (DSC)** [40]. **Distance-based SA (DSA)** calculates the Euclidean distance [13] between the activation traces of a new test and the training set. DSA is computed as $DSA = \frac{\|\alpha_N(x)-\alpha_N(x_i)\|}{\|\alpha_N(x_a)-\alpha_N(x_b)\|}$. DSC divides the range of DSA values into k segments and measures the coverage over these DSA segments.

Neuron path coverage criteria. Inspired by the control flow graph in traditional software, neuron path coverage criteria measure the coverage of decision logic paths in DNNs. To obtain decision logic paths, these criteria identify important neurons and connect the important neurons in each layer to obtain critical decision path. These paths are further clustered into abstract paths, on this basis neuron path coverage criteria calculate the ratio of covered abstract paths.

— **Structure-based neuron path coverage (SNPC)** [85], which is designed based on the control-flow of the DNN (i.e., neurons in the path). Suppose $G_{f(x)} = \left\{\hat{p}_\beta^{f(x),1}, \dots, \hat{p}_\beta^{f(x),k}\right\}$ is the set of abstract paths that are extracted from clusters of the class $f(x)$. $b^l_{x,\hat{p}}$ is the covered bucket of the distance value between the test path and abstract paths. SNPC is defined as $SNPC(X) = \frac{|\{b^l_{x,\hat{p}} | \forall x \in D, \forall \hat{p} \in G_{f(x)}, \forall l \in f\}|}{n,k,|\hat{p}|,m}$.

— **Activation-based neuron path coverage (ANPC)** [85], which considers the dataflow, i.e., the neuron outputs in the path. ANPC is computed as $ANPC(X) = \frac{|\{d^l_{x,\hat{p}} | \forall x \in D, \forall \hat{p} \in G_{f(x)}, \forall l \in f\}|}{n,k,|\hat{p}|,m}$, where $d^l_{x,\hat{p}}$ denotes the distance, $d^l_{x,\hat{p}}$ is the weighted distance of the neuron activation value of $b^l_{x,\hat{p}}$.

Black-box coverage criteria. DL black-box coverage criteria are inspired by traditional black-box testing in software testing. As an evaluation metric that does not rely on DNN structure and has low testing cost, black-box coverage criteria is often based on the distribution of training data and testing data.

— **Manifold combination coverage (MCC)** [7], which maps the input space into a low-dimensional space via an encoder and decoder function, measures how well the low-dimensional space is covered by the test samples. The encoding function $q : \mathbb{X} \rightarrow \mathbb{R}^d$ is defined to encode the input space $\mathbb{X}$ into a d-dimensional space. Based on the bening function $q^d_k : \mathbb{R}^d \rightarrow \mathbb{Z}^d_k$, The k-section d-dimesional MCC of a test suite D is computed as $MCC(D, k, d) = |\{p^d_k(q(x)) | x \in D\}|/k^d$. Based on the experimental setup of Byun et al. [7], we set the hyperparameters k=6 and d=7 in all experiments.

2.3 OOD Detection

Traditional machine learning models are trained based on the close-world assumption, where the test samples are assumed to be **independent and identically distributed (I.I.D.)** to the training data, i.e., ID samples [22]. However, in an open-world scenario, tests can be OOD, which means the distribution of these test samples is different from that of the training data [19]. The distribution shifts are usually caused by semantic shifts [31] or covariate shifts [4]. As the OOD samples are usually unknown to the target model during inference, the model often fails to identify OOD samples and make correct predictions.

Detecting OOD samples is crucial for the safety of DL systems [76, 89]. Formally, let $\mathcal{X}$ and $\mathcal{Y}$ be the input space and the label space respectively for training a model. The data distribution (ID) is formulated as a joint probability distribution $P(x, y)$ over the space $(\mathcal{X} \times \mathcal{Y})$, where $x \in \mathcal{X}$ and

Table 1. Datasets, DNN Models, and Training Statistics

Dataset	DNN Model	#Neurons	#Layers	Accuracy	Optimizer	Init. LR	#Epochs
MNIST [14]	LeNet-1	26	7	98.56%	Adadelta	default	10
	LeNet-4	126	8	98.32%	Adadelta	default	10
	LeNet-5	246	9	98.93%	Adadelta	default	10
CIFAR-10 [1]	ResNet-20	1,194	20	84.16%	Adam	0.1	176
	ResNet-50	3,210	50	86.69%	Adam	0.1	155
	MobileNet	13,005	90	82.84%	Adam	0.1	161
CIFAR-100 [1]	VGG-13	7,524	13	41.09%	SGD	0.01	194
	VGG-16	10,084	16	44.11%	SGD	0.01	174
	VGG-19	12,644	19	51.29%	SGD	0.01	183

$y \in \mathcal{Y}$. OOD detection aims at detecting test samples that come from a shifted distribution from the ID samples, i.e., $P'(x, y) \neq P(x, y)$.

3 Empirical Study

In this section, as shown in Table 1, we conduct a comprehensive empirical study based on 6 widely-used datasets (i.e., MNIST, CIFAR-10, CIFAR-100, FashionMNIST, LSUN, and SVHN) and 9 popular image classifiers (i.e., LeNet-1, LeNet-4, LeNet-5, ResNet-20, ResNet-50, MobileNet, VGG-13, VGG-16, and VGG-19) to investigate the practical use of 11 test criteria, aiming to answer the four research questions mentioned in the Introduction. All the experiments are conducted on an Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz and an NVIDIA Quadro RTX 8000 GPU.

3.1 Data Preparation

The detailed information of datasets and DNN models used in our study is shown in Table 1. For RQ1, RQ2, and RQ3, we implemented nine DNN models on three popular image datasets (i.e., MNIST, CIFAR-10, and CIFAR-100), which have been widely-used by previous studies [56, 86]. MNIST is a widely-used recognition dataset, which contains 60,000 training samples and 10,000 test samples in 10 classes (digits from 0 to 9). CIFAR-10 and CIFAR-100 are two labeled subsets of Tiny ImageNet [2]. CIFAR-10 consists of 60,000 general color images in 10 classes, where 50,000 are for training and 10,000 are for testing. CIFAR-100 has 100 classes, each of which contains 500 training images and 100 testing images.

To conduct a fair comparison, we follow previous studies that proposed or evaluated the same test coverage criteria as this article [3, 5, 23, 25, 28, 36, 56, 64, 73, 79, 85, 86, 88]. Specifically, for the MNIST dataset, we use three LeNet models (i.e., LeNet-1, LeNet-4, and LeNet-5) to represent the simple linear models. For the CIFAR-10 dataset, we use two ResNet models (i.e., ResNet-20 and ResNet-50) to represent residual neural networks and use MobileNet to represent low-parameter models which are more suitable for embedded applications. For the CIFAR-100 dataset, we use three VGG models (i.e., VGG-13, VGG-16, and VGG-19) to represent models with small convolutional kernels.

As shown in Table 1, we trained LeNet-1, LeNet-4, and LeNet-5 for 10 epochs with a batch size of 1,024, and used the Adadelta optimizer with the cross-entropy loss. For ResNet-20, ResNet-50, and MobileNet, we set the initial learning rate to 0.1 and used the Adam optimizer with the ReduceLROnPlateau learning rate scheduler. The batch size was set to 1,024, and the maximum epoch was set to 200. We used EarlyStopping with the patience to be 20. We trained three VGG models with the **stochastic gradient decline (SGD)** optimizer and a multi-step learning rate scheduler. We also set the maximum training epoch to 200 and used EarlyStopping for training.

3.2 RQ1. Indicator of Test Diversity

As indicated by the fundamental assumption of structural testing, i.e., higher structural coverage means higher diversity in the test set [10], we first investigate whether DNN coverage is indicative of test diversity.

3.2.1 Coverage Profile Dissimilarities between Tests with Different Functionalities. To answer RQ1, we first investigate whether tests with different functionalities result in different coverage profiles. Due to the lack of explainability, the functionalities of DNNs can hardly be designated or numbered. To mitigate this problem, we assume that test inputs with similar contents (e.g., two images of airplanes) exercise similar functionalities of DNN-based image classifiers. Similarly, tests with different contents (e.g., an image of a cat *v.s* an image of an airplane) are assumed to exercise different functionalities of a DNN classifier.

We study 11 structural coverage criteria mentioned in Section 2.2 with three datasets (i.e., MNIST, CIFAR-10, and CIFAR-100) and 9 DNN models shown in Table 1. For the models on the MNIST and CIFAR-10 datasets, we randomly picked 1,000 pairs of tests for each category, and for the models on CIFAR-100, we chose 100 pairs of tests for each category. Thus we obtained 10,000 pairs of same-categorized tests in total for each model. To obtain tests with different functionalities, we randomly picked 10,000 pairs of tests and ensured that tests within each pair belonged to different categories. We measure the coverage profile dissimilarity between a pair of tests by the Jaccard distance as $\text{Distance}_{f(t_i,t_j)} = 1 - \frac{|f(t_i) \cap f(t_j)|}{|f(t_i) \cup f(t_j)|}$, where t_i and t_j are two tests, and $f(\cdot)$ represents the function that executes a test and attains a coverage profile with regard to a specific criterion. Specifically, given a test, a coverage profile is a set of covered test units, such as neurons, values, and decision paths, which are dependent on the type of test criterion. Given a test x, $f(x)$ represents a Boolean vector that encodes whether each test unit (e.g., a neuron, a value region, or a path) is covered or not by the test. We can see the smaller the **average Jaccard distances (AVD)** of an input pair is, the more similar coverage profiles they have. We set the hyper-parameters of coverage metrics mainly according to previous works. Specifically, following DeepXplore [64], we set the threshold of neuron activation to 0.75 for NC. Based on the experiment of Lei et al. [56], we report the ratio of neurons that have once been the most active ones in each layer for TKNC. For IDC, we follow the settings of Simos et al. [25] and set the number of important neurons in each layer to 6. For KMNC, LSC, DSC, SNPC, and ANPC, we set the number of sections k to 10. The reason is that when k is too large (e.g., larger than 500), the effectiveness of these test criteria decreases quickly. For MCC, we follow the settings of Byun et al. [7], where we set the dimension of the encoding space d to 7 and the number of sections k to 6. The details can be seen in Section 3.3.1.

