7-2022

NPC: Neuron path coverage via characterizing decision logic of deep neural networks

Xiaofei XIE
Singapore Management University, xfxie@smu.edu.sg

Tianlin LI

Jian WANG

Lei MA

Qing GUO

See next page for additional authors

NPC: Neuron Path Coverage via Characterizing Decision Logic of Deep Neural Networks

XIAOFEI XIE*, Singapore Management University, Singapore
TIANLIN LI*, Nanyang Technological University, Singapore
JIAN WANG, Nanyang Technological University, Singapore
LEI MA, Kyushu University, Japan
QING GUO†, Nanyang Technological University, Singapore
FELIX JUEFEI-XU, Alibaba Group, USA
YANG LIU†, Nanyang Technological University, Singapore, Zhejiang Sci-Tech University, China

Deep learning has recently been widely applied to many applications across different domains, *e.g.*, image classification and audio recognition. However, the quality of Deep Neural Networks (DNNs) still raises concerns in the practical operational environment, which calls for systematic testing, especially in safety-critical scenarios. Inspired by software testing, a number of structural coverage criteria are designed and proposed to measure the test adequacy of DNNs. However, due to the blackbox nature of DNN, the existing structural coverage criteria are difficult to interpret, making it hard to understand the underlying principles of these criteria. The relationship between the structural coverage and the decision logic of DNNs is unknown. Moreover, recent studies have further revealed the non-existence of correlation between the structural coverage and DNN defect detection, which further posts concerns on what a suitable DNN testing criterion should be.

In this paper, we propose the *interpretable* coverage criteria through constructing the decision structure of a DNN. Mirroring the control flow graph of the traditional program, we first extract a decision graph from a DNN based on its interpretation, where a path of the decision graph represents a decision logic of the DNN. Based on the control flow and data flow of the decision graph, we propose two variants of path coverage to measure the adequacy of the test cases in exercising the decision logic. The higher the path coverage, the more diverse decision logic the DNN is expected to be explored. Our large-scale evaluation results demonstrate that: the path in the decision graph is effective in characterizing the decision of the DNN, and the proposed coverage criteria are also sensitive with errors including natural errors and adversarial examples, and strongly correlated with the output impartiality.

CCS Concepts: • **Software and its engineering → Software testing and debugging**; • **Computing methodologies → Neural networks**.

Additional Key Words and Phrases: Deep Learning Testing, Testing Coverage Criteria, Model Interpretation

1 INTRODUCTION

Deep Learning (DL) has achieved tremendous success and demonstrated its great potential in solving complex tasks in many cutting-edge real-world applications such as image classification [12], speech recognition [44], natural language processing [9] and software engineering tasks (*e.g.*, commit message generation [22], vulnerability detection [19] and clone detection [21]). We have also seen an increasing demand to apply DL in some safety-critical areas, such as autonomous driving [7], healthcare [4] and security applications [29]. However, Deep Neural Networks (DNNs) have been demonstrated with quality issues, *e.g.*, to be vulnerable to adversarial attacks in spite of having

*Equal contribution.
†Corresponding Authors.

Authors' addresses: Xiaofei Xie, xiaofei.xfxie@gmail.com, Singapore Management University, Singapore; Tianlin Li, tianlin001@e.ntu.edu.sg, Nanyang Technological University, Singapore; Jian Wang, jian.wang@ntu.edu.sg, Nanyang Technological University, Singapore; Lei Ma, malei@ait.kyushu-u.ac.jp, Kyushu University, Japan; Qing Guo, qing.guo@ntu.edu.sg, Nanyang Technological University, Singapore; Felix Juefei-Xu, juefei.xu@gmail.com, Alibaba Group, USA; Yang Liu, yangliu@ntu.edu.sg, Nanyang Technological University, Singapore, and Zhejiang Sci-Tech University, China.

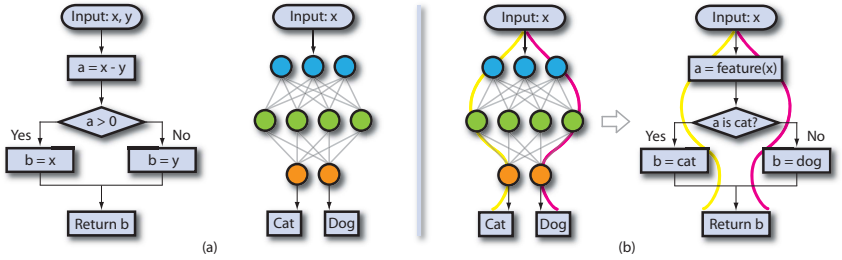

Fig. 1. (a) Difference between traditional software and DL software, (b) Paths in the DL software.

achieved high accuracy [5]. We have already witnessed real-world cases caused by quality issues of DNNs, *e.g.*, Tesla/Uber accidents [47] and incorrect diagnosis in healthcare [4]. The systematic assessment of the quality of DNNs is urgently needed, especially in safety- and security-critical scenarios.

Even until recently, the common way of quality evaluation of DNNs still mostly relies on train-validation-test splits to calculate the accuracy, which may over-estimate the performance of the model [41]. This motivates the software engineering community to propose a variety of novel approaches and tools for systematically testing the quality of the DL software (*i.e.*, the DNN). However, due to the fundamental difference between the traditional code-based software and the DL software (see Fig. 1(a)), the software testing techniques (*e.g.*, code coverage) could not be directly applied in DL software, making testing of the DL software be a new challenge.

To this end, inspired by the white-box testing of the traditional software, a number of approaches towards testing of DL systems have been recently proposed [28]. Specifically, some structural coverage criteria are proposed to measure the test adequacy of the test suite via analyzing the internal neuron activities, *e.g.*, Neuron Coverage (NC) [39], its variants (*e.g.*, KMNC, NBC, SNAC) [31], the combinatorial testing based criteria [30, 42], the Surprise Adequacy [23] and etc. Based on the coverage criteria, automated testing techniques (*e.g.*, DeepXplore [39], DeepTest [46] and DeepHunter [51]) are further developed to generate test cases for maximizing the coverage.

DNN testing aims to cover diverse internal decision logics of DNN such that more bugs (*i.e.*, incorrect decisions) could be detected [42, 54]. Due to the black-box nature and complexity of DNNs, it is unknown and hard to understand the underlying principles of existing coverage criteria [28]. Specifically, for traditional software, the defects reside on particular instances of the program's control and data flow [33]. Thus, the code-based coverage criteria (*e.g.*, statement coverage, branch coverage) are meaningful and essential as covering more statements or branches is more likely to capture diverse behavior (*i.e.*, program logic) in terms of the control and data flow. However, unlike the traditional software, the decision logic of DNN is opaque to human, making it hard to understand the relationship between the decision logic and existing coverage criteria. For example, it is not clear whether (de-)activating more neurons (*i.e.*, NC) is better to capture more diverse behaviors of the DNN. Thus, one may wonder what is the semantics and meaning behind existing coverage criteria.

Some recent research [10, 20, 28, 52] started to perform a more in-depth investigation of existing coverage criteria and have demonstrated their limitations in different perspectives. Harel-Canada et al. have shown that NC is neither positively nor strongly correlated with defect detection, naturalness and output impartiality [20]. For example, their experimental results have shown that only a few inputs could already achieve 100% NC [42]. Li et al. demonstrated that the existing testing criteria are not effective or sensitive in distinguishing same-size test sets of *natural* examples

including different levels of misclassification rates [28]. Dong et al. further showed that there is a limited correlation between the coverage criteria and the robustness of the DNN [10]. These results force us to rethink what could be a better coverage criterion of DNN and how could we design more interpretable coverage criteria.

Motivated by the aforementioned challenges and limitations, in this paper, we propose coverage criteria from the new angle, called Neuron Path Coverage (NPC). We argue that a better coverage criterion should be more relevant to the decision logic of the DNN. To this end, NPC is designed to be *interpretable*[1] (*i.e.*, easily understandable) and can better characterize the decision structure of the DNN. In traditional software, a path in the control flow graph could represent the program logic of the software (*e.g.*, login, logout, invalid user input checking and etc.). Inspired by this, we extract *Critical Decision Paths* (CDPs) from the DNN, which represent the internal decision of the DNN on given inputs. For example, in Fig. 1(b), the DNN makes the decision along the corresponding CDPs (*e.g.*, the yellow path or the red path), which is to classify the input as a dog or a cat. Based on the interpretation technique [3], we extract *critical* neurons in each layer, which dominate the decision of the DNN on the target input. The critical neurons between the layers form the CDP. Compared with the existing neuron-based coverage metric (*e.g.*, NC [39], k-multisection Neuron Coverage [31], IDC [15]), CDP considers not only the critical neurons in one layer but also the relationships among layers. Moreover, the CDP is clearly related to decision-making.

We then recover the learned decision logic of the DNN by extracting CDPs from all training samples. Intuitively, the CDP can be regarded as the control flow of the DNN that shows the important decision logic. Hence, the objective of DL testing is to generate diverse test inputs that can trigger different decision logic (*i.e.*, trigger different control/data flow of the DNN). We firstly construct a Decision Graph (DG) to represent the overall decision logics based on the training data. Since the number of training samples may be very large, we propose the path abstraction technique to build the abstract CDP. The abstract state represents the common decision logic that is shared by some inputs. Based on the DG, we define two variants of path-based coverage criteria, named Structure-based Neuron Path Coverage (SNPC) and Activation-based Neuron Path Coverage (ANPC). Specifically, SNPC is designed based on the control-flow of the DNN (*i.e.*, neurons in the path) while ANPC is designed based on the data-flow of the DNN (*i.e.*, neuron activation values in the path). An input can increase the coverage if it can trigger a different CDP or the similar CDP but with different activation values.

To demonstrate the effectiveness of our technique, we evaluate NPC on 4 widely-used datasets. Specifically, we first evaluate the accuracy of the extracted paths, *i.e.*, whether they are critical and useful in explaining the decision of the DNN. Our results show that the CDPs are critical for the prediction of inputs. After masking the outputs along the CDPs, the prediction results are changed greatly (*e.g.*, 98.9% inconsistency rate on average). Moreover, the CDP is accurate to distinguish decision behaviors between different inputs in the same class. Then, we evaluate the usefulness of the proposed coverage criteria and the results show that NPC could distinguish the same-size test sets including a different number of natural errors. In addition, NPC is positively correlated with the output impartiality [20], *i.e.*, test set with high output bias has lower coverage. We also evaluate the efficiency of the coverage calculation, and the results show that the time overhead is not very expensive although NPC adopts the interpretation analysis.

In summary, this paper makes the following main contributions:

- We propose a method to construct a decision structure of the DNN based on the CDP extraction and abstraction.

[1]We say that NPC is interpretable as it is designed to identify the key neurons that contribute more on the decision of the model based on the interpretation technique, *i.e.*, Layer-Wise Relevance Propagation (LRP) [3]

- Based on the decision structure, we propose two variants of coverage criteria that are more relevant to the decision-making logic of the DNN.
- We conduct the comprehensive evaluation and demonstrate the effectiveness of the proposed approach. Our results show that the CDP is closely relevant to the decision-making; The proposed coverage criteria (*i.e.*, SNPC and ANPC) are positively correlated with the detection of natural errors and output impartiality, respectively.

2 BACKGROUND AND MOTIVATION

In this section, we briefly introduce software testing, the challenges of DL testing, and the interpretation technique LRP.

