Weight-based Semantic Testing Approach for Deep Neural Networks

Amany Alshareef[1,*], Nicolas Berthier[1,2], Sven Schewe[1] and Xiaowei Huang[1]

[1]*University of Liverpool, Liverpool L69 3BX, United Kingdom*
[2]*OCamlPro, France*

Abstract
While deep learning models have achieved state-of-the-art performance in a variety of fields, their susceptibility to adversarial examples has raised serious concerns over their application in safety-critical domains. Existing testing methodologies fail to consider interactions between neurons and the semantic representation that formed in the DNN through the training process. This paper proposes a weight-based testing metric that uses feature importance weights to measure the coverage of the test set and facilitates the generation of additional test cases targeting higher weights' features. Evaluations were conducted to compare the initial and final coverage of the proposed weighting approach with normal BN-based feature coverage. The testing coverage experiments indicated that the proposed weight metrics achieved higher coverage compared to the original feature metrics while maintaining the effectiveness of finding adversarial samples during the test case generation process.

Keywords
DNN testing metrics, Bayesian abstraction, Feature coverage, Importance weights

1. Introduction

Software testing provides evidence to demonstrate that the system meets its requirements or is error-free. The fact that deep learning models are data-driven, not requirements-driven, makes defining their testing criteria challenging. Technically, the accuracy of the learning models is reported based on the test dataset. This standard metric for measuring the model's overall performance cannot be sufficient or trustworthy in the safety-related domain, where most testing scenarios are randomly chosen from the entire dataset. Besides, the provided test data may not have good coverage of the data distribution the model is trained on and may not represent the data obtained in the real world.

Furthermore, most of the current proposed DNN testing techniques rely on neuron activation as a metric to measure the test data coverage. Such a criterion does not prove its correlation to the system's decision logic [1]. Moreover, these methods aim to transform the input data space to generate more test input and completely ignore the model-internal representations and their roles in the output decision. Observing that real-world high-dimensional data lie on low-dimensional manifolds motivates investigating where the data lie and modeling it to be analysed instead of confined to the input domain. There is little attempt to understand machine learning's hidden representations and generate additional test cases based on them. This paper presents a testing approach for neural networks that leverages the learned representations and feature importance to evaluate the test data's coverage. The features' importance weights reflect how the contribution of each hidden feature, extracted from the lower-dimensional latent feature space, to the overall output of the network is distributed across the network. This identified feature contribution enables the determination of the causal relationship between the neurons and the model behaviour. Therefore, the importance weights relate to the semantic representation and provide insights into the interaction mechanism underlying the output decision-making process.

The proposed approach to design weight-based semantic testing metrics for neural networks is using the Bayesian network abstraction model of Berthier et al. [2]. The authors introduced a dimensionality reduction technique using feature extraction algorithms to abstract the behaviour of a neural network into a Bayesian network (BN). The work in [3] utilised that BN model to quantify the importance of a neural network's latent features. It developed a BN-based sensitivity analysis algorithm that estimates the importance of a neural network's latent features by analysing an associated BN's sensitivity to distributional shifts. They integrated various metrics to compute the difference between the original probability distributions represented by the abstracted BN and the distributions obtained after perturbation. Each latent feature was then assigned a weight value based on the measured sensitivity distance.

In this work, we transform the traditional binary coverage approach to a weighted probability problem and define our coverage metric based on the latent features

The IJCAI-2023 AISafety and SafeRL Joint Workshop.
*Corresponding author.
amany.alshareef@liverpool.ac.uk (A. Alshareef); nicolas.berthier@ocamlpro.com (N. Berthier); sven.schewe@liverpool.ac.uk (S. Schewe); xiaowei.huang@liverpool.ac.uk (X. Huang)

importance. The neural network's latent feature space refers to the internal, hidden representations learned by the network. These representations are not directly observable but rather are formed as a result of the network processing input data and capturing their important patterns [4]. The proposed weight-based testing criterion emphasises that maximum test coverage is obtained from the presence of important features that have a dominating influence on other features and the output decision. To summarise, the main contributions of the paper are :

- Semantic testing metrics measure a test dataset's coverage based on the calculated feature weights.
- A guided systematic approach that samples test cases targeting the higher-priority features.
- Empirical studies on the quality of the proposed weight-based coverage compared with the original BN-based coverage

2. Existing Testing Techniques

Deep neural network testing is an active research area where safety-critical applications are being deployed with them. Numerous techniques have been developed to address the challenges of testing these learning systems in terms of test coverage criteria, test generation, and test oracles.