Results. Table 2 displays the AVDs between the coverage profiles of tests. We use *"same"* and *"diff"* to distinguish AVDs between test pairs that belong to the *same* and *different* categories (i.e., AVD_{same} and AVD_{dif}, respectively). For SNAC and NBC, since tests randomly selected from the test sets rarely touch corner-case regions (i.e., higher than the upper boundary or lower than the lower boundary), causing SNAC/NBC to be 0 in most cases. Therefore, we did not show them in Table 2. Note that LSC for VGG-13 cannot be computed due to the singular matrix of the activation trace that prohibits the fitting of the Gaussian kernel function. Hence, we use "-" to denote the LSC values for VGG-13. We compute the experimental results of the MCC separately for each dataset, as the black-box coverage criterion MCC is solely influenced by the distribution of training data and the test suite, independent of the model.

Overall, it can be seen from Table 2 that for all DNNs and test criteria, AVD_{same} is smaller than AVD_{dif} in most cases. To conduct a statistical analysis, we generate a p-value from a t test for each model, which exhibits the significance of the results. It indicates that tests in the same category

Table 2. Average Jaccard Distance between Coverage Profiles of Tests

Criteria	Cate.	MNIST			CIFAR-10			CIFAR-100		
		LeNet-1	**LeNet-4**	**LeNet-5**	**ResNet-20**	**ResNet-50**	**MobileNet**	**VGG-13**	**VGG-16**	**VGG-19**
NC	Same	0.146	0.541	0.595	0.756	0.616	0.744	0.802	0.814	0.737
	diff	0.853	0.948	0.965	0.926	0.762	0.913	0.921	0.926	0.925
	p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
KMNC	Same	0.719	0.691	0.630	0.810	0.733	0.729	0.853	0.815	0.829
	diff	0.819	0.800	0.756	0.845	0.776	0.803	0.892	0.861	0.897
	p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TKNC	Same	0.233	0.456	0.406	0.720	0.705	0.900	0.819	0.881	0.835
	diff	0.704	0.816	0.774	0.786	0.780	0.961	0.911	0.928	0.942
	p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
IDC	Same	0.205	0.179	0.117	0.203	0.280	0.201	0.042	0.039	0.004
	diff	0.312	0.295	0.265	0.316	0.427	0.374	0.058	0.057	0.007
	p-value	9e−228	0.000	0.000	8e−227	0.000	0.000	6e−16	3e−17	3e−5
LSC	Same	0.586	0.528	0.517	0.726	0.713	0.378	-	0.034	0.114
	diff	0.684	0.695	0.645	0.744	0.742	0.421	-	0.034	0.117
	p-value	7e−131	3e−204	3e−222	6e−8	6e−26	3e−9	NaN	0.746	0.133
DSC	Same	0.614	0.670	0.675	0.817	0.804	0.793	0.497	0.557	0.519
	diff	0.676	0.752	0.747	0.821	0.810	0.808	0.508	0.568	0.539
	p-value	4e−106	1e−206	6e−163	5e−6	3e−13	4e−21	5e−13	4e−10	2e−35
SNPC	Same	0.662	0.702	0.694	0.877	0.901	0.863	0.933	0.944	0.937
	diff	0.999	0.999	0.999	0.945	0.967	0.941	0.982	0.987	0.984
	p-value	0.000	0.000	0.000	8e-184	0.000	8e-173	1e−152	3e−151	2e−164
ANPC	Same	0.518	0.617	0.693	0.852	0.834	0.838	0.887	0.899	0.895
	diff	0.999	0.999	0.999	0.937	0.947	0.931	0.967	0.974	0.973
	p-value	0.000	0.000	0.000	9e-113	1e-169	1e-115	2e−139	1e−134	5e−141
MCC	Same		0.834			0.876			0.902	
	diff		0.917			0.929			0.935	
	p-value		0.000			0.000			0.000	

tend to have similar coverage profiles compared to those in different categories. Meanwhile, we can also observe that different structural coverage criteria have different capabilities in distinguishing tests in different categories. For example, considering test coverage over LeNet-1, AVD$_{dif}$ is larger than AVD$_{same}$ by 0.7 for NC. However, for IDC in the same model, AVD$_{dif}$ is larger than AVD$_{same}$ by only 0.1. It indicates that in LeNet-1, tests from the same categories are more likely to activate similar neurons, while those from different categories may activate different neurons, resulting in a large gap between their NC coverage profiles. Similarly, for TKNC, the gap between AVD$_{same}$ and AVD$_{dif}$ is slightly smaller than NC. It indicates that tests from the same categories are more likely to share the most active neurons in each layer compared to tests from different categories. For two neuron path coverage, including SNPC and ANPC, the gap between AVD$_{same}$ and AVD$_{dif}$ is more than 0.4. It shows that test cases in similar categories do have similar coverage paths.

Compared with NC and TKNC, for KMNC, LSC, DSC, IDC and MCC, the AVD$_{same}$ and AVD$_{dif}$ values are close to each other in most cases, so that it is difficult to identify similar/different behaviors by these criteria. For instance, AVD$_{same}$ for KMNC ranges from 0.630 to 0.853, and AVD$_{dif}$ for KMNC ranges from 0.756 to 0.892. The results show that most test inputs have very different KMNC coverage profiles regardless of their categories. A possible reason may be that KMNC is *too fine-grained* (dividing each neuron into a number of sections), and thus can hardly capture the characteristics of different model behaviors. In contrast, the AVDs of IDC are much lower than those of KMNC. It indicates that most test inputs cover similar clusters of important

Table 3. Correlation between Coverage and #Input Categories

Dataset	Model	NC	KMNC	TKNC	IDC	NBC	SNAC	DSC	LSC	SNPC	ANPC	MCC
MNIST	LN-1	**0.70**	**0.87**	**0.76**	**0.37**	0.00	0.04	0.15	**0.37**	**0.98**	**0.90**	**0.67**
	LN-4	**0.82**	**0.86**	**0.87**	**0.52**	0.03	0.22	**0.40**	**0.42**	**0.98**	**0.93**	
	LN-5	**0.90**	**0.89**	**0.93**	**0.72**	0.00	0.02	0.05	0.08	**0.97**	**0.93**	
C-10	RN-20	**0.63**	**0.77**	**0.77**	**0.62**	0.01	0.07	0.14	0.05	**0.85**	**0.68**	**0.54**
	RN-50	**0.55**	**0.83**	**0.82**	**0.40**	0.00	0.07	0.24	0.12	**0.85**	**0.75**	
	MNet	**0.79**	**0.83**	**0.87**	**0.67**	0.04	0.12	0.03	0.11	**0.74**	**0.45**	

The correlation coefficients greater than 0.3 are in bold.

neurons, leading to very small AVDs. We repeated the experiment 30 times to avoid noise in the measurements. We found that for all test coverage criteria, the variances of Jaccard distances are smaller than 1e-3, which means the results are stable.

3.2.2 Correlation between Coverage and the Number of Input Categories. To further study whether test coverage is indicative of test diversity, we conduct a correlation analysis between coverage and the number of input categories. We conduct experiments on 11 criteria and 2 datasets that contain 10 classes (i.e., MNIST and CIFAR-10). The DNNs used for each dataset are shown in Table 1. We varied the number of input categories in test suites from 1 to 10. Specifically, given the number of input categories n, we randomly selected 1,000 test cases from n categories to form a test suite. Then, we obtained the test coverage and the number of input categories for correlation analysis. For example, in the case of five input categories, we first randomly selected five categories, and then randomly selected 200 test samples for each category to obtain 1,000 samples as the test suite. We repeated this process 30 times to avoid noise and ended up with 30 test suites for each number of input categories. Finally, we obtained 300 pairs of test coverage and the number of input categories for each DNN, which were finally used to calculate a correlation coefficient between test coverage and input categories for each DNN and each criterion.

We adopt three correlation analysis algorithms, i.e., Pearson product-moment correlation [63], Spearman's rank order correlation [71], and Kendall's τ rank correlation coefficient [39] to measure the correlations. Since there are no significant differences between the results of the three algorithms, we only present the results of Kendall's τ rank correlation coefficient [39], which has no requirement of the linear coefficient or the normal distribution of dataset like Pearson and Spearman's correlation coefficients. Note that the hyper-parameters of test criteria are the same as that in Section 3.2.1.

Results. Table 3 shows the correlation coefficients between test coverage and the number of input categories. The correlation coefficients greater than 0.3 are in bold. As MCC is independent of the model structure, the experimental results for MCC are specific to each dataset. It can be seen that for neuron-based criteria (i.e., NC, KMNC, TKNC, and IDC), neuron path criteria (i.e., SNPC and ANPC), and black-box criteria (i.e., MCC), test coverage is positively correlated with the number of input categories. However, for the boundary coverage criteria (i.e., SNAC and NBC) and the surprise coverage criteria (i.e., DSC and LSC), there is a limited correlation between coverage and the number of categories within same-sized test suites. The results indicate that neuron-based criteria are more indicative of test diversity than the criteria based on boundary or surprise coverage. A possible explanation is that the boundary and surprise coverage criteria focus on the distance between training and test sets, while other criteria aim to measure the test adequacy of neuron activities. For instance, KMNC measures the ratio of covered value segments within the range of the training set.

Answer to RQ1: Structural coverage criteria (i.e., KMNC, TKNC, NC, IDC, SNPC and ANPC) and black-box coverage criteria (i.e., MCC) have moderate to strong correlations with the diversity of *natural inputs*, and thus can be used to evaluate test diversity before deployment. However, for SNAC, NBC, DSC, and LSC, there is limited correlation between test coverage and test diversity.

3.3 RQ2. Setting Effects

Since many factors may affect the correlation between test coverage and the input diversity, to answer RQ2, we investigate the impact of these factors, including the hyper-parameters and target DNNs.

3.3.1 The Effects of Criteria Hyper-Parameters. To investigate the effects of parameter settings when distinguishing tests with different behaviors, we set different values to each hyper-parameter of test criteria. (1) For NC, we set the predefined activation threshold t to 0.1, 0.25, 0.5, 0.75, and 0.9, respectively. (2) For KMNC, we set k to 10, 50, 100, 200, 500, 1,000, and 10,000 to study the effects of parameter settings. (3) For TKNC, we set k to 1, 2, 3, 4, and 5, respectively. (4) For the surprise coverage metrics (i.e., LSC and DSC), we divided the range of SA values into 10, 50, 100, 200, 500, and 1,000 segments, and calculated the ratio of covered segments. (5) For the IDC, the hyper-parameter is the number of selected important neurons in the last layer. We set it to 4, 6, 8, 10, and 12. (6) For the boundary coverage metrics (i.e., SNAC and NBC), the values of the upper and lower boundary are determined by the training set, so there are no hyper-parameters for SNAC and NBC. (7) For the neuron path coverage metrics (i.e., SNPC and ANPC), we divided the range of distance values into 10, 50, 100, 200, 500, and 1,000 segments and calculated the path coverage. To make it clear, we use δ to denote the gap between the AJD of tests from the same and different categories, i.e., $\delta = \text{AVD}_{dif} - \text{AVD}_{same}$. The test suites are the same as that in Section 3.2.1.