2.1 Traditional Software Testing

Software testing has been widely studied in the past decades to evaluate the quality of the software. A central question of software testing is "what is a test criterion?", which defines what constitutes an adequate test [56]. In traditional software, the program logic is usually described by the programmer and can be represented as the control flow graph (see Fig. 1(a)). Based on the clear and easy-to-understand logic representation in the program, different coverage criteria have been proposed to measure the test adequacy, *e.g.*, *statement coverage, branch coverage*, and *path coverage*. These criteria are related to the program logic and the underlying principles are also intuitive: more program logic or behaviors could be explored by increasing the coverage (*e.g.*, executing new statements or paths of the program).

2.2 Challenges and Motivation

Deep Neural Networks (DNNs) consist of multiple layers of neurons, which perform the nonlinear calculation for feature extraction and transformation. Similar to traditional software testing, the goal of DL testing is to test the decision logic of the DNN such that the incorrect prediction could be detected [42]. However, unlike the control flow graph of the program, the logic of the DNN is opaque to humans, making it hard to design suitable coverage criteria. Neuron outputs of the DNN could reveal the behavior of the DNN, thus some neuron outputs based coverage criteria [23, 31, 39] are proposed. However, it is not immediately clear how the neuron outputs reflect the internal decision logic, making it hard to understand its underlying principles and the meaningfulness of existing coverage criteria [28].

The value range of neuron outputs is huge and it is often impractical to enumerate all possibilities. For efficiency, existing coverage criteria adopt different strategies to reduce the testing space. For example, NC [39] only considers two states (*i.e.*, activated/inactivated) for the output of each neuron. k-multisection neuron coverage [31] discretizes the output value to k buckets. Surprise metrics [23] select one of the layers to calculate the surprise score. The limitation is that it is unclear about the relationship between the coverage criteria and the decision logic of the DNN (*e.g.*, whether a new decision-making behavior could be triggered by activating one neuron in NC). Consequently, if the coverage criteria are too coarse, some decision logic can be missed. For example, since NC is too coarse, several inputs can achieve very high neuron coverage [20, 42]. If the coverage criteria are too fine-grained, a huge number of inputs are needed. It is because many of them would trigger very similar decision logic with little difference. The further results [10, 20, 28, 42] have also demonstrated the limitation of the existing coverage criteria, which call for new testing criteria design. Overall, this work is motivated by the recent findings [20, 28, 42] on the limitation of the existing coverage criteria: ① the existing coverage criteria lack the interpretability and are unaware of the decision logic [28, 42], ② the existing coverage criteria are not sensitive to the defect

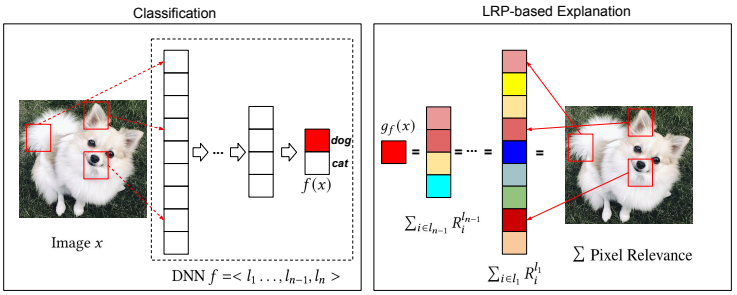

Fig. 2. Visualization of the LRP Process [3]

detection including both of adversarial examples and natural errors [20, 28] and ③ the existing coverage is not positively correlated with the output diversity [20].

Motivation: Similar to the traditional software testing, a better coverage criterion for DL testing should be *interpretable* and more closely *related* to the decision logic of the DNN, which is what the existing coverage criteria lack. When new inputs increase the coverage, it is expected that new decision logic would be exercised and covered.

2.3 Layer-Wise Relevance Propagation (LRP)

Definition 1. *A Deep Neural Network (DNN) f consists of multiple layers $\langle l_1, \ldots, l_n \rangle$, where l_1 is the input layer, l_n is the output layer, and $l_2, \ldots, l_{n-1}$ are hidden layers. The inputs of each layer are from outputs of the previous layer.*

In this work, we mainly focus on the classifier $f : X \rightarrow Y$, where X is a set of inputs and Y is a set of classes. Given an input $x \in X$, $f(x) = \arg\max_y O_n$, where O_n is a $|Y|$-dimensional vector, which is the output of the output layer l_n. The input is classified as the class y, and we use $g_f(x) = O_n(y)$ to represent the output value (*e.g.*, the logits) for the class y. Due to the complex non-linear calculation in the DNN, it is usually hard to understand the decision.

Layer-Wise Relevance Propagation (LRP) [3] is an effective way to interpret the decision by calculating the key features in the input. Given a d-dimensional input x and the classifier f, x is classified as the class y. To interpret the decision, LRP calculates the relevance (or contribution) of each dimension (*i.e.*, pixel) in the input. Specifically, the classifier f outputs $g_f(x)$ on the class y and LRP calculates the relevance of each input dimension:

$$g_f(x) = \sum_{i=1}^{d} R_i^{l_1}$$

where $R_i^{l_1}$ represents the relevance of the dimension i in the input layer l_1 (*e.g.*, the pixel in an image). The sum of the relevance scores is equal to the output value $g_f(x)$, which means that each dimension can contribute to the final decision of predicting y (*i.e.*, $g_f(x)$). Intuitively, $R_i^{l_1} < 0$ contributes evidence against the presence of the class which is to be classified while $R_i^{l_1} > 0$ contributes evidence for its presence [3].

The relevance is calculated by a layer-by-layer backward propagation (from the output layer l_n to the input layer l_1) such that:

$$g_f(x) = R_y^{l_n} = \sum_{i \in l_{n-1}} R_i^{l_{n-1}} = \ldots = \sum_{i \in l_1} R_i^{l_1}$$

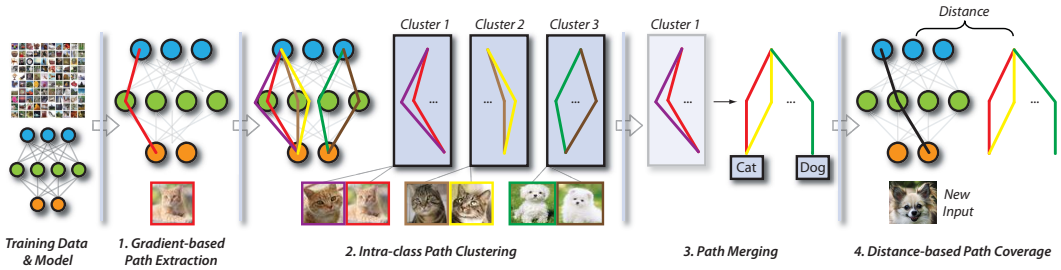

Fig. 3. Overview of this work.

where R_i^l represents the relevance of the neuron i at the layer l. The relevance calculation starts from the output neuron y at the last layer l_n (*i.e.*, $R_y^{l_n}$) until the relevance $R_i^{l_1}$ of the input layer is calculated. At each layer, the sum of the relevance of all neurons is equal to the output $g_f(x)$. To be more specific, each relevance is calculated based on the following relation:

$$R_j^{(l_{m+1})} = \sum_{i \in (l_m)} R_{i \leftarrow j}^{(l_m, l_{m+1})}, R_i^{(l_m)} = \sum_{j \in (l_{m+1})} R_{i \leftarrow j}^{(l_m, l_{m+1})}$$

where $R_j^{(l_{m+1})}$ represents the relevance of the neuron j at the layer l_{m+1}, $R_{i \leftarrow j}^{(l_m, l_{m+1})}$ represents the relevance of the neuron i at the layer l_m, where the output of the neuron i is the input of the neuron j at the layer l_{m+1}. The relevance calculation starts from the output neuron at the last layer l_n is $R_y^{(l_n)} = g_f(x)$ until the relevance $R_i^{(1)}$ of the first layer (*i.e.*, the input layer) is calculated.

Fig. 2 shows the LRP calculation process on an image. In the left box, we show the "black-box" prediction where the image is classified as a dog and outputs $g_f(x)$. To interpret why it is classified as the dog, LRP calculates the relevance layer-by-layer starting from the output $g_f(x)$ (shown in the right box). Finally, neurons/pixels that have more impact on the prediction are identified. Note that different colors represent the different relevance scores of the corresponding neurons. Here we use the red colored squares to represent the neurons that are more critical for extracting features of dogs.

3 METHODOLOGY

3.1 Overview

To design coverage criteria that are related to the decision logic of DNN, it is necessary to understand the decision-making of the black-box DNN (we say DNN is black-box since it is usually uninterpretable). Fortunately, some recent progress has been made on the interpretation of machine learning models [3, 14, 49], which can be used for the interpretable coverage criterion.

In this work, we applied the *Layer-Wise Relevance Propagation* (LRP) [3] to reveal the internal decision of an input. LRP is an effective approach that interprets the classification decisions by the decomposition of nonlinear classifiers (*e.g.*, DNNs) such that the key features in the input (*e.g.*, pixels in the images) could be identified. In particular, given an input, LRP calculates the *relevance* of neurons in each layer starting from the last layer to the input layer in terms of the prediction result. The relevance represents the effect of neurons in the decision. Based on the relevance, we define the *Critical Decision Path* (CDP) that is a set of critical neurons (*i.e.*, with larger relevance in each layer) for the decision. Consider the left figure in Fig. 1(b) as an example, the yellow path and the red path represent different decision logic for the prediction of dogs and cats, respectively.

Our evaluation results show that similar inputs (*e.g.*, dogs) tend to share a similar CDP (*i.e.*, their decision logic is similar) while different inputs (*e.g.*, dogs and cats) have different CDPs.

Inspired by the path coverage criterion in software testing, we propose the Neuron Path Coverage (NPC). For example, in the control flow graph of Fig. 1(a), it returns the maximum value of the inputs x and y. The two paths show the clear logic of the program: the left path represents that x is the larger one while the right path represents that y is the larger one. Similarly, the extracted CDPs are used to show the logic of the DNN. For ease of understanding, we show a dummy graph in the right figure of Fig. 1(b), if the input executes the yellow path, it is classified as the cat, otherwise, a dog.

Fig. 3 shows a detailed workflow of our work. The decision logic of the DNN is determined by the training data, given the target DNN, we first individually extract the CDP for each training data (Section 3.2). The extracted CDPs represent the decision logic that is learned from the training data. Given an input, we measure the coverage by calculating the distance between the CDP of a given input and CDPs of training data (*i.e.*, the difference of the decision logic). However, calculating the distance with all training samples is computationally expensive. To improve the efficiency, we perform the intra-class path clustering and abstraction on the CDPs of training samples (Section 3.3). Specifically, the assumption is that the samples with similar decision logic may have similar CDP. Thus, for the inputs predicted as the same class, we group their CDPs into k clusters. It means that inputs with similar decision logic are clustered into one group. For example, similar cats are clustered together (*e.g.*, the yellow cats and brown cats in Fig. 3). For each cluster, we further propose the path abstraction that merges CDPs of inputs (in the cluster) into an abstract CDP. The abstract CDP represents the decision logic for all inputs in this cluster. Finally, the number of paths is greatly reduced. Based on the abstract CDPs, we propose the distance-based coverage criteria (Section 3.4), which measure to what extent a new decision logic is covered by the given test suite. Specifically, we first identify the abstract CDP of the given input and then propose two coverage criteria. From the control flow perspective, we calculate the *similarity* between the new CDP and the abstract CDP of the corresponding cluster. From the data-flow perspective, we take the activation value of neurons in the abstract CDP into consideration and calculate the *distance* (along the abstract CDP) between the input and training samples in the cluster. The similarity and distance reveal how different the decision behavior of the new input is. Similar to [23], we divided the similarity/distance to m buckets and calculate how many buckets are covered by a given test suite. To adopt the coverage criteria in testing DNNs, we need to select the optimal hyperparameters for the CDP extraction and abstraction. We propose a method to select the parameters by evaluating the accuracy of the CDPs on affecting the decision.