2.1. Testing Metrics and Coverage Criteria

Testing coverage metrics are measurements used to evaluate the adequacy of testing by providing a quantitative evaluation of how thoroughly a deep neural network has been tested according to specific criteria. Coverage-guided deep neural network testing techniques are a class of testing methods that aim to increase the coverage of the network during testing, with the goal of covering different regions of the input space and revealing as many potential bugs and unexpected behaviours as possible. Enforcing higher coverage during the testing process makes the network under investigation more likely to be robust and reliable. We divide the related existing testing works in the literature into two categories:

(i) Structural coverage metrics that are defined based on the syntactic characteristics of the NNs, and

(ii) High-level semantic coverage metrics that are concentrated on the semantic representations created by NNs.

Most proposed testing approaches have been focused on the *structural testing coverage* to measure the coverage of the dataset relied on the individual neuron activation. These techniques are based on the idea of activating as many neurons in the network as possible during the testing phase. The more neurons that are activated with a specific value, the more complete the testing of the network is considered to be.

Neuron activation was firstly introduced by Pei et al. [5] as a systematic metric which calculates the number of activated neurons (w.r.t. ReLU activation function) during the testing. They proposed the DeepXplore which is a white box differential testing algorithm for generating test inputs that can discover inconsistencies between multiple DNNs. Following the neuron coverage (NC) principle, DeepTest [6] and Dlfuzz [7] have made some improvements to it. Although the NC metric has been shown to be effective at finding hidden bugs and has been used to test real-world DNNs, investigations by [8] demonstrated that NC is too coarse and easy to achieve.

Further approaches such as DeepGauge [9], and DeepCover [8] have been developed to extend neuron coverage, with a focus on various activation value factors. Beyond that, more testing metrics, i.e., quantitative projection coverage [10], safety coverage [11] and surprise coverage [12], have been designed based on the activation functions and the syntactic connections between neurons in successive layers.

Unfortunately, neuron activation and other structural coverage techniques have proven to be less effective in validating the safety behaviours of intelligent systems. A study by Li et al. [13] showed that there is no correlation between the number of misclassified natural input tests and their structural coverage on the corresponding neural networks. There is still considerable ambiguity about how such coverage criteria directly relate to the decision logic of black-box machine learning systems. Especially in that case, the semantic relationship between layers is ignored. Additionally, structural coverage has a limited correlation with network robustness, where high neuron coverage does not imply the network is robust to all possible inputs or will behave well on unseen data [1, 14].

There are relatively few testing strategies that address the semantic aspects of DNN's internal representation. One recent effort is the BN-based feature coverage introduced in [2] that is improved with weights in this study. Two testing coverage metrics are defined based on the suggested BN abstraction: the BN-based feature coverage (BFCov) and the BN-based feature-dependence coverage (BFdCov). These metrics give the proportion of hidden features or causal relationships between them that are adequately exercised by a set of inputs. Moreover, the authors implemented a combined metric $\mathrm{BFxCov}(\mathcal{B}_{\mathcal{N}}, \mathrm{x})$ of the two above as: $\mathrm{BFCov}(\mathcal{B}_{\mathcal{N}}, \mathrm{x}) \times \mathrm{BFdCov}(\mathcal{B}_{\mathcal{N}}, \mathrm{x})$. For the space limit, we include the BFCov metric below and refer the reader to the original document for the rest.