Results. Figure 2 displays the values of δ for criteria with different configurations. *Note that for KMNC, DSC, and LSC with #sections larger than 500, the computations of test coverage are out-of-memory for all models except LeNet-1. The reason is that these criteria divide the range of neuron values into k sections so that given a test for a model with n neurons, its coverage profile is a matrix with $n \times k$ elements.* Hence, we do not show these results in Figure 2. To avoid the potential impact of training datasets, we compute the average δ for each dataset separately. The larger δ is, the greater difference in coverage profiles between tests from the same and different categories. We can see that δ varies a lot when hyper-parameters get changed. Specifically, for KMNC, DSC, SNPC, and ANPC when we increase the values of their hyper-parameters, δ decreases intensively. Especially when the number of sections is greater than 50, δ is close to 0, which means AVD_{same} and AVD_{dif} are very close to each other, and tests from the same and different categories are not distinguishable by test criteria. A possible reason is that the hyper-parameters of these criteria denote the number of sections that a neuron value should be divided into. When the range of activation values is divided into many segments, test criteria, such as KMNC, DSC, and LSC become more fine-grained, causing similar behaviors to have different coverage profiles. In this case, these criteria can hardly distinguish tests with different behaviors.

In most cases, we can see that δ increases with the increase of the threshold for NC, indicating that NC is more effective in capturing the differences between tests from the same and different categories with a higher threshold. Similarly, for TKNC, we can also see δ reaches the highest when the top-1 active neuron (with the highest value) in each layer is used for computing coverage. However, δ drops sharply when k is 2, and then slowly climbs up with more active neurons are considered. A possible explanation is that the most active neuron in each layer might be more

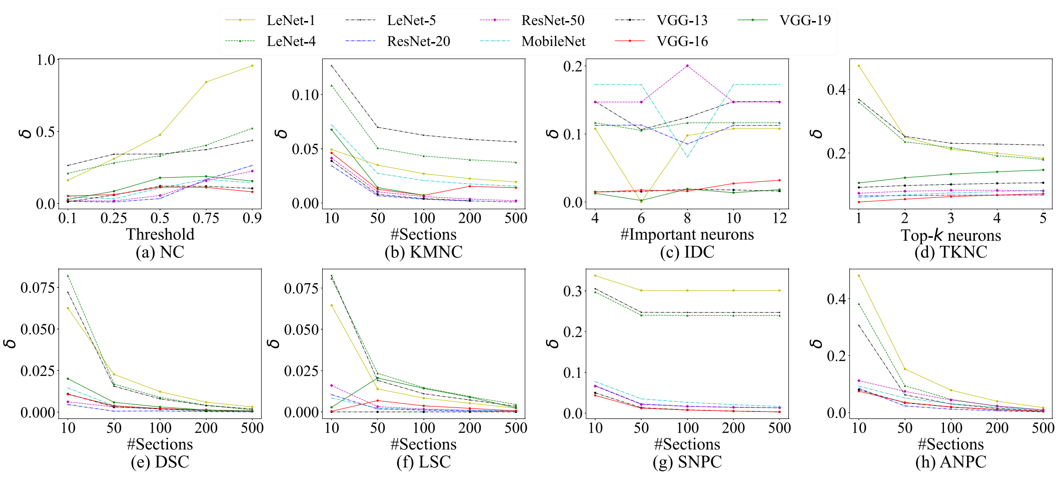

Fig. 2. Test criteria with different hyper-parameters. For KMNC, DSC, and LSC with #sections larger than 500, the computations of test coverage are out-of-memory for all models except LeNet-1.

useful than other neurons to distinguish tests with different behaviors. However, when we take into consideration more neurons, there are more neuron patterns, which may also be helpful in evaluating the differences between tests with different model behaviors. Compared to other coverage criteria, the change of δ in IDC has no obvious patterns on all 9 models. One possible reason is the randomness inherent in IDC. For example, the INCC is clustered by k-means, which are affected by the clustering initialization.

3.3.2 The Effects of Target DNNs. Besides the hyper-parameters of test criteria, we also investigate the impact of target DNNs on the performance of these criteria.

Results. Figure 3 summaries the δ values of coverage criteria over 9 DNNs. Combining the results from Figures 2 and 3, it can be observed that in terms of NC, KMNC, TKNC, DSC, LSC, SNPC, and ANPCδ for models from the LeNet family (i.e., LeNet-1, LeNet-4, and LeNet-5) are much higher than δ for the rest models. Given the detailed information of target DNNs in Table 1, it can be inferred that the effectiveness of these criteria gradually decreased when we increased the complexity of the target DNNs, (e.g., DNNs with more neurons and layers). By analyzing the test coverage profiles, we find that when the target model contains a large number of neurons, the ratio of the neurons or value sections that are shared by different tests is relatively small. From the view of these criteria, *most tests have totally different coverage profiles over medium- and large-scale models.* For this reason, when the target model is too large, these test criteria may treat tests with similar behaviors as totally different tests, which makes it difficult to measure the input-output diversity of tests through these test criteria. Nevertheless, among these criteria, IDC is the only metric that has no big differences over different target DNNs. A possible explanation is that IDC only focuses on the important neurons in the last layer, which are less affected by the complexity of DNNs than other criteria.

Answer to RQ2: The hyper-parameters have a big influence on the effectiveness of criteria. It might be too fine-grained to distinguish tests with different functionalities due to inappropriate settings (e.g., setting k to 1000 and 10,000 for KMNC). In addition, test criteria are much more effective on tiny classifiers (e.g., LeNet-1) than medium- and large-scale models, and criteria with computational feasibility are desirable.

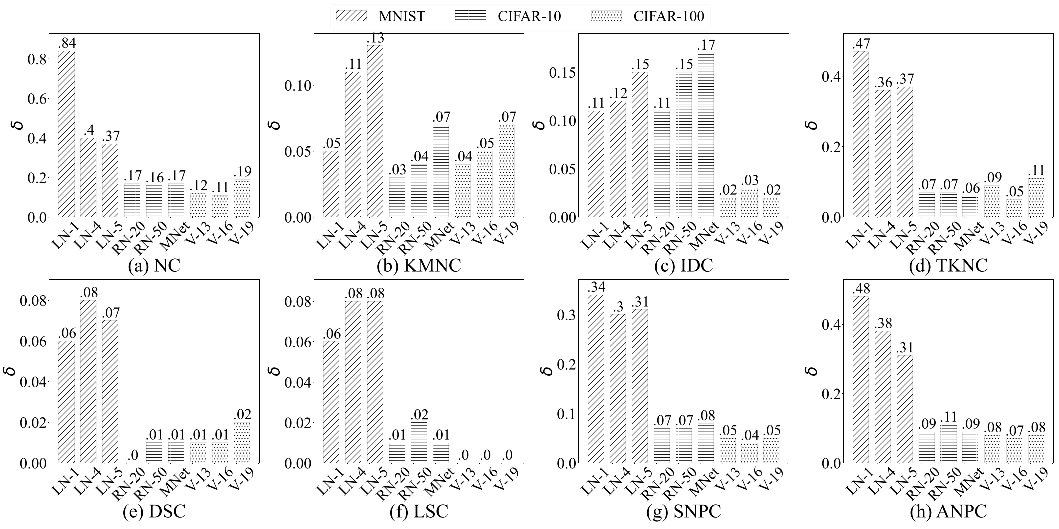

Fig. 3. δ on different target DNNs trained on the LeNet, ResNet, and VGG Families.

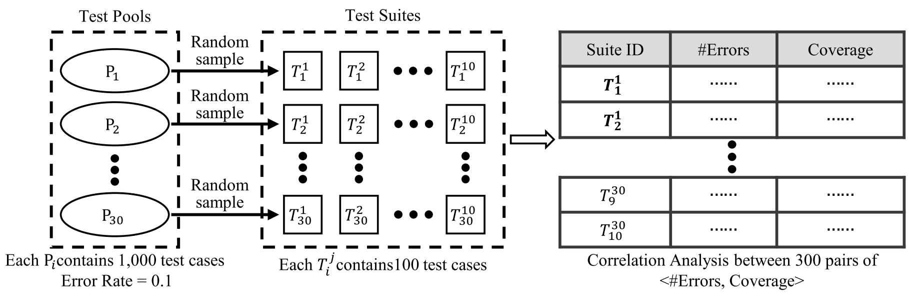
Test Pools
Test Suites
P_1
P_2
P_{30}
Random sample
Random sample
Random sample
T_1^1 T_1^2 T_1^{10}
T_2^1 T_2^2 T_2^{10}
T_{30}^1 T_{30}^2 T_{30}^{10}
Each P_i contains 1,000 test cases
Error Rate = 0.1
Each T_i^j contains 100 test cases
Suite ID
#Errors
Coverage
T_1^1
T_2^1
T_9^{30}
T_{10}^{30}
Correlation Analysis between 300 pairs of <#Errors, Coverage>

Fig. 4. The process of correlation analysis between test coverage and the number of natural faults in test suites.

3.4 RQ3. ID Fault Detection

Discovering defects hidden in DNNs at an early stage can help reduce risks after deployment. To answer RQ3, we aim to investigate the usefulness of existing DNN criteria to detect ID faults. Note that, unlike previous studies on coverage-based adversarial attacks [56, 64, 86, 97], in this article, we focus on *natural faults* that lead target DNNs to produce unexpected predictions.

3.4.1 Correlation between Coverage and the Number of Faults. We first conduct a correlation analysis between test coverage and the fault-detection capability of *same-sized* test suites. We assume that the test set of each dataset is from the same distribution as the training set.

As is show in Figure 4, given a DNN, we first randomly constructed 30 test pools. Each pool is composed of 1,000 natural tests randomly picked from the test sets with the same error rate (i.e., 0.1). We then randomly generated 10 test suites of the same size from each pool, and for each target model, we finally obtained 300 test suites. Finally, we calculated test coverage with regard to each test criterion. To eliminate the effects of test suite size, we fixed it to 100 and computed Kendall's τ rank correlation coefficient [39] between test coverage and the number of ID faults within test suites.