3.2 Path Extraction

In this section, we will introduce how to extract the critical decision path for a given input.

As described in Section 2.3, the relevance R_i^l represents the contribution to the final decision. The larger relevance the relevance R_i^l, the more contribution the neuron i at the layer l. Since the DNN prediction is based on the features extracted and transformed layer-by-layer, we use the critical neurons with large relevance in each layer to represent the internal decision of the DNN. Then we define the critical decision path along all layers of the DNN as follows:

Definition 2 (Critical Decision Path). *Given an input x and the DNN f consisting of n layers, we define its α-Critical Decision Path (α-CDP) as a sequence of neurons sets $p = \langle s_1, \ldots, s_{n-1} \rangle$, where s_i is a set of critical neurons that have the largest relevance values at the layer l_i and*

$$(\forall j \in s_i . R_j^{l_i} > 0) \wedge \left(\sum_{j \in s_i} R_j^{l_i} > \alpha \cdot g_f(x)\right)$$

where α is a parameter that controls the number of the selected critical neurons at each layer.

The neuron is *critical* if it provides a positive contribution to the decision (*i.e.*, the positive relevance). We first sort neurons of each layer by the relevance values, and the CDP is composed of the most critical neurons at each layer. As a layer may have multiple critical neurons, we use the parameter α to prioritize the more critical ones. The smaller the α, the fewer critical neurons are selected at each layer. Other neurons that are not selected in the CDP are called *Non-Critical Decision Path (NCDP)*.

We define the width of the CDP as the average ratio of neurons selected at each layer. It is worth mentioning that, for different inputs, the width of the CDP may be different even with the same value for the parameter α. For each input, we select the CDP that has a large effect on the prediction while the NCDP (*i.e.*, other neurons) has little impact. Furthermore, the width of the CDP (*i.e.*, the selected neurons) should be as small as possible such that it can represent different decision logic accurately. Consider an extreme case like that, we could use all neurons of the network as the CDP, which does not lose any information. However, it cannot distinguish the decision logic between different inputs because all inputs have the same CDP (*i.e.*, the DNN). Our evaluation demonstrated that CDPs with a suitable width could better reveal different predictions, *i.e.*, inputs predicted to be the same class share similar CDPs while inputs predicted to be different classes have different CDPs.

3.3 Path Abstraction

The decision logic of the DNN is determined by the training data, thus we recover its main decision logic that is learned from the training data. Given a classifier f and its training data T, we construct the CDP for each sample in T, which are used to estimate the learned logic of the DNN roughly. However, analyzing all CDPs of T may introduce high computational costs especially when the size of T is large. Moreover, some training data may share similar decision behaviors. To reduce the computational cost, we introduce a two-stage method to calculate the abstract CDPs.

Intra-Class Path Clustering. The assumption is that, the inputs that are predicted to be the same class, should have similar CDPs because the DNN has similar decisions on these inputs. Thus, our basic idea is to generate representative CDPs, which could characterize the common decision pattern on these inputs.

In particular, suppose the classifier f performs a classification task with a set of classes Y, we group the training samples on the predicted class $y \in Y$, *i.e.*, $T_y = \{x \in T | f(x) = y\}$. For each class y, although the decision logic on the inputs T_y is similar, they may still have some slight difference. For example, the decision of the yellow cats may be slightly different from the brown cats. To capture more fine-grained decision logic in each class y, we cluster CDPs of the samples in T_y into k groups. We apply the K-Means clustering algorithm in this paper. Specifically, for each sample, we obtain a vector that represents the status of all neurons of the DNN. If a neuron is included in the CDP, its status is marked as 1 in the vector, otherwise, it is marked as 0. Note that all vectors have the same size such that they can be clustered. $C_j^y \subseteq T_y$ denotes the training samples in the j^{th} cluster. The samples in the same cluster have more similar decision-making logic than samples in other clusters.

Path Merging. For CDPs in each cluster, we then propose the path abstraction by merging them into an abstract path. Specifically, for C_j^y in the j^{th} cluster, we calculate an abstract CDP $\hat{p} = \langle \hat{s}_1, \ldots, \hat{s}_{n-1} \rangle$ by merging neurons in each layer: $\forall i \in \{1, \ldots, n-1\} : \hat{s}_i = \bigcup_{x \in C_j^y} s_i^x$. For each

neuron $t \in \hat{s}_i$, we calculate its weight as:

$$w_t = \frac{|\{x | x \in C_j^y \wedge t \in s_i^x\}|}{|C_j^y|}$$

where s_i^x is the set of critical neurons at the layer i in the CDP of the input x.

Intuitively, for a set of inputs, the abstract CDP is the union of their critical neurons at each layer. The weight of each neuron w_t represents the ratio of CDPs that the neuron t belongs to. The larger the weight, the more critical this neuron is (*i.e.*, it is critical for more CDPs of training samples in this cluster). After merging the paths, the abstract CDP could be much larger than the CDP of one input. There are some neurons in the abstract CDP, which may have small weight. Statistically, neurons with smaller weights may be less important for interpreting the decision on the set of inputs. Thus, we further introduce the parameter β to select the critical neurons whose weights are above the threshold and obtain the abstract CDP $\hat{p}_\beta = \langle \hat{s}_{1,\beta}, \ldots, \hat{s}_{n-1,\beta} \rangle$, where

$$\hat{s}_{i,\beta} = \{t | t \in \hat{s}_i \wedge w_t > \beta\}$$

Based on the abstract CDPs, we characterize the decision logic of a DNN as a decision graph, which is similar to the control flow of programs.

Definition 3 (Decision Graph). *The Decision Graph (DG) of a classifier f is composed of a set of abstract CDPs $G = \{(\hat{p}_\beta^{1,1}, \ldots, \hat{p}_\beta^{1,k}), \ldots, (\hat{p}_\beta^{n,1}, \ldots, \hat{p}_\beta^{n,k})\}$, where n is the number of the classes, k is the number of the clusters for each class and $\hat{p}_\beta^{i,j}$ represents the corresponding abstract CDP with regards to the j^{th} cluster of the class i.*

3.4 Neuron Path Coverage

The decision graph represents the main logic of the DNN learned from its training data. Given a new input, it is expected that the input could be predicted correctly via the learned decision logic. DL testing aims to identify failed inputs that are possibly caused by triggering either *unknown* logic or *incorrect* logic. Specifically, *unknown* logic means that the CDP of the input has low similarity with abstract CDPs in the DG, *i.e.*, the DNN may not be able to learn the logic with the training data. *Incorrect* logic means that the input exercises the incorrect CDP, *e.g.*, one dog may have a similar CDP with a cat. We propose the path coverage to measure the adequacy of a given test suite. Intuitively, the more paths covered, the more logic explored.

Specifically, inspired by the traditional software testing, we consider the control flow and data flow of the DNN and propose two variants of path-based coverage criteria: the *Structure-based Neuron Path Coverage (SNPC)* and the *Activation-based Neuron Path Coverage (ANPC)*. *SNPC* mainly considers the control flow, *i.e.*, the path structure while *ANPC* considers the data flow, *i.e.*, the neuron outputs in the path.

Theoretically, like paths in traditional programs, the number of the paths in DNN could be infinite. To mitigate this problem, we follow the strategy [23] by calculating the distance and use bucketing to discretize the space of the distance.

3.4.1 Structure-based Neuron Path Coverage. Given an input x with the CDP p_x, we first identify the clusters based on the predicted result $f(x)$. Suppose $G_{f(x)} = \{\hat{p}_\beta^{f(x),1}, \ldots, \hat{p}_\beta^{f(x),k}\}$ is the set of abstract paths that are extracted from clusters of the class $f(x)$. We measure the similarity between p_x and each of the abstract paths in $G_{f(x)}$, which represents the difference of the decision behavior of the given input x with the existing abstract paths. Then we divide the similarity into multiple buckets and calculate the coverage, *i.e.*, how many buckets are covered.

Given two paths p_x and $\hat{p}$, we calculate the similarity layer by layer. For a layer l, the similarity is:

$$J_{p_x^l,\hat{p}^l} = \frac{s_l^x \cap \hat{s}_l}{s_l^x \cup \hat{s}_l} \tag{1}$$

where s_l^x and $\hat{s}_l$ are the neurons in the l^{th} layer of p_x and $\hat{p}$, respectively.

Then we divide the continuous space of the similarity (*i.e.*, [0,1]) into m equal buckets $B = \{b_1, \ldots, b_m\}$. For an input x and a target abstract path $\hat{p}$, the covered bucket at layer l is denoted as:

$$b^l_{x,\hat{p}} = b_i \text{ if } J_{p_x^l,\hat{p}^l} \in (\frac{i-1}{m}, \frac{i}{m}]$$

Given a test suite X, we define the Structure-based Neuron Path Coverage as follows:

$$SNPC(X) = \frac{|\{b^l_{x,\hat{p}} | \forall x \in X, \forall \hat{p} \in G_{f(x)}, \forall l \in f\}|}{n \cdot k \cdot |\hat{p}| \cdot m} \tag{2}$$

where $\hat{p}$ is the corresponding abstract CDP, $|\hat{p}|$ is the total number of layers, n is the number of the total classes, k is the number of clusters in the class $f(x)$ and m is the number of buckets.

Specifically, for each layer of the abstract path, we divide the neuron similarity $[0, 1]$ (*i.e.*, Jaccard similarity) between two layers into m buckets. Finally, there are a total of $n \cdot k \cdot |\hat{p}| \cdot m$ buckets, where $n \cdot k$ is the total number of abstract CDPs and $|\hat{p}|$ is the number of layers. SNPC calculates how many buckets can be covered by a test suite X. For a test case $x \in X$ and a layer l, we calculate the layer similarity between the CDP of x and the corresponding abstract CDP $\hat{p}$. The similarity falls into a specific bucket $b^l_{x,\hat{p}}$. We identify all buckets covered by the test suite X, *i.e.*, $\{b^l_{x,\hat{p}} | \forall x \in X, \forall \hat{p} \in G_{f(x)}, \forall l \in f\}$, and calculate the coverage.

We take the similarities with all $\hat{p}$ into consideration for a more fine-grained coverage analysis. This is because one test sample could be relevant with multiple abstract paths in the same class. For example, one abstract path may represent dogs with white color while another abstract path may represent dogs with long ears. Thus, comparing the similarity between the CDP of a test input and different abstract CDPs could measure the input on different feature levels (*i.e.*, with different abstract CDPs).

The higher the coverage criteria, the more diverse logic the DNN has been explored. However, *SNPC* only considers the control flow, which might be coarse in some cases. To be more fine-grained, we therefore propose another variant of path coverage that adopts the concrete activation values of neurons in the path (*i.e.*, data flow).