Definition 2.1 (BN-based Feature Coverage). *Given a trained DNN $\mathcal{N}$, the* $\mathrm{BFCov}(\mathcal{B}_{\mathcal{N}}, \mathcal{X})$ coverage *of a* non-empty *set of inputs* $X \subset \mathbb{D}_X$ *is obtained via the BN abstraction* $\mathcal{B}_{\mathcal{N},X}$ *as*

$$\frac{1}{\left|V_{\mathcal{N},X}\right|} \sum_{\langle\!\langle f^{\sharp}_{i,j} \rangle\!\rangle \in V_{\mathcal{N},X}} \frac{\left|\left\{ f^{\sharp k}_{i,j} \in \mathbb{F}^{\sharp}_{i,j} \mid \mathcal{P}_i\left(f^{\sharp k}_{i,j}\right) \geq \varepsilon \right\}\right|}{\left|\mathbb{F}^{\sharp}_{i,j}\right|}, \tag{1}$$

Informally, $\mathrm{BFCov}(\mathcal{B}_{\mathcal{N},\mathrm{X}})$ ranges over [0, 1], and gives the percentage of feature intervals that are adequately exercised by X. Intuitively, the coverage metric checks the marginal probabilities $\mathcal{P}_i$ for every interval $f^{\sharp k}_{i,j} \in \mathbb{F}^{\sharp}_{i,j}$ in the BN's node that appears with a probability bigger than ε.

Furthermore, a closely related work to the proposed importance-based semantic testing is the DeepImportance approach presented in [15]. The authors developed a testing approach based on the Importance-Driven (IDC) test adequacy criterion that employs the layer-wise relevance propagation to identify the important neurons. They then evaluated the test's adequacy by targeting different combinations of important neurons' behaviours.

2.2. Test Cases Generation Algorithms

The existing set of test case generation algorithms for DNNs is categorised into: *(i) Input mutation based,* which generates new test inputs, either natural or adversarial, by altering the original data using transformation rules; *(ii) Fuzzing based,* which generates invalid random input data to detect faults and vulnerabilities in the model; and *(iii) Symbolic execution based,* which is an analysis technique that tests whether specific inputs cause each part of a system to be executed.

Concolic testing is a testing technique in which concrete execution directs the symbolic analysis to generate a high coverage test suite. The DeepConcolic introduced in [16] used a concolic testing algorithm that alternates concrete executions, which evaluate the test input using the trained DNN under test, and symbolic analyses, which synthesise new test inputs based on some test target that is chosen to increase coverage.

2.3. Test Cases Evaluation

A **Test oracle** is a reference or ground truth that provides the expected output for a given input and is used to compare the output of the system to determine its accuracy.

Overall, the structural coverage criteria focus on the patterns that appear in the outputs of ReLU activation functions, while the semantic coverage metrics are high-level criteria that focus on the features that have been learned by hidden layers of the DNN. The proposed semantic metrics are based on the model-internal representations and their contribution to the output behaviours.

3. The BN Weighted Feature Model

The goal of the weighted feature model is to construct an abstracted Bayesian network that includes the importance weight for each node of the BN. We first review the main components of building the BN abstraction. The creation of the BN model goes through the following:

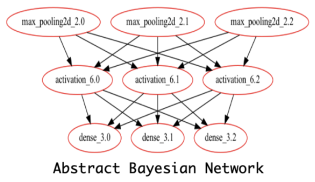
1. **Extraction of hidden features.** In this stage, the hidden features learned by the DNN layers are identified by using feature extraction techniques, *i.e.,* Principal Component Analysis (PCA) and Independent Component Analysis (ICA), on neuron activation values induced by a given training set;
2. **Feature space discretisation.** The extracted features range over a continuous space, therefore, each feature component is discretised into a finite set of feature intervals according to various strategies, i.e., density- and uniform-based;
3. **Bayesian network construction.** This consists in representing the probabilistic distribution of each extracted feature with a node in the BN. Each node is associated with either a marginal probability table for hidden features of the first layer, or a conditional probability table (CPT) for hidden or output layers.

Preliminaries. The BN $\mathcal{B}_{\mathcal{N},X} = (V, E, P)$ is an abstracted model constructed from the DNN $\mathcal{N}$ and training dataset X. The V are nodes containing the extracted latent features from the $\mathcal{N}$. Each feature is defined as a pair $f_{i,j}$ and partitioned into a finite set of m intervals denoted with $^\sharp$ exponents. The E are directed edges indicating dependencies between features in successive layers, and P maps each node in V to a probability table representing the conditional probability of the current feature over its parent features w.r.t. X.