Table 4. Correlation between Test Coverage and the Number of Natural Faults in Test Suites (Size = 100)

DNN Model	NC	KMNC	TKNC	IDC	NBC	SNAC	LSC	DSC	SNPC	ANPC	MCC
LeNet-1	NaN	**0.173**	−0.041	−0.028	−0.021	0.023	**0.321**	**0.360**	**0.305**	**0.151**	
LeNet-4	0.061	0.077	0.042	**−0.101**	0.000	0.006	**0.318**	**0.296**	0.102	**0.140**	0.0237
LeNet-5	0.030	0.016	0.063	−0.039	0.066	0.065	**0.249**	**0.351**	**0.229**	0.103	
ResNet-20	−0.079	−0.011	−0.074	−0.051	0.010	0.015	0.073	**0.096**	0.096	0.030	
ResNet-50	0.071	−0.021	0.020	nan	−0.050	−0.047	**0.122**	**0.125**	-0.143	−0.081	−0.0145
MobileNet	0.057	−0.023	**0.085**	−0.017	−0.058	−0.072	0.032	**0.115**	0.012	0.027	
VGG-13	0.021	0.077	**0.099**	0.043	−0.054	0.035	NaN	**0.142**	−0.038	0.014	
VGG-16	−0.013	−0.014	0.042	−0.039	−0.061	−0.074	0.063	**0.120**	0.022	0.005	0.0175
VGG-19	**0.100**	**0.139**	**0.114**	0.049	0.012	−0.029	0.072	**0.197**	−0.038	0.039	

The bold with shaded values indicate significant and strong correlations (p-value < 0.05 and absolute value of the correlation coefficient greater than or close to 0.1).

*Results***.** Table 4 displays Kendall's correlation coefficients between test coverage and the number of ID faults for 11 metrics and 9 DNN models. The experimental results for MCC are specific to the dataset and are independent of the model structure. To be clear, we mark p-value<0.05 (i.e., significant correlation) with a gray background, and correlation coefficients with an absolute value greater than or close to 0.1 are bolded. Among all the investigated criteria, only DSC is consistently and positively correlated with the number of ID faults within test suites. For LSC, there is a positive correlation between test coverage and the number of ID faults in LeNet-1, LeNet-4, LeNet-5, and ResNet-50. However, the correlation between DSC/LSC and the number of ID faults is not very strong, which is similar to test coverage for traditional software [37]. A possible reason is that with a small test suite, high coverage may not only increase the possibility of discovering new faults but also filter out redundant faults that cannot improve coverage. It can also be seen that with the increasing size of target DNNs, the correlation between DSC/LSC and the number of ID faults decreases. The observations of correlation over different DNNs are consistent with the results in Figure 3 (RQ2). SNPC and ANPC show similar performance to LSC, the neuron path coverage is positively correlated with the natural faults in a partial model of the MNIST dataset. The conclusion that the correlation between coverage and ID faults decreases with the increase of model scale also exists in neuron path coverage.

As for KMNC and TKNC, test coverage are weakly correlated with the number of ID faults in some cases. For NC, NBC, SNAC, IDC and MCC, there is a limited correlation between test coverage and the number of ID faults within test suites. By analyzing the coverage of these criteria, we found that SNAC and NBC are close to 0 in most cases. For instance, in LeNet-1, there are only 3 out of 10,000 natural inputs in the test set whose neuron values are beyond the boundary of the training dataset. In contrast, NC reaches the maximum coverage (i.e., 48.6%) on LeNet-1 with only 6 tests, and then maintains the maximum coverage even when more tests are executed. Similarly, only a small set of tests can reach 100% IDC coverage in most cases, leading the correlation coefficients for IDC to be NaNs or zeros in most cases.

3.4.2 Coverage-Based Test Prioritization. To further investigate the practical use of DNN coverage criteria to find ID faults, we take use of these criteria to guide test prioritization. We adopt the coverage-addition method to test prioritization for DNNs [91]. Specifically, we started with an empty test set. At each iteration, we computed the test coverage of the current set and then selected the test input that improved the existing coverage the most. The process of test input selection stopped when there exist no tests improving test coverage. By this step, we obtained a queue of test inputs from the test pool. Then, we added the remaining tests in the descending order

Table 5. Average Percentage of Fault Detection

Dataset	Model	Test Prioritization Based on Structural Criteria									Baselines	
		NC	KM.	TK.	IDC	NBC	SNAC	LSA	DSA	MCC	SOCP	DG.
MNIST	LN-1	0.10	0.51	0.60	0.60	0.60	0.60	0.88	**0.98**		0.97	0.97
	LN-4	0.32	0.53	0.59	0.59	0.59	0.59	0.81	**0.96**	0.76	0.96	0.96
	LN-5	0.38	0.46	0.59	0.59	0.57	0.56	0.85	**0.99**		0.98	0.98
C-10	RN-20	0.49	0.50	0.50	0.50	0.50	0.50	0.68	**0.86**		0.81	0.81
	RN-50	0.42	0.51	0.51	0.50	0.51	0.51	0.73	**0.89**	0.73	0.84	0.84
	MNet	0.60	0.54	0.51	0.51	0.58	0.49	0.71	**0.89**		0.81	0.80
C-100	V-13	0.49	0.50	0.50	0.50	0.50	0.50	0.53	**0.62**		0.59	0.59
	V-16	0.48	0.50	0.50	0.50	0.50	0.50	0.58	**0.86**	0.78	0.61	0.62
	V-19	0.46	0.50	0.50	0.50	0.50	0.50	0.53	**0.69**		0.62	0.62

The bold with shaded values represent the highest APFD value achieved for each model under different coverage metrics.

of their test coverage. By this means, we prioritize all tests based on structural test coverage. Note that instead of the surprise coverage metrics (i.e., LSC and DSC), we use SA to prioritize tests. The intuition behind is that a test is likely to be faulty if it is "surprise" and novel to the model, which is the definition of SA [40]. Therefore, we prioritize tests with higher SA values for LSC and DSC (i.e., LSA and DSA). To evaluate the performance of these criteria, we compare coverage-oriented test prioritization with two state-of-the-art methods for DNN test prioritization: DeepGini [23] and **softmax output as confidence prediction (SOCP)** [8].

To conduct a fair comparison, we computed the APFD [91] values for nine DNN models shown in Table 1: APFD $= 1 - \frac{\sum_{i=1}^{k} q_i}{kn} + \frac{1}{2n}$, where n represents the number of test inputs, k represents the number of ID faults, and q_i denotes the order of the ith ID fault in the prioritized queue.

Results. Table 5 shows the values of APFD for different methods. The highest APFD value for each model under different coverage criteria is highlighted with bold text and shading. The higher APFD is, the earlier testers can find the ID faults. The experimental results for MCC are specific to the dataset and are independent of the model structure. Overall, it can be seen that the DSA-based test prioritization achieves the highest APFD scores over nine models among all methods. DeepGini and SOCP achieve competitive performance with DSA. The results indicate that based on the distance from tests to the training dataset, DSA is more effective in discovering ID defects of DNN models than SOCP and DeepGini. Moreover, the performance of LSA is worse than that of DSA, SOCP, and DeepGini, but better than the rest criteria. It implies that although LSA is less effective than DSA in finding ID faults, tests that are less likely to appear in the training dataset are more likely to be faulty.

However, it can also be seen that structural coverage criteria (i.e., NC, KMNC, TKNC, SNAC, NBC, and IDC) and black-box coverage criteria (i.e., MCC) are less useful than the SA (i.e., DSA and LSA) in prioritizing ID faults. Specifically, it can be seen that the performance of NC is the worst among all criteria. The reason is that only a small set of tests (less than 1%) can reach the maximum coverage of NC. In addition, the APFD values over ResNet-20, ResNet-50, and MobileNet are much lower than those over the LeNet family. All methods are less effective on CIFAR-100 than MNIST and CIFAR-10. The observation is consistent with the results in RQ1 and RQ2, which indicate that the complexity and structure of target DNNs might affect the effectiveness of existing structural criteria in prioritizing tests.

Answer to RQ3: Among all the investigated criteria, only DSC is consistently and positively correlated with the number of ID faults in test suites. DSA is an effective indicator to guide test prioritization and can find ID faults earlier than the state-of-the-art methods. Nevertheless, the effectiveness of DSC/DSA decreases when faced with more complex DNNs and datasets.

3.5 RQ4. OOD Detection

Considering the real deployment scenario, in this article, we also evaluate the relations between DNN test criteria and OOD data. The reason behind that it is difficult to ensure that all test inputs are ID data, and OOD data is also a key factor that leads to erroneous behaviors of DNN-based software. On the other hand, DNN test criteria were proposed to execute new functionalities of DNNs by covering new neurons, value regions, neuron paths, and so on. Hence, it is natural to measure whether these test criteria can help identify new inputs that have never been seen or even far away from the expected behaviors. To verify this hypothesis, we *first* compare the test coverage of OOD and ID datasets, so as to study whether OOD samples can improve test coverage compared with ID samples. *Then*, to investigate whether OOD samples can trigger new test units (e.g., new neurons) that have never been covered by the training sets, we also computed the percentages of ID and OOD samples that triggers new test units compared with the training datasets (i.e., the percentage of samples that improve the training coverage. Finally, we apply structural coverage criteria for OOD detection and compare them with several baselines. Moreover, since the Surprise Coverage metrics (i.e., LSC and DSC) are based on different surprise boundaries for different datasets, to be fair, we also compute the SA [40] (i.e., LSA and DSA) that measure the relative novelty (i.e., surprise) of each sample with respect to the training set.

To answer RQ4, we employed three widely-used OOD datasets that have been used in previous studies [5, 15, 31, 32, 52, 68], i.e., FashionMNIST [83], LSUN [92], and SVHN [61]. FashionMNIST is a MNIST-like dataset comprised of 28×28 grayscale article images, which contains 60,000 training samples and 10,000 test samples. LSUN is widely used as the benchmark of scene categorization, containing 899 categories and 130,519 images. SVHN is a real-world dataset of house numbers, consisting of 604,388 training samples and 26,032 test samples. Each sample is a 32×32 image of a house number comprised of the digits 0–9.