3.4.2 Activation-based Neuron Path Coverage. In SNPC, we adopt the Jaccard similarity to calculate the structural similarity between the CDP of the input x and the abstract CDP of a cluster. To be more fine-grained, we consider the data flow of the CDP, *i.e.*, the activation values of neurons in the CDP and calculate the distance between the values of two CDPs. Specifically, given an input x and a cluster j, we first select the closest training sample x' from the cluster j, which has highest similarity with x:

$$x' = \arg\max_{x' \in C_j^{f(x)}} J_{p_x,p_{x'}}$$

where $J_{p_x,p_{x'}}$ is the Jaccard similarity between CDPs of x and x'. Then, we calculate the distance between the activation values of x and x' along the abstract CDP in the cluster. Intuitively, the larger the distance, the more different decision behaviors are triggered by the new input x, *i.e.*, compared with the samples in cluster j, more behavior is covered by the new input. For the input

x, we use $A(x, \hat{p}^l)$ to denote the activation values of neurons in the l^{th} layer of the abstract path $\hat{p}$. The distance at the l^{th} layer between x and x' is defined as:

$$D^l_{x,x'} = ||A(x, \hat{p}^l) - A(x', \hat{p}^l)||$$

Similar to [23], we introduce an upper bound U and divide the distance $[0, U]$ into m buckets: $B = \{b_1, \ldots, b_m\}$. Given an input x and an abstract path $\hat{p}$ in the corresponding cluster, we define $d^l_{x,\hat{p}}$ as the covered bucket, *i.e.*, the layer distance $D^l_{x,x'}$ falls into a bucket:

$$d^l_{x,\hat{p}} = b_i \text{ if } D^l_{x,x'} \in (U \cdot \frac{i-1}{m}, U \cdot \frac{i}{m}]$$

Given a test suite X, we define the Activation-based Neuron Path Coverage (ANPC) as:

$$ANPC(X) = \frac{|\{d^l_{x,\hat{p}} | \forall x \in X, \forall \hat{p} \in G_{f(x)}, \forall l \in f\}|}{n \cdot k \cdot |\hat{p}| \cdot m} \tag{3}$$

From the definitions, we can see that the key difference between SNPC and ANPC is that SNPC calculates the distance (*i.e.*, $b^l_{x,\hat{p}_j}$) based on the path structure similarity while ANPC calculates the distance (*i.e.*, $d^l_{x,\hat{p}}$) based on the neuron activation distance. With the coverage criteria, we could measure the adequacy of the test suite with regards to the decision behaviors.

3.4.3 Hyperparameter Tuning. In the two coverage criteria, we have three hyperparameters, *i.e.*, α, β and k, which determine the precision of the (abstract) CDPs with regards to representing the decision logic. To adopt the proposed coverage criteria in DL testing, the hyperparameter tuning is required. We propose a strategy to evaluate the precision of the CDPs (under a set of hyperparameters) on representing the decision logic as follows:

(1) For each training data, we extract its CDP and the NCDP. The expectation is that the CDP has large impact on the decision of the data while the NCDP has less impact on the decision. Moreover, the width of the CDP should be as small as possible such that only critical neurons are selected. Specifically, we respectively dropout the outputs of neurons (*i.e.*, mask the results as zero) in the CDP and the NCDP, and evaluate whether the prediction result is changed (*i.e.*, inconsistent). We evaluate the inconsistency rate on all training data on some hyperparameters and select the best ones that satisfies our assumption.
(2) Similarly, we evaluate the precision of abstract CDPs. For the abstract CDP of a cluster, we calculate the inconsistency rate on all training data of the cluster. For each data, we dropout the same neurons that belong to the abstract CDP. The inconsistency rate is expected to be high by masking the neurons of the abstract CDP and low by masking the neurons of the abstract NCDP.

In our paper, the value ranges of the three hyperparameters α, β and k are $\{0.7, 0.8, 0.9, 1.0\}$, $\{0.6, 0.7, 0.8, 0.9\}$ and $\{1, 4, 7\}$, respectively.

4 EVALUATION

4.1 Setup

To evaluate the effectiveness of our approach, we have implemented the coverage criteria based on the PyTorch framework. We design experiments to demonstrate the usefulness of our method in mitigating these challenges. Specifically, we aim to investigate and answer the following research questions:

- **RQ1**: How does the CDP affect the decision of the DNN? The CDP only selects the critical neurons in each layer. We design the experiment to evaluate whether such neurons are critical for DNN decisions.
- **RQ2**: How accurate is the abstract CDP in revealing internal decisions? Since the abstract CDP is extracted from CDPs of training samples in the corresponding cluster, we aim to evaluate whether the abstract CDP is still accurate in representing the DNN decision on these training samples in the cluster.
- **RQ3**: Is NPC sensitive with errors including natural errors and adversarial examples? We investigate whether the proposed coverage criteria are sensitive in distinguishing the decision difference between erroneous samples and benign samples.
- **RQ4**: Is NPC correlated to the output impartiality? A test suite should exercise diverse behaviors and should not prefer only a few output values [28]. We investigate whether the coverage criteria can reveal the impartiality of the test suite, *i.e.*, the test suite with diverse behaviors should have high coverage while the test suite with biased behavior should have low coverage.
- **RQ5**: How efficient is the calculation of NPC? Since NPC extracts decision behaviors based on the interpretation method, we investigate whether the interpretation analysis could introduce much time overhead for the coverage calculation.

4.2 Subject Datasets and DNNs

We selected three widely used datasets in the image classification domain and trained four models with competitive test accuracy, which are widely used in the previous works [23, 31, 39].

- **MNIST** [26] is for handwritten digit image recognition (*i.e.*, handwritten digits from 0 to 9), containing 60,000 training data and 10,000 test data. We select the five layer Convolutional Neural Network [23] (named *SADL-1*) and achieve an accuracy of 99.1%.
- **CIFAR-10** [24] is a collection of images for general-purpose image classification, including 50,000 training data and 10,000 test data in 10 different classes (*e.g.*, airplanes, cars, birds, and cats). We select two models, *i.e.*, the 12-layer Convolutional Neural Network [23] (named *SADL-2*) and VGG16, which achieved the accuracy of 90.36% and 89.17%, respectively.
- **SVHN** [36] dataset is house numbers in Google Street View images, including 73,257 training data and 26,032 test data. We select the model AlexNet that has an accuracy of 94.31%.
- **ImageNet** is a large-scale visual recognition challenge (ILSVRC) dataset. The ImageNet contains a large number of training data (*i.e.*, over one million) and test data (*i.e.*, 50,000) with 1000 classes, each of which is of size $224 \times 224 \times 3$. To reduce the complexity of training and CDP evaluation, we select the first 10 classes to train a classifier with the model VGG16 that performs the classification on the 10 categories. The model is trained with an accuracy of 76.81%.

4.3 Baselines

We select the state-of-the-art coverage criteria as the baselines for the evaluation, *i.e.*, Neuron Coverage (NC) [39], k-Multisection Neuron Coverage (KMNC) [31], Neuron Boundary Coverage (NBC) [31], Likelihood-based Surprise Coverage (LSC) [31], Distance-based Surprise Coverage (DSC) [23] and Importance-Driven Coverage (IDC) [15]. Due to that the original IDC only supports Keras models, we implemented a PyTorch version of IDC for the comparison.

To be comprehensive, for NC, we set the threshold as 0, 0.2, 0.5 and 0.75 following [20]. For KMNC and NBC, we follow the configuration [31, 51] and set the k value as 1,000 and 10. For LSC and DSC, we follow the configuration [23], set the upper bound 2,000 and 2.0 respectively, and set the number of buckets as 1,000. For the layer selection, we select the same layer in SADL-1 and SADL-2, which is used in [23]. For VGG16 in CIFAR and ImageNet, we select the last layer since it

Table 1. The average width (%) and inconsistency rate (%) after masking neurons in the CDP and NCDP

Dataset	Model	$\alpha = 0.7$			$\alpha = 0.8$			$\alpha = 0.9$			$\alpha = 1$		
		Width	Inc.C	Inc.NC	Width	Inc.C	Inc.NC	Width	Inc.C	Inc.NC	Width	Inc.C	Inc.NC
MNIST	SADL-1	9.3	89.3	9.6	**12.8**	**95.8**	**1.9**	17.2	97	0	33.3	98.9	1.8
CIFAR	SADL-2	**20.9**	**100**	**0**	27.2	100	0	36.3	100	0	63.4	100	0
	VGG16	10.2	100	12.1	13.3	100	1.5	**17.9**	**100**	**0.1**	33.8	100	0
SVHN	AlexNet	**23.0**	**100**	**2.1**	29.6	100	1.2	38.2	100	0.2	64.6	100	0
ImageNet	VGG16	**41.1**	**99.3**	**12.5**	51.9	99.8	13.9	65.9	99.9	13.2	98.8	99.8	0.2

achieved the best result. For IDC, we follow the same configuration [15] and select the penultimate layer for each model. In the selected layer, we choose the top 12 important neurons. For SNPC and ANPC, we set the number of buckets as 200 and the upper bound as 2.0.

4.4 RQ1: Relationship between CDP and the DNN Decision

4.4.1 Results of masking neurons of CDPs/NCDPs.

Mask all neurons. To evaluate whether the CDP has a critical impact on the decision of the DNN, we dropout the outputs of neurons in the CDP (*i.e.*, mask them as zero) and check whether the predicted result is inconsistent with the original result. Intuitively, we say that the CDP is accurate to reveal the decision if 1) the predicted output is largely affected after the neurons of the CDP are masked and 2) the predicted output is less affected after neurons in NCDP are masked. We extract CDPs for all training data and calculate the inconsistency rate after masking neurons in their CDPs and the corresponding NCDPs, respectively. Specifically, the inconsistency rate of the classifier is defined as:

$$\frac{\{x|x \in X \wedge f(x) \neq f_m(x)\}}{|X|}$$

where X is a set of test cases, $f(x)$ is the prediction result of the original classifier f on the input x and $f_m(x)$ is the prediction result after masking the CDP or NCDP of x. Intuitively, the higher the inconsistency rate, the larger the impact of the masked path.

Table 1 shows the average inconsistency rate by masking all neurons in the CDPs/NCDPs of all training data. The inconsistency rate shows the percentage of samples, whose prediction results are changed after the masking. To extract the path, we configure the parameter α (see Definition 2) with different values, *i.e.*, 0.7, 0.8, 0.9 and 1. Under each configuration, we show the average width of the CDP (Column *Width*), which is the average ratio of neurons at each layer of the CDP, and the inconsistency rate after masking CDP (Column *Inc.C*) and masking NCDP (Column *Inc.NC*).

The results show that the width of the CDP increases as the parameter α increases. For example, when α is 0.8, the average width on VGG-16 is 13.3%, indicating that 13.3% critical neurons are selected at each layer. When α is 1, the width becomes much larger (*i.e.*, 33.8%). Furthermore, we found that almost all predicted results are changed when the CDPs are masked (*e.g.*, for SADL-2 and VGG16, the inconsistency rate is 100% for all configuration), which demonstrates that the CDP is very *critical* for the decision. Conversely, when we mask other neurons that are not in the CDP (*i.e.*, the NCDP), the inconsistency is very low (0% to 3%). In particular, when α is small, the width is small (*i.e.*, less neurons are selected) and the performance is relatively low. For example, when α is 0.7, the inconsistency rate of masking CDP on SADL-1 is 89.3%, which is lower than 98.9% ($\alpha = 1$); The inconsistency rate of masking NCDP in CIFAR-VGG16 is much higher (12.1%) than other configurations regarding α, which shows that there are some critical neurons (in NCDP) that are missed. Based on the results in Table 1, in the following experiments, we set α as 0.8,

Table 2. The average inconsistency rate (%) after masking parts of neurons in the CDP.