The feature sensitivity analysis process discussed in [3] is calculated based on the change in the BN's probability distribution as follows:

$$W_{i,j} = \frac{\delta(P_{ref}, P'_f)}{\sum_{f \in \mathbb{F}^\sharp} \delta(P_{ref}, P'_f)} \tag{2}$$

Where W_{ij} is the sensitivity weight of the feature $f^{\sharp}_{i,j}$, P_{ref} is the original (reference) probability distribution represented by the BN, P'_f is the probability after perturbing $f_{i,j}$, and δ is a function returning the distance between two probabilities distribution according to a given metric d_p. $\mathbb{F}^\sharp$ is the set of considered latent features.



distances perturbed feature	d_L2	d_L∞	d_JS	d_Wass1	d_corr	d_cos	d_RMSE	d_MAE	d_AF
(max_pooling2d_2, 0)	0.074784	0.110878	0.087355	0.111071	0.063686	0.069357	0.111033	0.111073	0.056730
(max_pooling2d_2, 1)	0.117755	0.111138	0.109214	0.111083	0.112240	0.110499	0.111125	0.111114	0.120953
(max_pooling2d_2, 2)	0.092172	0.111144	0.099690	0.111079	0.082057	0.084531	0.111076	0.111092	0.077190
(activation_6, 0)	0.120717	0.111142	0.126169	0.111135	0.131510	0.127733	0.111130	0.111121	0.127265
(activation_6, 1)	0.115556	0.111138	0.125377	0.111122	0.109105	0.115235	0.111121	0.111121	0.116449
(activation_6, 2)	0.124626	0.111144	0.121799	0.111125	0.136390	0.137618	0.111137	0.111123	0.136042
(dense_3, 0)	0.120537	0.111135	0.114257	0.111129	0.130693	0.129624	0.111130	0.111120	0.126873
(dense_3, 1)	0.116058	0.111144	0.107665	0.111129	0.110986	0.106896	0.111122	0.111116	0.117464
(dense_3, 2)	0.117794	0.111137	0.108475	0.111127	0.123333	0.118508	0.111125	0.111118	0.121034

Figure 1: Example of computed distances from the feature sensitivity weighting method introduced in [3], annotated with the used distance metrics d_p at the header. The diagram illustrates the used structure of the Bayesian Network.

The previous equation represents unnormalised weights, which is further normalised: $\sum_{i,j} W_{i,j} = 1$. Thus, we obtain W_f, which is a vector of the computed weights for the extracted latent features from the DNN.

Definition 3.1 (Weighted Feature Model). *A weighted feature model over a hidden DNN's features $\mathbb{F}^\sharp$ is a function $f\colon \mathbb{F}^\sharp_{i,j} \rightarrow \mathbb{R}^{\geq 0}$ that maps features into their importance values according to their sensitivity weights resulting from selected d_p.*

Example 1. *Figure 1 shows a BN constructed from three selected layers of a CNN: max_pooling2d_1, activation_6, and dense_3. The activation values are computed for each considered layer, then the dimensionality reduction is applied and produced three feature component that is discretised into five intervals. The figure illustrates an example of the latent features' sensitivity weights obtained using various distance metrics.*

4. Weight-based Testing

This section provides a detailed technical description of the proposed weight-based semantic testing metrics and algorithm. We first introduce and define the concepts of feature coverage. Then, we describe how the test cases are generated using the Concolic testing algorithm.

4.1. Weight Feature Metric

The BN abstraction and the hidden feature weights are utilised to develop new coverage metrics that assess the quality of a test dataset in terms of reporting the coverage based on the non-uniform contribution theory. That is, the metrics focus on the semantic values of the neuron activation instead of the syntactic values of the adjusted weights, which is a very local and less decisive criterion.