3.5.1 ID Coverage vs. OOD Coverage. We first study whether OOD samples produce different DNN coverage from ID samples. We conduct experiments on 6 widely-used datasets shown in Section 3.1). As in previous studies [5, 30, 67, 68, 84], for models trained on MNIST, CIFAR-10, and CIFAR-100, we use FashionMNIST, LSUN, and SVHN as their OOD datasets, respectively. For the ID datasets, since each model achieves competitive performance with high accuracy on the test sets, we assume that most data in the test sets conforms to the I.I.D. assumption with the corresponding training sets. Since the OOD datasets are much larger than the ID datasets, to conduct a fair comparison, we randomly select the same number of samples from the OOD datasets and scale the OOD samples to have the same dimensions as the test sets. Note that the Surprise Coverage metrics (i.e., LSC and DSC) are based on different surprise boundaries for different datasets. To be fair, we compute the SA values (i.e., LSA and DSA) to measure the relative novelty with respect to the training set. For the cases where LSA can not be computed due to the singular matrix of the activation trace, we use "-" to denote the LSA values.

Results. Table 6 shows the test coverage and the percentages of tests that improve the training coverage over nine DNN models. Values where the percentage of tests improving training coverage is close to or equal to 100% are highlighted in bold. The experimental results of MCC remain consistent across different model structures within each dataset, as MCC is solely influenced by the

Table 6. Test Coverage (Percentage of Tests That Improve Training Coverage) for ID and OOD Datasets

Training Set	Model	Test Set	Test Coverage (Tests Improving Training Coverage)							Surprise Adequacy	
			NC	KMNC	TKNC	IDC	NBC	SNAC	MCC	LSA	DSA
MNIST	LN-1	O_1	40.38% (0.04%)	38.27% (0.00%)	71.15% (0.89%)	100.00% (0.00%)	61.54% (**100.00%**)	76.92% (**100.00%**)	45.23% (1.07%)	1.1×10^7	2.5×10^3
		I_1	38.46% (0.00%)	98.27% (0.00%)	76.92% (0.00%)	100.00% (0.00%)	11.54% (0.09%)	7.69% (0.04%)	97.62% (0.37%)	-7.00	0.87
	LN-4	O_1	62.84% (0.00%)	68.18% (0.00%)	66.89% (1.69%)	100.00% (0.00%)	54.39% (**100.00%**)	93.92% (**100.00%**)	45.23% (1.07%)	1.6×10^7	2.3×10^3
		I_1	63.51% (0.00%)	96.28% (0.00%)	74.32% (0.01%)	100.00% (0.00%)	7.43% (0.19%)	8.78% (0.15%)	97.62% (0.37%)	-3.35	0.70
	LN-5	O_1	69.78% (1.25%)	80.56% (0.06%)	60.07% (0.26%)	100.00% (0.00%)	52.05% (**100.00%**)	96.64% (**100.00%**)	45.23% (1.07%)	2.5×10^7	1.6×10^3
		I_1	73.88% (0.00%)	96.68% (0.00%)	70.90% (0.01%)	100.00% (0.00%)	6.16% (0.24%)	10.82% (0.22%)	97.62% (0.37%)	-6.41	0.71
C-10	RN-20	O_2	10.97% (98.60%)	44.87% (25.20%)	11.67% (84.59%)	75.00% (0.00%)	67.41% (**100.00%**)	67.43% (**100.00%**)	37.76% (8.98%)	7.7×10^6	1.1×10^3
		I_2	7.70% (0.06%)	93.48% (0.07%)	55.02% (0.11%)	100.00% (0.00%)	13.00% (1.90%)	14.86% (1.12%)	95.43% (3.49%)	-36.14	1.10
	RN-50	O_2	12.90% (**100.00%**)	35.11% (**100.00%**)	7.92% (**100.00%**)	66.67% (0.00%)	55.27% (**100.00%**)	58.88% (**100.00%**)	37.76% (8.98%)	1.3×10^8	9.3×10^2
		I_2	9.83% (0.18%)	83.63% (0.21%)	45.71% (0.24%)	100.00% (0.00%)	11.14% (3.99%)	11.95% (2.19%)	95.43% (3.49%)	-31.45	1.38
	MNet	O_2	20.49% (99.99%)	61.29% (49.98%)	8.55% (84.81%)	83.33% (0.00%)	6.87% (**100.00%**)	7.88% (**100.00%**)	37.76% (8.98%)	1.9×10^5	5.8×10^2
		I_2	34.53% (1.46%)	96.16% (0.21%)	31.65% (1.98%)	100.00% (0.00%)	12.19% (14.08%)	14.04% (8.97%)	95.43% (3.49%)	-25.35	0.97
C-100	V-13	O_3	27.76% (72.21%)	35.49% (0.75%)	4.09% (99.82%)	70.00% (0.00%)	64.69% (**100.00%**)	45.83% (**100.00%**)	35.98% (12.37%)	-	1.2×10^4
		I_3	49.28% (0.12%)	99.35% (0.01%)	24.68% (0.25%)	100.00% (0.00%)	13.42% (5.93%)	11.72% (3.28%)	98.71% (7.68%)	-	1.48
	V-16	O_3	5.00% (**99.12%**)	19.86% (**0.17%**)	0.64% (**100.00%**)	83.33% (0.00%)	61.16% (**100.00%**)	59.02% (**100.00%**)	35.98% (12.37%)	6.3×10^6	8.2×10^4
		I_3	62.72% (0.33%)	98.23% (0.06%)	59.62% (0.70%)	100.00% (0.00%)	12.93% (14.32%)	11.95% (9.67%)	98.71% (7.68%)	5.0×10^4	1.38
	V-19	O_3	13.07% (**99.94%**)	53.03% (9.51%)	2.30% (99.84%)	80.00% (0.00%)	75.79% (**100.00%**)	70.37% (**100.00%**)	35.98% (12.37%)	1.4×10^8	1.6×10^4
		I_3	51.56% (0.42%)	99.03% (0.11%)	29.16% (0.62%)	80.00% (0.00%)	16.45% (15.36%)	15.58% (11.07%)	98.71% (7.68%)	-32.67	1.39

O_1 (OOD dataset 1): FashionMNIST, O_2 (OOD dataset 2): LSUN, O_3 (OOD dataset 3): SVHN, I_1 (ID dataset 1): MNIST Testset, I_2 (ID dataset 2): CIFAR-10 Testset, I_3 (ID dataset 3): CIFAR-100. Values where the percentage of tests improving training coverage is close to or equal to 100% are highlighted in bold.

data distribution. Generally, it can be seen that the SA values (i.e., LSA and DSA) of OOD data are much higher than those of ID data. Specifically, DSA of OOD data is 10^2 to 10^4 times higher than that of ID data. LSA of OOD data is 10^5 to 10^8 times higher than that of ID data. The results indicate that OOD data are novel and "surprise" to the training dataset, leading them to have high SA.

It can also be observed that with regard to NBC, OOD coverage is higher than ID coverage on eight out of nine DNN models by 44.13% to 59.34%. For SNAC, OOD coverage is higher than ID coverage on eight out of nine models by to 34.11%–85.82%. On the other hand, we can also observe that almost all OOD samples improve the SNAC/NBC coverage of the training sets, while only a few ID samples (0.04%–15.36%) touch corner regions measured by SNAC and NBC. The results suggest that OOD data are more likely to trigger corner regions than ID data, and thus SNAC/NBC might be indicative of OOD samples. For NC, KMNC, TKNC, and IDC, there are no significant differences between test coverage of OOD and ID data on small-sized models from the LeNet family. When we increase the complexity of models, it can be seen that the test coverage on OOD data

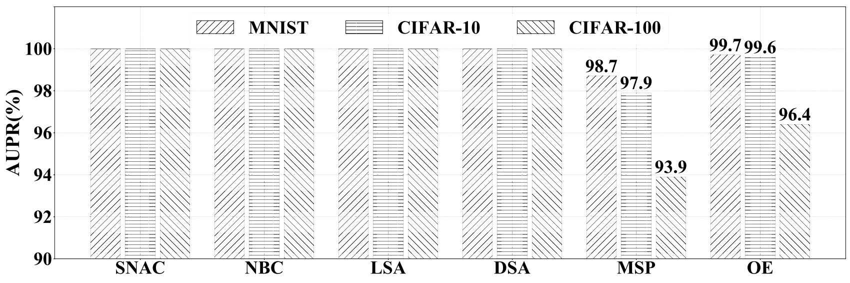

Fig. 5. OOD detection accuracy.

is much lower than the test coverage on ID data. MCC, the black-box coverage criteria, exhibits results comparable to the NC criteria. It is consistent with the fact that the distributions of OOD and ID datasets are different, so they might have different ranges of neuron values and patterns of active neurons.

Moreover, it can also be seen from Table 6 that with regard to all test criteria except IDC, the number of OOD samples that improve test coverage is much larger than that of ID samples over the models trained on models larger than the LeNet family. However, over the models trained on MNIST, we observe no significant differences between the percentages of inputs that improve the test coverage with regard to NC, KMNC, and TKNC. By analyzing the coverage profiles, a possible explanation is that in small-sized DNNs trained on MNIST, both the OOD and ID datasets can easily reach the maximum model coverage with regard to NC, KMNC, and TKNC. Therefore, it is difficult for OOD data to improve the NC, KMNC, and TKNC coverage on small-sized models. Similarly, it can be seen that neither of the OOD and ID data can improve the IDC coverage, since the IDC coverage of the training dataset is close to 100% in all the models.

3.5.2 Coverage-Based OOD Detection. To further Investigate the practical use of DNN coverage criteria to detect OOD samples, we implement the coverage-based OOD detection methods and conduct a comparison study with two baselines: **Maximum Softmax Probability (MSP)** and **Outlier Exposure (OE)** [32]. Note that not all coverage criteria can be converted to the OOD scores. For KMNC, TKNC, and IDC, since every single input has the same coverage, these criteria are not suitable for OOD detection. Instead, we use the coverage of each test with regard to SNAC and NBC as the coverage-based OOD score. We also normalized the SA (i.e., LSA and DSA) values as the SA-based OOD scores. The ID and OOD datasets in this experiment are the same as that in Section 3.5.1. For each experiment, we randomly selected a test set comprised of 10,000 ID samples and 10,000 OOD samples. To avoid randomness, we repeated each experiment 30 times and averaged the results. As in previous studies [15, 42, 44, 52, 65], we use the widely-used metric **Area under the precision-recall curve (AUPR)** to evaluate the performance of the coverage-based methods and the baselines. AUPR is a threshold-unaware metric that summarizes the performance of a detection method, a higher AUPR value means that the confidence value is credible. The precision-recall curve is a graph plotting the $precision = TP/(TP + FP)$ against $recall = TP/(TP + FN)$ for confidence by varying a threshold. In our experiment, we use the coverage of the test case as the confidence.