Dataset	Model	Inconsistency Rate				
		20%	20%-40%	40%-60%	60%-80%	80%-100%
MNIST	SADL-1	99.13	97.53	92.46	79.73	49.02
CIFAR	SADL-2	9.13	7.65	3.29	2.39	0.58
	VGG16	97.94	84.13	62.42	43.21	22.64
SVHN	AlexNet	99.96	99.45	78.7	42.86	15.95
ImageNet	VGG16	73.22	27.02	15.07	9.91	6.21

Table 3. The average similarity of CDPs within a class or between different classes

α	SADL-1			SADL-2			VGG16(CIFAR-10)			AlexNet			VGG16(ImageNet)		
	Width	Intra	Inter	Width	Intra	Inter	Width	Intra	Inter	Width	Intra	Inter	Width	Intra	Inter
0.7	9.30%	57.7%	34.7%	20.9%	48.6%	23.2%	10.2%	86.5%	22.2%	23.0%	47.8%	30.0%	41.1%	54.9%	32.2%
0.8	12.80%	62.8%	39.1%	27.2%	56.5%	41.8%	13.3%	60.9%	26.0%	29.6%	54.5%	36.5%	51.9%	60.8%	41.6%
0.9	17.20%	67.6%	45.3%	36.3%	64.4%	38.0%	17.9%	67.1%	32.0%	38.2%	63.2%	46.1%	65.9%	70.9%	55.8%
1	33.30%	74.5%	56.3%	63.4%	86.5%	62.0%	33.8%	78.3%	56.1%	64.6%	78.3%	75.4%	98.8%	99.9%	97.9%

0.7, 0.9, 0.7 and 0.7 for MNIST-SADL-1, CIFAR-SADL-2, CIFAR-VGG16 and SVHN-AlexNet and Imagenet-VGG16, respectively. We also evaluate the CDPs on test data and the results show the similar trends.

Mask parts of neurons of CDPs. To further evaluate the impact of CDP, we conduct a fine-grained evaluation that only masks a small part of neurons based on their relevance. Specifically, for a given CDP, we sort the neurons at each layer of the CDP and then split these sorted neurons into 5 equal buckets that have different relevance values, *i.e.*, the top 20% neurons with high relevance, the 20%-40% neurons, the 40%-80% neurons and the last 20% neurons (*i.e.*, 80%-100%) with low relevance. We get 5 sub-CDPs that contain neurons with different relevance values. Then we mask each of the sub-CDPs and calculate the inconsistency rate. Table 2 shows the inconsistency rate by only masking parts of the CDP. Overall, we could find that the neurons with higher relevance can have a larger impact on the prediction. After we mask the most important part (*i.e.*, the top 20% neurons), the largest inconsistency rate is caused, which indicates that these neurons have larger impact. When the last 20% neurons are masked, the inconsistency rate is the smallest.

We also observe that the results depend on the tasks and models. For example, on CIFAR-VGG16, after masking the top 20% neurons of CDPs, most inputs are inconsistent with the original prediction (97.94%), which shows that these neurons are quite important for the prediction. However, for the model SADL-2, only 9.13% inputs are inconsistent after masking the top 20% neurons of the CDPs. Refer to the results in Table 1, after masking all neurons in the CDP ($\alpha = 0.7$), the inconsistency rate on SADL-2 becomes 100%. It means that all neurons of the CDPs on SADL-2 are important for the prediction. Consider the results of CIFAR-VGG16 and ImageNet-VGG16, although the same model architecture is used, their results are not consistent, indicating that the CDPs are different on different datasets.

4.4.2 Results on effect of the width. The width of the CDP may affect its accuracy in representing the decision logic. If the width is too large, the CDP contains some non-critical neurons that may also be included in other CDP, making it hard to distinguish them. We investigate the similarity between different classes. Specifically, for each class, we randomly select 100 test data that is

correctly predicted as the target class. Then we calculate the average similarity of CDPs in the same class or between different classes as follows:

$$Intra = avg(\{sim(x_i^y, x_j^y) | y \in Y \wedge i \neq j \wedge x_i^y \in X^y \wedge x_j^y \in X^y\})$$

$$Inter = avg(\{sim(x_i^{y_m}, x_j^{y_n}) | y_m \in Y \wedge y_n \in Y \wedge m \neq n \wedge x_i^{y_m} \in X^{y_m} \wedge x_j^{y_n} \in X^{y_n}\})$$

where Y is the set of all classes, X^y is a set of test data (its size is 100) in the class y, $sim(x, y) = \frac{\sum_{l \in f} J_{p_x^l, p_y^l}}{|f|}$ represents the similarity between CDPs of x and y (see Equation 1). Intuitively, the similarity between CDPs of the same class should be greater than the similarity between CDPs of two different classes.

Table 3 shows the average similarity between two CDPs in the same class (*i.e.*, Column *Intra*) and in different classes (*i.e.*, Column *Inter*) on the training data. The results show that the average similarity increases with the increase of the width, which further demonstrates that, if the CDP is too wide, it is not accurate to show the decision difference on two inputs. If the width of the CDP is too small, then it may miss some critical neurons (see Table 1).

Answer to RQ1: CDPs are critical for the prediction while NCDPs are not critical, which reveals that the CDPs are more relevant to the internal decision logic of the DNN.

4.5 RQ2: Effectiveness of Path Abstraction

Setup. Since the abstract path represents the decision logic of training samples in the corresponding cluster, we first investigate whether the abstract path is still critical for the prediction of inputs in the cluster. Similar to RQ1, we mask the abstract CDP/NCDP and see the average inconsistency rate of training inputs in the cluster. Then, we evaluate whether the path abstraction could identify similar decision logic by measuring the similarity with or without clustering. Specifically, for each cluster that contains a set of training samples X, we can calculate its abstract CDP based on all CDPs of X. To evaluate the precision of the path abstraction, for every input $x \in X$, we masked the same neurons (*i.e.*, the abstract CDP) and evaluate the average inconsistency rate. Note that, in RQ1, we mask different neurons for different inputs of X based on their individual CDPs while we mask the same neurons (in the abstract CDP) for all inputs of X in RQ2.

4.5.1 Impact on prediction. For the path abstraction of each DNN, we investigate a total of 16 configurations with different parameters (k, β), where k is the number of the clusters in each class and β is the threshold to select neurons with high weights (the first column in Table 4). Note that, when $k = 1$, the abstract CDP is extracted by merging CDPs of all inputs that are predicted as the same class.

The results in Table 4 show that, with the increase of the number of clusters, the width tends to be slightly larger in most cases under the same threshold β (see Column *Wid.*), indicating that the average weight of neurons is also larger (*i.e.*, CDPs in each cluster share more common neurons). We count the number of the change of Inc.C and found that 81.25% (39/48) of the Inc.C change is increasing or unchanged when the number of clusters increases. More fine-grained clustering (*i.e.*, more clusters) could merge the similar CDPs together, by which the abstract CDP is more *stable* (larger average weights) and *accurate* (higher inconsistency rate after masking CDPs). The results demonstrate the usefulness of clustering and merging.

Compared with the results on single CDPs (Table 1), we found that the abstract path could still achieve competitive performance and even better than single CDPs on some models. It means that the path abstraction can be used to select commonly important neurons for all training data in a specific cluster (*i.e.*, high inconsistency rate after masking CDPs and low inconsistency rate

Table 4. The average width and inconsistency rate (%) after masking neurons in the abstract CDP and NCDP

(k, β)	SADL-1			SADL-2			VGG16(CIFAR-10)			AlexNet			VGG16(ImageNet)		
	Wid.	Inc.C	Inc.NC	Wid.	Inc.C	Inc.NC	Wid.	Inc.C	Inc.NC	Wid.	Inc.C	Inc.NC	Wid.	Inc.C	Inc.NC
(1, 0.6)	16.9	93.6	2.3	13.9	99.9	2.4	15.5	99.8	4.3	17.3	99.4	4.9	41.5	99.8	1.7
(1, 0.7)	14.6	89.2	8.4	9.4	99.8	2.4	13.1	99.3	4.3	13.6	99.1	13.9	34.4	99.8	1.7
(1, 0.8)	12.5	78.8	29.9	6.4	99.2	2.4	10.8	98.8	4.3	9.8	96.3	16.3	26.4	99.8	1.7
(1, 0.9)	10.4	73.6	59.8	4.1	90.9	2.4	7.9	100	10.1	5.7	50.9	37.6	17.7	99.8	1.7
(4, 0.6)	16.7	94.5	2.4	14.6	99.9	2.0	15.8	99.9	4.3	**18.5**	**99.5**	**4.4**	54.5	100	2.9
(4, 0.7)	15.4	95	2.5	10.8	99.9	2.0	13.5	99.9	4.8	15.1	99	5.8	**49**	**100**	**1.9**
(4, 0.8)	**14.6**	**94.1**	**3.9**	7.6	99.6	2.0	11.2	99.9	4.8	11.8	96.4	16.2	40.4	100	2.67
(4, 0.9)	12.2	88.1	14.3	4.8	95.3	2.0	8.4	100	9.2	7.6	80	26.7	34.3	100	2.6
(7, 0.6)	16.9	96.3	2.6	14.9	99.9	1.5	15.9	99.8	2.8	18.4	99.4	5.6	64.2	100	4.3
(7, 0.7)	15.9	96.1	3.2	11.2	99.8	1.5	13.7	99.9	3.5	15.2	98.9	6	59.5	100	2.9
(7, 0.8)	14.6	92.5	2.9	7.9	99.7	1.5	11.4	99.9	4.4	12.1	97.6	9.7	51.8	100	3.6
(7, 0.9)	12.5	88.7	9.5	**5.4**	**98.2**	**1.6**	**8.7**	**100**	**4.9**	8	90.5	19	46.7	100	3.1
(10, 0.6)	16.9	95.3	1.7	15.2	99.9	1.1	15.9	99.9	2.0	18.5	99.4	5.2	73.2	100	4.3
(10, 0.7)	16.1	95.1	3.1	11.5	99.8	1.2	13.7	99.9	2.7	15.4	98.9	6.3	70.5	100	4.3
(10, 0.8)	14.6	93.0	4.3	8.3	99.8	1.3	11.5	99.9	3.4	12.4	98.4	8.5	58.3	100	3.9
(10, 0.9)	13.5	89.2	8.8	5.4	98.8	1.3	8.7	99.9	8.8	8.6	92.9	17.7	54.3	100	2.7

after masking NCDPs). For example, on SADL-2, the inconsistency rates of CDP and NCDP are 98.2% and 1.6% ($k = 7$), which are close to results of single CDPs (*i.e.*, 100% and 0%) and the width is much smaller (*e.g.*, 5.4% for abstract CDPs and 27.2% for single CDPs). For AlexNet, using the abstraction in the cluster, we could select more accurate CDPs (*e.g.*, the average width of the abstract CDPs and the single CDPs is 18.5% and 23.0%). Finally, in the following experiments, we select the configuration (4, 0.8), (7, 0.9), (7, 0.9), (4, 0.6) and (4, 0.7) as the parameters for SADL-1, SADL-2, VGG-16, AlexNet and VGG-16 (ImangeNet), respectively (the bold numbers). These configurations are selected due to that we expect to select the neurons that can be representative and critical. Firstly, we could not leave critical neurons out, thus we need to ensure a great inconsistent rate under the selected configurations. Secondly, the width of critical neurons should not be great because the critical neurons are expected to be representative and discriminative. If the width is too large, there will be a large overlap between different critical neurons.

4.5.2 Decision Revealing. Table 5 shows the results on the average similarity with and without clustering. Column *#Clus.* shows the number of clusters configured for each model. Without clustering, Column *#Intra_Cla.* shows the average similarity between CDPs that are in the same class (*i.e.*, the intra-class similarity). Column *#Inter_Cla.* shows the average similarity between CDPs that belong to different classes (*i.e.*, the inter-class similarity). After the clustering, we show the average similarity between CDPs in the same cluster (*i.e.*, intra-cluster similarity shown in Column *#Intra_Clus.*) and the average similarity between CDPs in different clusters of the *same* class (*i.e.*, inter-cluster similarity shown in Column *#Inter_Clus.*). Note that inter-cluster similarity only compares clusters that belong to the same class

Comparing the results between intra-class similarity (without clustering) and intra-cluster similarity (with clustering), we found that the average similarity increases (*e.g.*, from 0.628 to 0.708 in SADL-1). In particular, in the same class, the inter-cluster similarity is much smaller than intra-cluster similarity although the two clusters belong to the same class (*e.g.*, 0.708 for intra-cluster and 0.381 for inter-cluster). Moreover, the inter-cluster similarity is even close to the inter-class similarity. The results show that samples in the same class could have different decision logic (*e.g.*, the black dogs and white dogs may be different). By the path abstraction, we generate fine-grained CDPs which could distinguish such differences well.

Table 5. The average similarity with and without clustering

Model	#Clus.	Intra_Cla.	Inter_Cla.	Intra_Clus.	Inter_Clus.
SADL-1	4	0.628	0.391	0.708	0.381
SADL-2	7	0.486	0.232	0.524	0.260
VGG16(CIFAR-10)	7	0.671	0.320	0.703	0.316
AlexNet	4	0.478	0.300	0.524	0.298
VGG16(ImageNet)	4	0.549	0.322	0.546	0.331

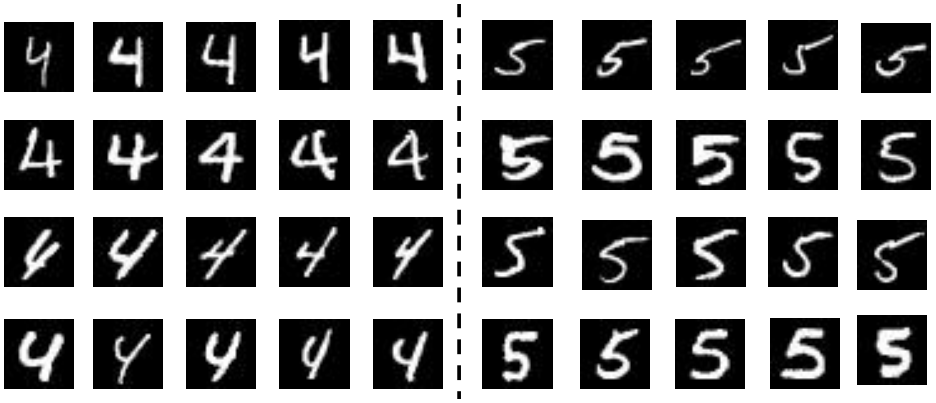

Fig. 4. Inputs sampled from clusters in the class 4 and 5

Fig. 4 shows some examples whose CDPs are clustered into different groups (on SADL-1). Each row represents one cluster. We can observe that, from the human's perception, the inputs in the same cluster look very similar. For example, the digit "4" looks similar to the digit "9" in the fourth cluster and the digit "5" has a slight deflection in the first cluster. The results further show that inputs that trigger the similar decision logic is likely to be grouped into one cluster based on the extracted CDPs.

Answer to RQ2: Based on the aggregation (*i.e.*, union) of all CDPs in each cluster, the abstract CDP is still critical for predicting inputs in a cluster. Moreover, the abstract CDP is sensitive to reveal different decision logic. Inputs in the same cluster have similar decision and inputs in different clusters of the same class tend to have different decision behavior.

4.6 RQ3: Sensitivity with Defect Detection

Setup. Motivated by [20, 28], this experiment evaluates the effectiveness in detecting defects including natural errors[2] and adversarial examples. We follow the similar design in [28]: first, we randomly select 1,000 benign samples (denoted as s_0) from the test data. Then, we construct several test suites by replacing a small number of samples of s_0 with the same number of errors such that they have the same size. To be specific, for natural errors, we select a certain number of inputs from the test data in each class, which are predicted incorrectly. For adversarial examples, we adopt the PGD attack [34] to generate a large number of adversarial examples, some of which are then randomly selected in each class. For natural errors and adversarial examples, we generate 6 test

[2]In this paper, "natural error" represents the natural input that is not manually crafted and is misclassified by the target model.

suites including different numbers of errors (*i.e.*, 1%, 2%, 3%, 5%, 7%, 10%), respectively. We use S to denote the set of the test suites including the initial set s_0. To reduce the randomness, we repeat the process 5 times and calculate the average results.

Results. Fig. 5 shows the results on the test suites including a different number of errors. To conduct the comparison between different approaches, all the coverage is normalized to [-1,1]. Specifically, for each test suite $s \in S$, we calculate its coverage change compared with the initial test suite s_0 with regard to the coverage criterion Cov, *i.e.*, $\Delta = \{Cov(s) - Cov(s_0) | s \in S\}$. For each $s \in S$, we normalize the change of the coverage as follows:

$$NCov(s) = \frac{Cov(s) - Cov(s_0)}{max(\Delta) - min(\Delta)} \tag{4}$$

Note that the coverage change can be a negative value.

The first row shows the average results on natural errors (*Nat.*) and the second row shows the results on adversarial examples (*AE*). The horizontal axis represents the percentage of errors while the vertical axis represents the normalized coverage.

The results show that the coverage criteria exhibit different performance on different number of errors. In general, we could find that SNPC and ANPC tend to be more stable than other methods in terms of the sensitivity to errors. In most cases, the trends of SNPC and ANPC are in line with the expectations, *i.e.*, the coverage can increase when more errors (adversarial examples or natural errors) are introduced. NBC is the most unstable one because it mainly focuses on the boundary behaviors of the model. For KMNC and NC (with threshold 0.5), we observe that they perform better in natural errors than adversarial examples. The coverage of NC and KMNC increases when the number of natural errors increase. However, their coverage does not always increase on adversarial examples. Particularly, KMNC decreases on AEs of CIFAR-VGG16, ALexNet and ImageNet-VGG16. We conjecture that KMNC mainly considers the major functionality of the DNN while ignoring the boundary functionality, thus replacing some benign samples with the corner cases (*e.g.*, adversarial examples) could decrease the diversity of major behavior. LSC performs better in most cases but is insensitive in AlexNet, *i.e.*, the coverage does not change on both adversarial examples and natural error of AlexNet, which indicates that the DNN may have a large impact on LSC. DSC is not sensitive on natural errors while it is better on adversarial examples. It is worth pointing out that most coverage criteria tend to have irregular changes in AE (AlexNet) while SNPC still has an increasing trend.

Considering IDC that is also based on LRP, we can observe that the coverage increase is not as stable as SNPC and ANPC, *e.g.*, on Nat.(AlexNet), AE (SADL-2), AE (AlexNet) and AE (IVGG). Although both IDC and NPC adopt LRP to identify the critical neurons layer by layer, NPC is more fine-grained than IDC, *i.e.*, NPC performs the CDP-based clustering in each class. Thus, our results tend to be more stable.

Moreover, we also observe that there are some correlations between these coverage. Specifically, the trend of coverage increase with regards to different coverage criteria could be similar. For example, in Nat.(SDAL-1), LSC, DSC, ANPC and SNPC have the similar trend. In Nat.(SADL-2), SNPC and NC have the similar trend. Most of the coverage criteria have the similar trend in Nat.(VGG16), Nat.(IVGG) and AE(SADL-1).

Answer to RQ3: Overall, ANPC and SNPC are more sensitive to different errors than other coverage criteria due to the CDP awareness. Different coverage criteria could achieve similar increasing trend, indicating that they tend to have some correlation.

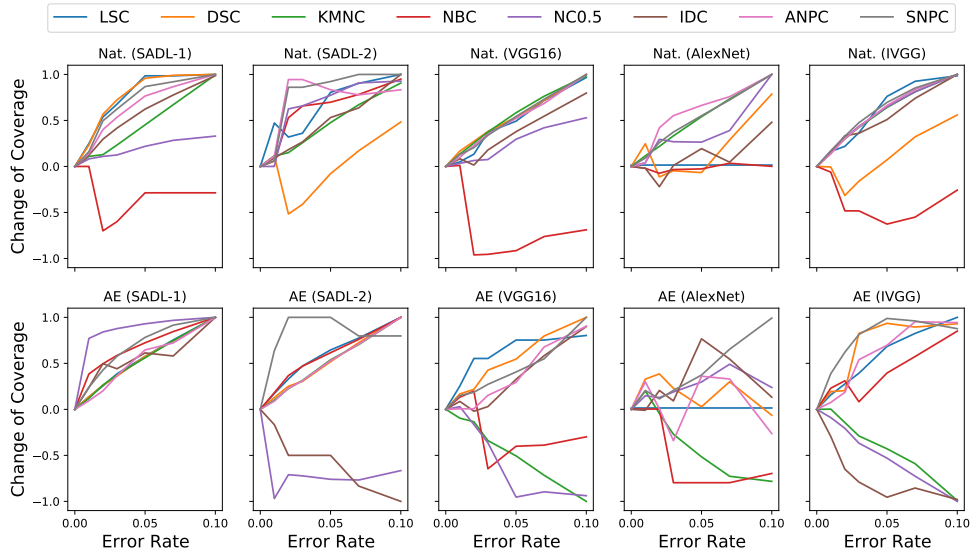

Fig. 5. Coverage change on test suites including different number of errors.

Table 6. Correlation between coverage criteria and output impartiality

		KMNC	NBC	NC(0.0)	NC(0.2)	NC(0.5)	NC(0.75)	LSA	DSA	IDC	ANPC	SNPC
size=100	SADL-1	-0.262	-0.262	-0.300	0.099	0.315	0.001	0.000	-0.638	-0.360	-0.244	**0.723**
	SADL-2	-0.029	-0.029	**0.553**	-0.397	-0.065	-0.111	0.000	0.181	**0.587**	**0.482**	**0.789**
	VGG16	-0.348	-0.348	0.037	-0.031	-0.210	-0.273	0.000	-0.145	0.052	0.012	-0.055
	AlexNet	-0.457	**0.593**	0.000	0.000	0.000	0.000	0.000	-0.320	**0.912**	**0.961**	**0.989**
	Avg.	-0.274	-0.011	0.072	-0.082	0.010	-0.096	0.000	-0.230	**0.298**	**0.601**	**0.612**
size=500	SADL-1	**0.639**	-0.116	0.000	0.000	0.000	0.000	0.000	-0.639	-0.707	**0.340**	**0.870**
	SADL-2	**0.543**	0.041	0.000	0.000	0.000	0.000	0.000	0.330	**0.473**	**0.584**	**0.817**
	VGG16	**0.414**	**0.589**	0.000	0.000	0.000	0.000	0.000	-0.255	**0.210**	**0.141**	**0.821**
	AlexNet	**0.539**	**0.586**	0.000	0.000	0.000	0.000	0.000	-0.170	**0.931**	**0.908**	**0.974**
	Avg.	**0.534**	0.275	0.000	0.000	0.000	0.000	0.000	-0.184	**0.227**	**0.893**	**0.871**

4.7 RQ4: Correlation with Output Impartiality

As described in [2, 20, 51], the test data or errors should be diverse in terms of the output. The existing work [42] has demonstrated that the coverage (*e.g.*, neuron coverage) can be improved by a few samples. However, a test suite that only contains one category of samples (*e.g.*, dog) is not enough to comprehensively reveal diverse defects of the DNN (*e.g.*, the errors in other categories). In addition, the coverage criteria are usually used to guide the test case generation. The coverage criteria that are sensitive to output diversity can guide the testing tools to generate more diverse test cases belonging to different categories while the insensitive coverage criteria may generate biased test cases [51]. Hence, we evaluate the correlation between the coverage criteria and the output diversity in this section.