Results. Figure 5 illustrates the results of OOD detection. From Figure 5, it can be seen that compared with the baselines, SNAC, NBC, LSA, and DSA are more accurate in detecting OOD data. Specifically, AUPR is close to 100% for all the three datasets when SNAC, NBC, LSA, and DSA are utilized to detect OOD data. Combining the results from Table 6 and Figure 5, it can be inferred that boundary coverage and SA are more indicative of OOD samples than other structural coverage criteria (i.e., NC, KMNC, TKNC, and IDC). The reason behind this could be that SNAC, NBC, LSA, and DSA are more related to the shift of the distribution between the training and testing datasets. For instance, NBC calculates the boundary values of the training dataset and measures the ratio of covered corner-case regions which are either higher than the upper boundary or lower than the lower boundary. Similarly, DSA calculates the distance between the activation traces of a new input and the training set. LSA estimates the relative surprise of a new test with regard to the training set with KDE. Since OOD samples are selected from a different distribution from the training set, they might be more likely to trigger the corner regions beyond the training set or have activation traces far from those of the training set. We do not insist on using test criteria for OOD detection. However, the effectiveness of some criteria on OOD detection does reveal their characteristics as indicators of test adequacy and the capability to select natural inputs.

Answer to RQ4: The boundary coverage metrics (i.e., SNAC and NBC) and the surprise adequacy metrics (i.e., DSA and LSA) are more effective indicators of OOD samples than neuron-based criteria (i.e., NC, TKNC, IDC, and KMNC) and outperform OOD detection baselines (i.e., MSP and OE). Their characteristics related to the shift of distribution may contribute to their effectiveness for OOD detection, which sheds light on proposing better coverage criteria that are suitable for DL systems.

4 Discussion

4.1 Computational Cost

Despite some correlation with test diversity and ID/OOD faults, the computational cost of the most investigated criteria may not be feasible. For instance, given a test, DSC requires traversing the whole training dataset, so as to find a pair of the nearest training samples in the same/different categories with the test. Moreover, LSA computes the Gaussian distance between a test and the training set for every layer in a target DNN model, which also needs high computational resources. Neuron path coverage is a coverage criterion with high computational costs, achieved by extracting abstract paths to represent the decision logic of neural networks. It is influenced by several hyperparameters, including path width α, the number of clusters $|\hat{p}|$, and the number of sections k. For complex neural network structures, determining the optimal hyperparameters incurs significant computational expense. In addition to time cost, neuron-based coverage criteria, such as NC and KMNC also require large memory for complex DNNs with a large number of neurons. For instance, KMNC divides the values of each neuron into a number of segments, and measures the coverage of these segments. Therefore, the time and memory cost of KMNC depends on the number of neurons of the target model, as well as the number of segments.

4.2 Future Research Directions

We highlight some useful insights to motivate future research on DL testing. (1) *Structural coverage criteria suitable for DNNs*. Existing criteria are mostly based on the coverage on neuron states. However, neuron-based coverage criteria are not effective enough to test DNN-based software. First, a neuron in a DNN model plays a different role from a statement or a branch in the source code. The activation of a neuron is not simply equal to the execution of a functionality or a feature of DNN-based software. Therefore, high NC does not imply high testing adequacy. Second, neurons are not equally important and independent from each other. For a feed-forward DNN, the values of neurons in each layer depend on the outputs of the preceding layer. Hence, structural coverage criteria that treat all neurons alike may not be suitable for DNNs. (2) *Computationally feasible testing*. A majority of studies on white-box DL testing require high computational resources. For instance, given a new test, the computation time for DSC is linearly correlated with the scale of the training set. The memory for NC and KMNC depends on the number of neurons and value sections in the target model. This article motivates more computationally feasible testing methods for DNN-based software, and inspires conducting experiments on medium- and large-scale DNN models which are commonly used in the real world or their problem domains (e.g., object detection [99]). (3) *Testing from the perspective of generalization.* Most of existing coverage criteria are proposed to generate adversarial samples and expose erroneous behaviors [64, 86, 97]. The experimental results in this article motivate testing and improving the generalization ability on ID and OOD natural samples, which is significant for the deployment of DNN-based software. For instance, boundary testing can be used to detect OOD samples that are likely to make the target model produce wrong predictions. (4) *Explainable testing.* Unlike traditional software testing where failures can directly help localize and fix bugs, the usefulness of failures to debug DNN-based software is limited except for retraining. Due to the lack of explainability, developers have little knowledge about why a DNN works or fails, leading tests less interesting to DNN-based software. This article inspires explainable testing which is more informative and useful to measure and improve the quality of DNNs. For example, compared with neuron-based coverage (e.g., NC and KMNC), SA measures the likelihood of a test or the distance to the training dataset, revealing more semantic information during testing.

5 Threats to Validity

The validity of this study may be subject to some threats. (1) The first threat is the representativeness of image datasets and target image classifiers, which might be not representative and have test bias. To mitigate it, we used six widely-used datasets and nine DNN models that have been the benchmarks for DL testing and OOD detection in previous studies [5, 30, 56, 62, 64, 67, 68, 84, 86]. The experiments on different target models show that the the complexity and structure of the target model and dataset have a significant impact on the effectiveness of these criteria. (2) Another threat might be the comprehensiveness of the coverage criteria used in this study, which do not cover all test criteria for DNNs. Specifically, we do not consider test criteria for stateful DL systems [20, 36] or non-coverage test criteria, such as top-k neuron pattern [56] which counts the number of neuron patterns covered by tests. Due to the limited computational cost, we do not consider the use of concolic testing [74] in this article. However, in this article, we study 11 popular structure coverage criteria specialized for general DNNs. We also investigate the impact of the configuration settings of these criteria. The results shed light on more effective and suitable test criteria for DNN-based software.

6 Related Work

In this section, we review the related works on DNN structural coverage criteria, summarize recent progress on DL testing, and introduce the related works on OOD detection.

6.1 Test Criteria Specialized for DNNs

Recently, there have been many studies that propose coverage criteria for neural networks. Pei et al. [64] first designed NC by the ratio of activated neurons. Multi-granularity testing criteria have been proposed in: KMNC [56].

NBC, SNAC, TKNC, and top-k neuron patterns. Inspired by combinatorial testing, Lei et al. [58] designed a set of combination coverage criteria and systematically explored the states of DNNs layer by layer. DeepCover [73] proposed a family of MC/DC coverage criteria that are tailored to DNNs. They combined a symbolic approach with a gradient-based search to generate new test cases. Based on the intuition that a good test input is supposed to be surprising to the training data, Kim et al. [40] proposed SA to measure the "surprise" (i.e., novelty) of a given test to the training data. Gerasimou et al. [25] designed a novel criterion specialized for DL systems, i.e., IDC. They established a functional understanding of the importance of DNN components, so as to measure the semantic diversity of tests [25]. Inspired by the control flow graph of traditional software, Xie et al. [85] proposed an interpretable coverage criteria for DNNs. They first extracted the decision logic of the target DNN, based on which they built a decision graph. Then, they proposed path coverage criteria to measure the extent to which the test cases have exercised the decision logic.

6.2 DL Testing Approaches

Based on NC, DeepXplore [64] designed the first white-box framework for real-world DL systems. Tensorfuzz [62] and DLfuzz [27] proposed coverage-guided fuzzing methods tailored to DNNs. Following this work, DeepHunter [86] implemented various mutation strategies in the fuzzing process. DeepSearch [94] developed a black-box fuzzing technique to find adversarial samples for image classifiers. DeepTest [75] and DeepRoad [97] explored metamorphic testing methods for DNN-based autonomous driving systems. Sun et al. [74] proposed the first concolic testing for DL systems. Inspired by behavioral testing in software engineering, CheckList [66] designed a matrix of general capabilities and test types for NLP models, and provided a tool to identify critical failures. DeepMutation [57] defined a set of mutation operators and proposed a mutation testing framework for DL systems to measure the quality of test data. Similarly, based on the idea that adversarial samples are more sensitive than natural inputs in mutation testing, Wang et al. [80] proposed to detect adversarial samples at runtime by imposing mutations on DNNs. DiffChaser [87] exploited the genetic algorithm to generate tests that lead to disagreements between version variants of a target DNN. Du et al. [20] first conducted the quantitative analysis of stateful DL systems. They designed two trace similarity metrics, as well as five coverage criteria for recurrent neural networks.

To boost the efficiency of DL testing, Li et al. [47] proposed to sample from the probability distribution of the output values in the last hidden layer of a target DNN, so as to estimate DNN performance. For operational calibration, they further proposed a Bayesian approach to gradually correct the confidence of a DNN model [49]. Moreover, to advance white-box testing, some studies proposed neuron-selection strategies to select a set of internal neurons to determine the direction of gradient-based mutation for fuzzing [27, 45]. Zhang et al. [96] prioritized tests according to frequency of neuron activation patterns. Dola et al. [17] exploited OOD detection to corporate the valid input space in test generation. Aghababaeyan et al. [3] proposed two diversity metrics for black-box testing of DNNs and concluded that the proposed metrics are more reliable indicators than DNN coverage criteria to effectively guide the testing. Ma et al. [59] investigated test selection with real data, adversarial data, and data for retraining. For real data, they only conducted a correlation analysis between the metrics and the binary classification result for each input. However, in this article, we investigate the effectiveness of 11 coverage metrics with regard to the indicators of test diversity and the detection capability of both ID and OOD faults, apply existing metrics to applications, such as test prioritization and OOD detection, and also study the impact of hyper-parameters and target DNNs of these criteria.

6.3 Empirical Studies on DNN Test Criteria

Inspired by the progress of DNN test criteria, many researchers conducted empirical studies toward the effectiveness of DNN test criteria. Gannamaneni et al. [24] evaluated NC as a DNN validation technique. They found that relatively high NC is reached early with only a few samples, however, the final coverage might be reached slowly. They also found that the granularity of test criteria in use determines the effectiveness to uncover faults, which is consistent with our observation. It is noteworthy that they pointed out that OOD samples may increase NC coverage. In this article, we conducted a further study on 11 test coverage criteria, and conducted comprehensive experiments on the fault-detection capabilities of these criteria for both ID and OOD samples.

Weiss et al. [82] conducted a large-scale study on test prioritization for DNNs. They compared seven DNN test criteria with five uncertainty metrics on eight datasets. In this article, we reached the same conclusion that DSA is the most effective metric among all DNN test criteria when detecting ID faults. However, as observed from Table 5, DSA achieved slightly better performance than DeepGini in our experiments, while in their article DeepGini achieved the best performance among all methods. The differences come from the different implementation details, i.e., for the case where the dimension of the activation value is equal to 3, we followed the original article that proposed DSA/LSA [40] and computed the average values of activation values for each neuron, while they refactored the code and flattened the activation value of neurons. As a result, their results were more easily influenced by the shadow layers with big output size, while our results were more easily influenced by layers with a large output channel.

Fabrice et al. [28] conducted an empirical study on the relations between NC and test naturalness. Specifically, they found that NC is not strongly and positively correlated with adversarial attack success and output impartiality. In addition, generating adversarial samples by increasing NC is likely to generate unnatural inputs. Unlike this article, we study the effectiveness of 11 DNN test criteria from the perspective of correctness, to measure that whether these criteria can be utilized to uncover natural ID faults and OOD samples, which are also key factors for erroneous behaviors in the real-world deployment. Li et al. [46] claimed that adversarial samples are persuasively distributed, and coverage-oriented adversarial attacks were actually adversary-oriented attacks. Moreover, they also conducted a preliminary study on detecting natural faults using KMNC with k to be 1,000 and 10,000. However, in this article, we first revealed that when k is too large, the effectiveness of KMNC decreases quickly. In addition, we also conducted experiments with other 10 test criteria and found that DSA was effective for prioritizing natural ID faults.

From the perspective of adversarial robustness, Dong et al. [18] evaluated the relations between coverage criteria and DL robustness metrics, and pointed out that existing coverage criteria have a limited role in guiding model robustness improvement. Yan et al. [88] conducted similar work to further prove that the coverage rate is not related to model robustness from the perspective of the influence of coverage criteria on robustness metric and the generation of adversarial examples. Yang et al. [90] conducted a replication study of the work by Yan et al. [88], and further demonstrated that coverage-driven methods are less effective than gradient-based methods. Moreover, they found that the defects generated by coverage-based methods cannot be repaired by performing adversarial training with gradient-based methods. Hence, defensive methods specialized for coverage-driven methods should be proposed. These studies only focused on the correlation between DNN coverage criteria and model robustness, especially adversarial robustness. However, the correctness of DNNs has not been considered when evaluating the effectiveness of existing DNN test criteria.

6.4 OOD Detection

Research on OOD detection can be dated back to 2017. Dan et al. [31] proposed a simple baseline to identify the ID samples, which utilized the maximum softmax probability as the indicator. There has been a large body of work in the field of OOD detection. The solutions range from *post hoc* methods to confidence enhancement methods and OOD data exposure-based methods.

Given a target DNN model, the *post hoc* OOD detection methods focused on deriving OOD scores based on the model outputs. ODIN [50] distinguished ID and OOD samples by taking temperature scaling and input perturbation to amplify the differences between ID and OOD data. Liu et al. [52] proposed to use energy score for OOD detection. Furthermore, Wang et al. [78] proposed JointEnergy, applying a similar idea to multi-label classification scenarios. Recently, React [72] was proposed to detect OOD data by using mismatched BatchNorm [38] statistics estimated on ID and OOD data. While these methods were mainly based on input perturbation, Huang et al. [35] proposed GradNorm, which derived the score function from the gradient space.

Prediction confidence was also widely exploited for OOD detection. Terrence et al. [15] and Wang et al. [81] estimated the decision confidence with an additional learning branch or class. Then, they used the estimated confidence for OOD detection. G-ODIN [34] proposed a specific loss function DeConf-C to retrain the model and detect OOD samples. Lin et al. [51] exploited intermediate classifier outputs within the DNNs. They utilized features from the optimal depth for OOD detection. The third group of OOD detection methods used a set of OOD samples, i.e., OOD data exposure-based methods. Dan et al. [32] and Dhamija et al. [16] built models to learn a high entropic prediction and suppressed the magnitude of OOD features. Similarly, Yu et al. [93] proposed Maximum Classifier Discrepancy, which was guided to enlarge the entropy discrepancy between ID and OOD samples. Another method considered all OOD data in an extra class and trained the model to separate ID and OOD data [9, 60]. Some methods applied distance metrics to detect OOD data. For instance, Lee et al. [44] calculated the minimum Mahalanobis distances of a sample to all class centroids for detection. Chen et al. [11] determined OOD data with the cosine similarity between the test sample features and the class features. When no OOD data is available, Li et al. [43] attempted to synthesize OOD samples by employing the Generative Adversarial Networks [26].

In this article, we investigate the practical use of existing structural criteria on OOD detection. The results show that the boundary test metrics (i.e., SNAC and NBC) and the SA metrics (i.e., LSA and DSA) are more indicative of OOD samples than other criteria, which might be used for OOD detection.

7 Conclusion

This article presents the first comprehensive study on the effectiveness of DNN test criteria towards model correctness. Considering the real-world deployment, we first investigate whether a test suite with high test coverage indicates a high test diversity. Then, we classify the threats to correctness into ID faults due to the generalization problem and OOD samples which are beyond the goal of the model. Our findings are as follows: (1) For measuring the diversity, coverage criteria considering the relationship between different neurons are more effective than coverage criteria that treat each neuron independently. For instance, the neuron-path criteria (i.e., SNPC and ANPC) show high correlation with test diversity, which is promising to measure test diversity for DNNs. (2) The hyper-parameters have a big influence on the effectiveness of criteria, especially those relevant to the granularity of test criteria. Meanwhile, the computational complexity is one of the important issues to be considered when designing DL test coverage criteria, especially for large-scale models. (3) Test criteria related to data distribution (i.e., LSA and DSA, SNAC, and NBC) can be used to prioritize both ID natural faults and OOD inputs. Furthermore, for OOD detection, the boundary metrics (i.e., SNAC and NBC) are also effective indicators with lower computational costs and higher detection efficiency compared with LSA and DSA. These findings motivate follow-up research on scalable test coverage criteria that improve the correctness of DNNs.

References

[1] 2021. CIFAR10 and CIFAR100. Retrieved from https://www.cs.toronto.edu/kriz/cifar.html

[2] 2021. Tiny-ImageNet. Retrieved from http://groups.csail.mit.edu/vision/TinyImages/

[3] Zohreh Aghababaeyan, Manel Abdellatif, Lionel Briand, and Mojtaba Bagherzadeh. 2021. Black-box testing of deep neural networks through test case diversity. arXiv:2112.12591. Retrieved from https://doi.org/10.1109/tse.2023.3243522

[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning* 79, 1 (2010), 151–175.

[5] David Berend, Xiaofei Xie, Lei Ma, Lingjun Zhou, Yang Liu, Chi Xu, and Jianjun Zhao. 2020. Cats are not fish: Deep learning testing calls for out-of-distribution awareness. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 1041–1052.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to end learning for self-driving cars. arXiv:1604.07316. Retrieved from https://doi.org/10.1109/ivs.2017.7995975

[7] Taejoon Byun, Sanjai Rayadurgam, and Mats P. E. Heimdahl. 2021. Black-box testing of deep neural networks. In *Proceedings of the IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE '21)*. IEEE, 309–320.

[8] Taejoon Byun, Vaibhav Sharma, Abhishek Vijayakumar, Sanjai Rayadurgam, and Darren Cofer. 2019. Input prioritization for testing neural networks. In *Proceedings of the IEEE International Conference on Artificial Intelligence Testing (AITest '19)*. IEEE, 63–70.

[9] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. 2021. ATOM: Robustifying out-of-distribution detection using outlier mining. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 430–445.

[10] Tsong Yueh Chen, Fei-Ching Kuo, Robert G. Merkel, and T. H. Tse. 2010. Adaptive random testing: The art of test case diversity. *Journal of Systems and Software* 83, 1 (2010), 60–66.

[11] Xingyu Chen, Xuguang Lan, Fuchun Sun, and Nanning Zheng. 2020. A boundary based out-of-distribution classifier for generalized zero-shot learning. In *Proceedings of the European Conference on Computer Vision*. Springer, 572–588.

[12] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 4774–4778.

[13] Per-Erik Danielsson. 1980. Euclidean distance mapping. *Computer Graphics and Image Processing* 14, 3 (1980), 227–248.

[14] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[15] Terrance DeVries and Graham W. Taylor. 2018. Learning confidence for out-of-distribution detection in neural networks. arXiv:1802.04865. Retrieved from https://arxiv.org/abs/1802.04865

[16] Akshay Raj Dhamija, Manuel Günther, and Terrance E. Boult. 2018. Reducing network agnostophobia. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 9175–9186.

[17] Swaroopa Dola, Matthew B. Dwyer, and Mary Lou Soffa. 2021. Distribution-aware testing of neural networks using generative models. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE '21)*. IEEE, 226–237.

[18] Yizhen Dong, Peixin Zhang, Jingyi Wang, Shuang Liu, Jun Sun, Jianye Hao, Xinyu Wang, Li Wang, Jin Song Dong, and Dai Ting. 2019. There is limited correlation between coverage and robustness for deep neural networks. arXiv:1911.05904. Retrieved from https://arxiv.org/abs/1911.05904

[19] Nick Drummond and Rob Shearer. 2006. The open world assumption. In *Proceedings of the eSI Workshop: The Closed World of Databases Meets the Open World of the Semantic Web*, Vol. 15. 1.

[20] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. DeepStellar: Model-based quantitative analysis of stateful deep learning systems. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 477–487.

[21] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Jianjun Zhao, and Yang Liu. 2018. DeepCruiser: Automated guided testing for stateful deep learning systems. arXiv:1812.05339. Retrieved from https://arxiv.org/abs/1812.05339

[22] Geli Fei and Bing Liu. 2016. Breaking the closed world assumption in text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 506–514.

[23] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. 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*. 177–188.

[24] Sujan Sai Gannamaneni, Maram Akila, Christian Heinzemann, and Matthias Woehrle. 2022. The good and the bad: Using neuron coverage as a DNN validation technique. In T. Fingscheidt, H. Gottschalk, & S. Houben (Eds.), *Deep Neural Networks and Data for Automated Driving: Robustness, Uncertainty Quantification, and Insights Towards Safety*. Springer International Publishing, Cham, 383–403.

[25] Simos Gerasimou, Hasan Ferit Eniser, and Alper Sen. 2020. Importance-driven deep learning system testing. In *Proceedings of the 42nd International Conference on Software Engineering*. 702–713.