Setup. We follow the same setting in [20] and use the proposed tool to generate a number of test suites, which have the same size. Specifically, it mainly contains three steps:

(1) For each dataset, we randomly select two seed test suites with different size (*i.e.*, 100 and 500).

(2) Based on each seed test suite, we adopt the tool [20] to generate new test suites that have the same size with the seed test suite (*i.e.*, generate new input by perturbing each input of the seed test suites). The tool [20] extends the existing adversarial attacks (*e.g.*, C&W [6] and PGD [34]) with the divergence function such that more diverse test cases can be generated. By setting different configurations (*e.g.*, diversity regularization weight, confidence), multiple test suites can be generated. For each seed test suite on SADL-1, SADL-2, VGG-16 (CIFAR-10) and AlexNet, we generate 48, 171, 167 and 168 new test suites[3], respectively. Note that, we did not include the results of ImageNet in this experiment because all adversarial examples generated by the tool [20] belong to the same class, which makes all correlation scores as NaN.
(3) For a generated test suite, we calculate the coverage and the output diversity, respectively. Finally, we calculate the *Pearson* correlation between them.

Specifically, we use the output impartiality metric [20] for a test suite T:

$$output\_impartiality(T) = \frac{\sum_{t \in C_k} P_{t=C_k} log P_{t=C_k}}{\frac{1}{|c|} log \frac{1}{|c|}}$$

where $|C|$ is the cardinality of classes and $P_{t=C_k}$ represents the percentage of the test cases t predicted to belong to class C_k.

Results. Table 6 shows the correlation on different models. The overall results demonstrated that, compared with existing coverage criteria, ANPC and SNPC are more strongly correlated with the output impartiality. In particular, we have the following findings:1) Compared with the test suite with smaller size (*e.g.*, 100), the correlation can be stronger when the size of the test cases becomes larger (*i.e.*, 500). For example, for the size 500, ANPC and SNPC are positively and strongly correlated with the output impartiality in all models while there are some negative correlations in the size 100. KMNC is also positively correlated with output impartially when the size is 500. 2) Considering the results of ANPC and SNPC, we found that SNPC is more strongly correlated with output comparability. It is because ANPC is more fine-grained and could capture the minor activation difference (*i.e.*, coverage increase) even when they have output bias (*i.e.*, the same prediction output).

For the results that ANPC and SNPC achieve negative correlation (*e.g.*, -0.244 on SADL-1, -0.055 on VGG16) when the size of the test suite is 100, our in-depth analysis reveals that, for some test suites with smaller size, the prediction outputs of the test cases are biased (*e.g.*, these test cases are predicted as the same class), which caused the low output impartiality. However, these samples can still trigger different CDPs (*i.e.*, different similarities compared with the corresponding CDP), which lead to high coverage. Hence, the correlation is negative. When the size becomes larger (*e.g.*, 500), the test suite becomes more stable (*i.e.*, the prediction results are less biased), which makes that the coverage is positively correlated with the output impartiality.

We also observe that the correlation of IDC is higher than other baselines but it is still less than ANPC and SNPC. The average correlation scores of IDC are less than ANPC and SNPC. For example, when the size is 100, the average score of IDC is 0.298 while the average scores of ANPC and SNPC are 0.601 and 0.612, respectively. Except for VGG16 (size=100), all correlation scores of SNPC are larger than IDC. The results indicate the effectiveness of our fine-grained coverage criteria.

Answer to RQ4: Generally, ANPC and SNPC are positively correlated with the output impartiality. The model and the size of test suite could affect the correlation.

[3]The number of test suites is different because it depends on the number of configurations in [20] (*i.e.*, a new test suite is generated by a configuration), where one variable in the configurations is the layers of the neuron network.

Table 7. The time overhead of coverage criteria on different models (seconds)

	ANPC	SNPC	KMNC	NBC	LSC	DSC	NC(0.0)	IDC
SADL-1	10.25(2.60)	8.83(2.60)	3.50	1.92	26.25	59.75	0.67	1.11
SADL-2	39.50(14.45)	47.33(14.45)	26.75	13.67	24.67	166.17	1.33	0.82
VGG16(CIFAR-10)	135.50(35.10)	91.17(35.10)	54.67	28.75	21.00	73.00	2.17	2.81
AlexNet	57.00(13.35)	29.50(13.35)	104.33	53.25	37.08	136.00	1.08	1.21
VGG16(ImageNet)	203.24(59.67)	119.75(59.67)	146.43	88.02	154.27	173.27	30.45	13.21

4.8 RQ5: Efficiency of NPC

We use the same dataset from RQ3 and compare the time overhead of calculating the coverage of a test suite (*i.e.*, 1000 samples). Note that, except for NC, all other coverage criteria depend on the training samples and there are some prepossessing process. For example, KMNC and NBC need to profile the training data. LSC and DSC need to calculate all activation values of the training data. ANPC and SNPC need to calculate the CDPs for all training data. IDC also requires the LRP calculation for all training data. Since the prepossessing phase can be executed only once and it is usually calculated offline, we ignore the time of the prepossessing and only compare the coverage calculation time for a given test set.

Table 7 shows the results. We found that, NC (with the threshold 0.0) is the most efficient one as it only depends on the neuron output. For ANPC and SNPC, we list the total time and the CDP path extraction time (the number in brackets).The results show that the path extraction is efficient. Compared with others, ANPC and SNPC take more time since more information are required (*i.e.*, the CDP calculation). IDC is more efficient because it only calculates the distance between the critical neuron values of the test cases and the training samples at one layer. Actually, it is a trade-off between effectiveness (*i.e.*, leveraging more information) and the efficiency.

Answer to RQ5: Although ANPC and SNPC adopt the interpretation analysis, the time overhead is not expensive. The CDP extraction is efficient and it only takes a proportion in the total time overhead of the coverage calculation.

5 THREATS TO VALIDITY AND DISCUSSION

Threats to Validity. The internal threats to validity lies in our implementations including the LRP implementation, the path extraction, the path abstraction, the coverage criteria and the implementation in each experiment. To reduce this threat, the co-authors carefully checked the correctness of our implementation.

The external threats to validity include the selection of the subjects and the tools used in the evaluation. Specifically, the selection of datasets and the used models could be a threat. To reduce this threat, we selected four widely-used subjects (MNIST, CIFAR-10, SVHN and ImageNet) and 4 DNNs (SADL-1, SADL-2, VGG-16 and AlexNet), where ImageNet is a large-scale dataset. Another threat could be the generated data including adversarial examples, the selected natural errors and test suites (in RQ3, RQ4 and RQ5). To mitigate this threat, we follow the same setting in the existing work [20, 28] and generate diverse data with multiple configurations. In addition, the usage of the existing tool [20] can be a threat. Since our method, the used models and the tool [20] are developed by PyTorch, we manually re-implement the tool [15, 23] by PyTorch, which may not the exactly the same with the original version. We integrate our method, LSC, DSC, NC, KMNC and NBC into the tool [20] to evaluate the correlation. We carefully checked the code to mitigate the threat.

The construct threats mainly lie in the randomness, the selected baselines, the parameters. First, one threat is the randomness in the selection of the data. To mitigate the treat, in RQ3, we repeat each experiment 5 times (*e.g.*, the generation of adversarial examples) and calculate the average results. In RQ4, we follow the setting [20] and generate a large number of test suites to reduce the randomness. Second, we adopted the state-of-the-art approaches that have been demonstrated to be insufficient in DL testing. We compare our method with them to demonstrate the advantages of our proposed coverage criteria. Thirdly, the hyper-parameters in our methods and the baselines could be a threat to validity. To reduce the threat, for the baselines, we follow the settings in the existing works [20, 23, 31]. For our method, we set multiple configurations in terms of the hyper-parameters α, β and the number of clusters k. However, in RQ3 and RQ4, we select the same values of α, β and k for all paths and classes. Actually, we can select the optimal configurations by adopting different α, β and k. In the future, we plan to evaluate our method with more configurations.

Limitation. Like the coverage criteria in traditional software, we think there is no one best coverage criterion and each one has some limitations. The main advantage of NPC lies in its interpretability and the clear relationship with the internal decision of the DNN. In this section, we summarize our limitations and discuss the potential solutions in the future.

- Currently, our approach mainly considers the classification since the current interpretation technique (*i.e.*, LRP) does not support the DNN with regression prediction. Even though the used interpretation technique is orthogonal to our approach, our approach is general as long as the critical neurons could be calculated. We believe the interpretation techniques that work on regression tasks in the future will further solve this limitation.
- Our approach mainly works on feed-forward neural networks. For recurrent neural networks (RNNs), whose number of iterations is not fixed, it could be challenging for the interpretation technique (*e.g.*, relevance calculation in LRP). In the future, we may extend LRP by unrolling the RNN to address this limitation.
- There is another limitation on the selection for the number of the clusters during the path abstraction. Currently, we select the same number for all classes, which may not be the optimal setting. In the future, we will investigate the automated selection on this parameter.

6 RELATED WORK

This section summarizes the related work on DL testing and adversarial attacks.

6.1 DL Testing

DeepXplore [39] proposes the first white-box coverage criteria, *i.e.*, Neuron Coverage, which calculates the percentage of activated neurons. A differential testing approach is proposed to detect the errors by increasing NC. DeepGauge [31] then extends NC and proposes a set of more fine-grained coverage criteria by considering the distribution of neuron outputs from training data.

Inspired by the coverage criteria in traditional software testing, some coverage metrics [30, 32, 43] are proposed. DeepCover [43] proposes the MC/DC coverage of DNNs based on the dependence between neurons in adjacent layers. DeepCT [30] adopts the combinatorial testing idea and proposes a coverage metric that considers the combination of different neurons at each layer. DeepMutation [32] adopts the mutation testing into DL testing and proposes a set of operators to generate mutants of the DNN. Furthermore, Sekhon *et al.* [42] analyzed the limitation of exiting coverage criteria and proposed a more fine-grained coverage metric that considers both of the relationships between two adjacent layers and the combinations of values of neurons at each layer.

Based on the neuron coverage, DeepPath [48] initially proposes the path-driven coverage criteria, which considers the sequentially linked connections of the DNN. Although both of DeepPath and our method focus on path-based coverage, the approach and the underlying principles are totally different. The path in DeepPath is the *syntactic* connections between neurons in different layers and the semantics of the path is unknown. Differently, we adopt the interpretation techniques to define paths, in which the neurons are selected based on the relevance with the decision. Thus, CDPs have more clear *semantics*, *i.e.*, they are related to the decision logic.

The aforementioned techniques mainly consider syntactic structure of DNN, which is black-box and hard to understand. Differently, Kim *et al.* [23] proposed the coverage criteria that measure the surprise of the inputs. The assumption is that surprising inputs introduce more diverse data such that more behaviors could be tested. Surprise metric measures the surprise score by considering *all* neuron outputs of one or several layers. It is still unclear how the surprise coverage (calculated from some layers) is related to the decision logic. Differently, our method not only selects the *critical* neurons at each layer but also considers the relationships between layers.