[26] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM* 63, 11 (2020), 139–144.

[27] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. 2018. DLFuzz: Differential fuzzing testing of deep learning systems. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 739–743.

[28] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. 2020. 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*. 851–862.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[30] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. 2019. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 41–50.

[31] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of the International Conference on Learning Representations*.

[32] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. 2019. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*.

[33] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861. Retrieved from https://arxiv.org/abs/1704.04861

[34] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. 2020. Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10951–10960.

[35] Rui Huang, Andrew Geng, and Yixuan Li. 2021a. On the importance of gradients for detecting distributional shifts in the wild. *Proceedings of the Advances in Neural Information Processing Systems*, 2021, 34: 677–689..

[36] Wei Huang, Youcheng Sun, Xingyu Zhao, James Sharp, Wenjie Ruan, Jie Meng, and Xiaowei Huang. 2021b. Coverage-guided testing for recurrent neural networks. *IEEE Transactions on Reliability* 71, 3 (2021), 1191–1206.

[37] Laura Inozemtseva and Reid Holmes. 2014. Coverage is not strongly correlated with test suite effectiveness. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 435–445.

[38] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*. PMLR, 448–456.

[39] Maurice G. Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1–2 (1938), 81–93.

[40] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering (ICSE '19)*. IEEE, 1039–1049.

[41] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.

[42] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2017. Training confidence-calibrated classifiers for detecting out-of-distribution samples. arXiv:1711.09325. Retrieved from https://arxiv.org/abs/1711.09325

[43] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2018a. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *Proceedings of the International Conference on Learning Representations*.

[44] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018b. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Proceedings of the Advances in Neural Information Processing Systems,* Vol. 31.

[45] Seokhyun Lee, Sooyoung Cha, Dain Lee, and Hakjoo Oh. 2020. Effective white-box testing of deep neural networks with adaptive neuron-selection strategy. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 165–176.

[46] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. 2019a. Structural coverage criteria for neural networks could be misleading. In *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results*. 89–92.

[47] Zenan Li, Xiaoxing Ma, Chang Xu, Chun Cao, Jingwei Xu, and Jian Lü. 2019b. Boosting operational DNN testing efficiency through conditioning. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 499–509.

[48] Zenan Li, Xiaoxing Ma, Chang Xu, Jingwei Xu, Chun Cao, and Jian Lü. 2019c. Operational calibration: Debugging confidence errors for DNNs in the field. DOI: https://doi.org/10.1145/3368089.3409696

[49] Zenan Li, Xiaoxing Ma, Chang Xu, Jingwei Xu, Chun Cao, and Jian Lü. 2020. Operational calibration: Debugging confidence errors for DNNs in the field. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020: 901–913.

[50] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2018. Enhancing the reliability of out-of-distribution image detection in neural networks. *Proceedings of the Internation Conference on Learning Representations*.

[51] Ziqian Lin, Sreya Dutta Roy, and Yixuan Li. 2021. MOOD: Multi-level out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 15313–15323.
[52] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020a. Energy-based out-of-distribution detection. *Proceedings of the Advances in Neural Information Processing Systems*, 2020, 33: 21464–21475.
[53] Yuhang Liu, Fandong Zhang, Qianyi Zhang, Siwen Wang, Yizhou Wang, and Yizhou Yu. 2020b. Cross-view correspondence reasoning based on bipartite graph convolutional network for mammogram mass detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 3812–3822.
[54] Wenjie Luo, Bin Yang, and Raquel Urtasun. 2018. Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition.* 3569–3577.
[55] Lei Ma, Felix Juefei-Xu, Minhui Xue, Bo Li, Li Li, Yang Liu, and Jianjun Zhao. 2019. DeepCT: Tomographic combinatorial testing for deep learning systems. In *Proceedings of the IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER '19).* IEEE, 614–618.
[56] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018a. DeepGauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131.
[57] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018b. DeepMutation: Mutation testing of deep learning systems. In *Proceedings of the 29th International Symposium on Software Reliability Engineering (ISSRE).* 100–111.
[58] Lei Ma, Fuyuan Zhang, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018c. Combinatorial testing for deep learning systems. arXiv:1806.07723.
[59] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. 2021. Test selection for deep learning systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 2 (2021), 1–22.
[60] Sina Mohseni, Mandar Pitale, J. B. S. Yadawa, and Zhangyang Wang. 2020. Self-supervised learning for generalizable out-of-distribution detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5216–5223.
[61] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 7.
[62] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. 2019. TensorFuzz: Debugging neural networks with coverage-guided fuzzing. In *Proceedings of the International Conference on Machine Learning*. 4901–4911.
[63] Karl Pearson. 1920. Notes on the history of correlation. *Biometrika* 13, 1 (1920), 25–45.
[64] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.
[65] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. 2019. Likelihood ratios for out-of-distribution detection. arXiv:1906.02845. Retrieved from https://arxiv.org/abs/1906.02845
[66] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* 4902–4912.
[67] Chandramouli Shama Sastry and Sageev Oore. 2020. Detecting out-of-distribution examples with gram matrices. In *Proceedings of the International Conference on Machine Learning.* PMLR, 8491–8501.
[68] Robin Tibor Schirrmeister, Yuxuan Zhou, Tonio Ball, and Dan Zhang. 2020. Understanding anomaly detection with deep invertible networks through hierarchies of distributions and features. arXiv:2006.10848. Retrieved from https://arxiv.org/abs/2006.10848
[69] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484.
[70] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556. Retrieved from https://arxiv.org/abs/1409.1556
[71] Charles Spearman. 1961. "General intelligence" objectively determined and measured. In *Studies in Individual Differences: The Search for Intelligence.* J. J. Jenkins and D. G. Paterson (Eds.). Appleton-Century-Crofts, 59–73.
[72] Yiyou Sun, Chuan Guo, and Yixuan Li. 2021. React: Out-of-distribution detection with rectified activations. In *Proceedings of the Advances in Neural Information Processing Systems*, 2021, 34: 144–157.
[73] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. 2019. Structural test coverage criteria for deep neural networks. *ACM Transactions on Embedded Computing Systems* 18, 5s (2019), 1–23.

[74] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. 2018. Concolic testing for deep neural networks. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 109–119.

[75] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*. 303–314.

[76] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L. Willke. 2018. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 550–564.

[77] Dong Wang, Ziyuan Wang, Chunrong Fang, Yanshan Chen, and Zhenyu Chen. 2019b. DeepPath: Path-driven testing criteria for deep neural networks. In *Proceedings of the IEEE International Conference On Artificial Intelligence Testing (AITest '19)*. IEEE, 119–120.

[78] Haoran Wang, Weitang Liu, Alex Bocchieri, and Yixuan Li. 2021c. Can multi-label classification networks know what they don't know? In *Proceedings of the Advances in Neural Information Processing Systems*, 2021, 34: 29074–29087.

[79] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. 2021a. RobOT: Robustness-oriented testing for deep learning systems. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE '21)*. IEEE, 300–311.

[80] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. 2019a. Adversarial sample detection for deep neural network through model mutation testing. In *Proceedings of the 41st International Conference on Software Engineering*. 1245–1256.

[81] Yezhen Wang, Bo Li, Tong Che, Kaiyang Zhou, Ziwei Liu, and Dongsheng Li. 2021b. energy-based open-world uncertainty modeling for confidence calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9302–9311.

[82] Michael Weiss and Paolo Tonella. 2022. Simple techniques work surprisingly well for neural network test prioritization and active learning (replicability study). In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 139–150.

[83] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747. Retrieved from https://arxiv.org/abs/1708.07747

[84] Zhisheng Xiao, Qing Yan, and Yali Amit. 2020. Likelihood regret: An out-of-distribution detection score for variational auto-encoder. arXiv:2003.02977. Retrieved from https://arxiv.org/abs/2003.02977

[85] Xiaofei Xie, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu, and Yang Liu. 2022. NPC: Neuron Path Coverage via characterizing decision logic of deep neural networks. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 3 (2022), 1–27.

[86] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019a. DeepHunter: A coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 146–157.

[87] Xiaofei Xie, Lei Ma, Haijun Wang, Yuekang Li, Yang Liu, and Xiaohong Li. 2019b. DiffChaser: Detecting disagreements for deep neural networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 5772–5778.

[88] Shenao Yan, Guanhong Tao, Xuwei Liu, Juan Zhai, Shiqing Ma, Lei Xu, and Xiangyu Zhang. 2020. Correlations between deep neural network model coverage criteria and model quality. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 775–787.

[89] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2021. Generalized out-of-distribution detection: A survey. DOI: https://doi.org/10.1007/s11263-024-02117-4

[90] Zhou Yang, Jieke Shi, Muhammad Hilmi Asyrofi, and David Lo. 2022. Revisiting neuron coverage metrics and quality of deep neural networks. In *Proceedings of the IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER '22)*. IEEE, 408–419.

[91] Shin Yoo and Mark Harman. 2012. Regression testing minimization, selection and prioritization: A survey. *Software Testing, Verification and Reliability* 22, 2 (2012), 67–120.

[92] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. 2015. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv:1506.03365. Retrieved from https://arxiv.org/abs/1506.03365

[93] Qing Yu and Kiyoharu Aizawa. 2019. Unsupervised out-of-distribution detection by maximum classifier discrepancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9518–9526.

[94] Fuyuan Zhang, Sankalan Pal Chowdhury, and Maria Christakis. 2020a. DeepSearch: Simple and effective blackbox fuzzing of deep neural networks. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020: 800–812.

[95] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020b. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* 48, 1 (2020), 1–36.

[96] Kai Zhang, Yongtai Zhang, Liwei Zhang, Hongyu Gao, Rongjie Yan, and Jun Yan. 2020c. Neuron activation frequency based test case prioritization. In *Proceedings of the International Symposium on Theoretical Aspects of Software Engineering (TASE '20)*. IEEE, 81–88.

[97] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 2018 33rd IEEE/ACM International Conference on Automated Software Engineering*. 132–142.

[98] Jia-Xing Zhao, Jiang-Jiang Liu, Deng-Ping Fan, Yang Cao, Jufeng Yang, and Ming-Ming Cheng. 2019a. EGNet: Edge guidance network for salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 8779–8788.

[99] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. 2019b. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems* 30, 11 (2019), 3212–3232.

Received 11 October 2022; revised 29 March 2024; accepted 14 May 2024