Based on the these criteria, some automated testing techniques [11, 37, 39, 43, 46, 51, 55] are proposed to generate test inputs towards increasing the coverage. In addition, while the coverage criteria are widely studied, the existing work [10, 13, 20, 28, 42, 52] found some preliminary evidence about the limitation of the *structural* criteria. For example, the coverage is too coarse and is not correlated to the defect detection and robustness. Such findings motivate this work that proposes *interpretable* coverage criteria by extracting the semantic structure (*i.e.*, the decision graph).

Recently, some coverage criteria [1, 15] are also proposed based on the decision of the DNN. The technique [1] adopts the explanation technique LIME [40] to extract a decision tree that explains the decision of the DNN on an input. Based on the decision tree, the symbolic execution is used to generate test cases that can maximize the path coverage on the decision tree. Different with [1], we extract the decision graph that could represent the global decision behaviors of the DNN. Moreover, in [1], it mainly focuses on the problem of fairness evaluation while it is hard to extract such a decision tree on the high dimensional data such as image domain. IDC [15] adopts the interpretation technique to select the important neurons in one layer. Based on the training data, it then groups the activation values of important neurons into a set of clusters and uses the clusters to measure the coverage. Similar with our technique, IDC also selects the critical neurons. Differently, NPC considers the relations between layers and proposes the path-based coverage instead of only one layer. Thus, NPC does not need to select one of the target layers. In addition, we propose two different path coverage criteria from the control flow and data flow of the DNN. Significantly, there are also two papers [53] and [27] extracting critical neurons to represent decision behaviors. However, they focus on adversarial attacks and there are relatively larger computation costs. Except for the testing on the trained model, some techniques (*e.g.*, [50]) have been proposed to build the self-checking system that can monitor DNN output and trigger an alarm if the output is likely to be incorrect after the model is deployed. For more relevant discussions on the recent progress of machine learning testing, we refer the interesting readers to the recent comprehensive survey [54].

6.2 Adversarial Attacks

Recently, there are many adversarial attack techniques such as adversarial noise attacks including *FGSM* [16], *JSMA* [38], *BIM* [25], *DeepFool* [35], and *C&W* [6], and natural degradation-based adversarial attacks including *ABBA* [17], *SPARK* [18], *AVA* [45], and *Pasadena* [8]. The adversarial examples are generated by adding visually imperceptible perturbation to an input such that the new input is misclassified. It is worth to think and discuss about the difference between deep learning testing and adversarial attacks. The common thing is that both the testing and attack could test

the robustness of the model by generating incorrect inputs. Differently, the adversarial attack is ad-hoc and generates the *specific* adversarial examples based on gradient calculation. DL testing aims to test the DNN more systematically by exploring more decision logic of the model. Thus, the errors generated by DL testing are expected to be more diverse (*i.e.*, triggering different logic) than adversarial attacks. For example, given one input, the attack techniques usually generate the similar adversarial examples while DL testing could generate different adversarial examples with the guidance of the coverage criteria.

7 CONCLUSION

Motivated by the recent findings [10, 20, 28] about the limitation of the existing DL testing criteria, this paper proposes the neuron path coverage via extracting the decision structure of the targeted DNN. For a DNN, we extract the decision logic based on the critical decision paths from the training data. Two path-based coverage criteria are then proposed to measure whether new decision logic is covered by the test cases. Specifically, SNPC is designed based on the control-flow of the DNN which measures whether different critical neurons are covered; ANPC is designed based on the data-flow of the DNN which measures the activation values of the critical neurons. To the best of our knowledge, this is the first work that proposes the interpretable coverage criteria from the control-flow and data-flow of the DNN. The evaluation results demonstrated the effectiveness of the CDP in the decision revealing of the DNN and the usefulness of the coverage criteria.

ACKNOWLEDGMENTS

This research is partially supported by the National Research Foundation, Singapore under its the AI Singapore Programme (AISG2-RP-2020-019), the National Research Foundation, Prime Ministers Office, Singapore under its National Cybersecurity R&D Program (Award No. NRF2018NCR-NCR005-0001), NRF Investigatorship NRFI06-2020-0022-0001, the National Research Foundation through its National Satellite of Excellence in Trustworthy Software Systems (NSOE-TSS) project under the National Cybersecurity R&D (NCR) Grant Award No. NRF2018NCR-NSOE003-0001, the Ministry of Education, Singapore under its Academic Research Fund Tier 3 (MOET32020-0004), the JSPS KAKENHI Grant No.JP20H04168, JP19K24348, JP19H04086, JP21H04877 and JST-Mirai Program Grant No.JPMJMI20B8, Japan. Lei Ma is also supported by Canada CIFAR AI Program and Natural Sciences and Engineering Research Council of Canada. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

REFERENCES

[1] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black Box Fairness Testing of Machine Learning Models. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019)*. Association for Computing Machinery, New York, NY, USA, 625–635. https://doi.org/10.1145/3338906.3338937

[2] Nadia Alshahwan and Mark Harman. 2014. Coverage and fault detection of the output-uniqueness test selection criteria. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. 181–192.

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.

[4] BBC. 2020. *AI 'outperforms' doctors diagnosing breast cancer.* https://www.bbc.com/news/health-50857759

[5] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 387–402.

[6] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy*. 39–57.

[7] Z. Chen and X. Huang. 2017. End-to-end learning for lane keeping of self-driving cars. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. 1856–1860.

[8] Yupeng Cheng, Qing Guo, Felix Juefei-Xu, Wei Feng, Shang-wei Lin, Weisi Lin, and Yang Liu. 2021. Pasadena: Perceptually Aware and Stealthy Adversarial Denoise Attack. *IEEE Transactions on Multimedia* (2021), 1–1.

[9] Datamonsters. 2017. 10 Applications of Artificial Neural Networks in Natural Language Processing. https://medium.com/@datamonsters/artificial-neural-networks-in-natural-language-processing-bcf62aa9151a

[10] 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 [cs.LG]

[11] 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 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019)*. Association for Computing Machinery, New York, NY, USA, 477–487. https://doi.org/10.1145/3338906.3338954

[12] Carnegie Endowment. 2019. The Global Expansion of AI Surveillance. https://carnegieendowment.org/2019/09/17/global-expansion-of-ai-surveillance-pub-79847

[13] 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* (Virtual Event, USA) *(ISSTA 2020)*. Association for Computing Machinery, New York, NY, USA, 177–188. https://doi.org/10.1145/3395363.3397357

[14] Ruth C Fong and Andrea Vedaldi. 2017. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*. 3429–3437.

[15] Simos Gerasimou, Hasan Ferit Eniser, Alper Sen, and Alper Cakan. 2020. Importance-Driven Deep Learning System Testing. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 322–323. https://doi.org/10.1145/3377812.3390793

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*. http://arxiv.org/abs/1412.6572

[17] Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, Jian Wang, Bing Yu, Wei Feng, and Yang Liu. 2020. Watch out! Motion Is Blurring the Vision of Your Deep Neural Networks. *NeurPIS* 33 (2020).

[18] Qing Guo, Xiaofei Xie, Felix Juefei-Xu, Lei Ma, Zhongguo Li, Wanli Xue, Wei Feng, and Yang Liu. 2020. SPARK: Spatial-aware online incremental attack against visual tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Vol. 2. Springer.

[19] Rahul Gupta, Aditya Kanade, and Shirish Shevade. 2019. Neural Attribution for Semantic Bug-Localization in Student Programs. In *Advances in Neural Information Processing Systems*. 11884–11894.

[20] 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 European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM.

[21] Lingxiao Jiang, Ghassan Misherghi, Zhendong Su, and Stephane Glondu. 2007. DECKARD: Scalable and Accurate Tree-Based Detection of Code Clones. In *Proceedings of the 29th International Conference on Software Engineering (ICSE '07)*. 96–105.

[22] Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically Generating Commit Messages from Diffs Using Neural Machine Translation. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017)*. IEEE Press, 135–146.

[23] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System Testing Using Surprise Adequacy. In *Proceedings of the 41st International Conference on Software Engineering* (Montreal, Quebec, Canada) *(ICSE '19)*. 1039–1049.

[24] Nair Krizhevsky, Hinton Vinod, Christopher Geoffrey, Mike Papadakis, and Anthony Ventresque. 2014. The cifar-10 dataset. http://www.cs.toronto.edu/kriz/cifar.html.

[25] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[26] Yann LeCun and Corrina Cortes. 1998. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[27] Tianlin Li, Aishan Liu, Xianglong Liu, Yitao Xu, Chongzhi Zhang, and Xiaofei Xie. 2021. Understanding adversarial robustness via critical attacking route. *Information Sciences* 547 (2021), 568–578. https://doi.org/10.1016/j.ins.2020.08.043

[28] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. 2019. Structural coverage criteria for neural networks could be misleading. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 89–92.

[29] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

[30] 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 *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 614–618.

[31] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131.

[32] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 100–111.

[33] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. 2019. Test Selection for Deep Learning Systems. arXiv:1904.13195 [cs.LG]

[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

[35] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 2574–2582.

[36] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning* (2011).

[37] Augustus Odena and Ian Goodfellow. 2019. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing. In *Proceedings of the Thirty-sixth International Conference on Machine Learning*.

[38] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 372–387.

[39] 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.

[40] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[41] 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.

[42] Jasmine Sekhon and Cody Fleming. 2019. Towards improved testing for deep learning. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 85–88.

[43] 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.

[44] TechCrunch. 2020. Nearly 70% of US smart speaker owners use Amazon Echo devices. https://techcrunch.com/2020/02/10/nearly-70-of-u-s-smart-speaker-owners-use-amazon-echo-devices/

[45] Binyu Tian, Felix Juefei-Xu, Qing Guo, Xiaofei Xie, Xiaohong Li, and Yang Liu. 2021. AVA: Adversarial Vignetting Attack against Visual Recognition. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 1046–1053.

[46] 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*. ACM, 303–314.

[47] Uber Accident. 2018. After Fatal Uber Crash, a Self-Driving Start-Up Moves Forward. https://www.nytimes.com/2018/05/07/technology/uber-crash-autonomous-driveai.html

[48] Dong Wang, Ziyuan Wang, Chunrong Fang, Yanshan Chen, and Zhenyu Chen. 2019. DeepPath: Path-Driven Testing Criteria for Deep Neural Networks. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 119–120.

[49] Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. 2018. Interpret neural networks by identifying critical data routing paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8906–8914.

[50] Yan Xiao, Ivan Beschastnikh, David S Rosenblum, Changsheng Sun, Sebastian Elbaum, Yun Lin, and Jin Song Dong. 2021. Self-Checking Deep Neural Networks in Deployment. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 372–384.

[51] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. 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 (ISSTA 2019)*. 146–157.

[52] 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.* Association for Computing Machinery, New York, NY, USA, 775–787. https://doi.org/10.1145/3368089.3409671

[53] Chongzhi Zhang, Aishan Liu, Xianglong Liu, Yitao Xu, Hang Yu, Yuqing Ma, and Tianlin Li. 2021. Interpreting and Improving Adversarial Robustness of Deep Neural Networks With Neuron Sensitivity. *IEEE Transactions on Image Processing* 30 (2021), 1291–1304. https://doi.org/10.1109/TIP.2020.3042083

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

[55] Xiyue Zhang, Xiaofei Xie, Lei Ma, Xiaoning Du, Qiang Hu, Yang Liu, Jianjun Zhao, and Meng Sun. 2020. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 739–751.

[56] Hong Zhu, Patrick A. V. Hall, and John H. R. May. 1997. Software Unit Test Coverage and Adequacy. *ACM Comput. Surv.* 29, 4 (1997), 366–427.