Definition 4.1 (Weight-based Feature Coverage). *Given a trained DNN $\mathcal{N}$, the* weight-based feature coverage *of a* non-empty *set of inputs $X \subset \mathbb{D}_X$ is obtained via the BN abstraction $\mathcal{B}_{\mathcal{N},X}$ as:* $\mathrm{WFCov}(\mathcal{B}_{\mathcal{N},\mathcal{X}}) \overset{def}{=}$

$$\sum_{(\!|f^\sharp_{i,j}|\!) \in V_{\mathcal{N},X}} w_{(\!|f^\sharp_{i,j}|\!)} \cdot \frac{\left|\left\{ f^{\sharp k}_{i,j} \in \mathbb{F}^\sharp_{i,j} \mid \mathcal{P}_i\left(f^{\sharp k}_{i,j}\right) \geq \varepsilon \right\}\right|}{\left|\mathbb{F}^\sharp_{i,j}\right|}, \tag{3}$$

where $\sum_{(\!|f^\sharp_{i,j}|\!) \in V_{\mathcal{N},X}} w_{(\!|f^\sharp_{i,j}|\!)} = 1$.

The coverage metric in the equation above ranges over $[0, 1]$, and gives the weighted proportion of features that are adequately exercised by X. The $\mathrm{WFCov}(\mathcal{B}_{\mathcal{N},\mathcal{X}})$ is similar to the basic feature coverage in Equation 1, where the factor $1/\left|V_{\mathcal{N},X}\right|$ that acts as an *equals* weight for all features is replaced with the computed importance weight.

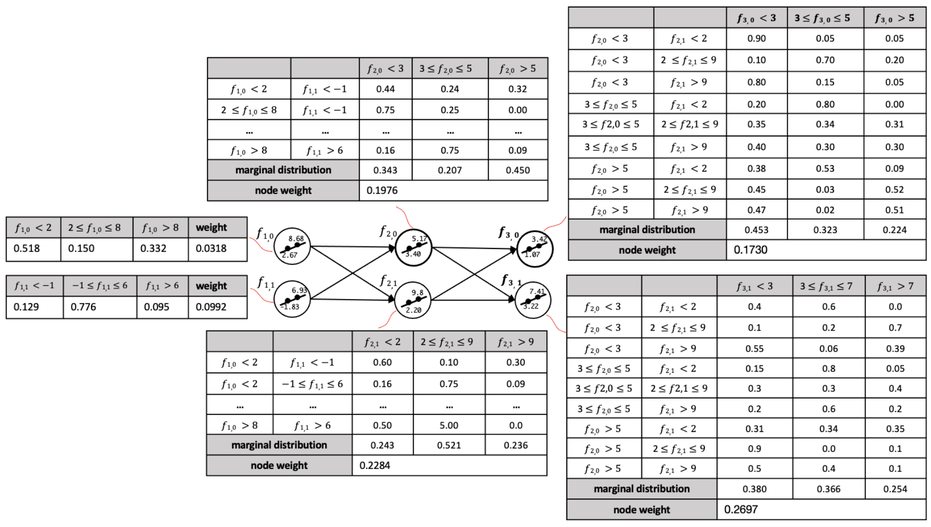
Example 2. *Consider the BN shown in Figure 2. For layer l_3, we can compute the following marginals based on the given conditional probability table for the node $f_{3,0}$ as:* $\Pr\left(f_{3,0} < 3\right) \approx 0.453$, $\Pr\left(3 \leq f_{3,0} \leq 5\right) \approx 0.323$, $\Pr\left(f_{3,0} > 5\right) \approx 0.224$. *Then the sum of intervals' marginals is multiplied with the node weight which results in* $1 \times 0.173 = 0.173$. *Assuming similar non-negligible marginal probabilities for the nodes pertained to layers l_1 and l_2, then we obtain each node coverage $3/3$ and then multiply it with its per-node weight to obtain the coverage of the test set with the* $\mathrm{WFCov}(\mathcal{B}_{\mathcal{N},\mathcal{X}}) = 1$.

4.2. Weight Feature Dependence Metric.

Further, the causal relationships exercised by a dataset X that the BN's conditional probabilities define are used to develop the following coverage metric:

Definition 4.2 (Weighted Feature Dependence Cov). *Given a trained DNN $\mathcal{N}$, the* weight-based feature dependence coverage *is obtained via the $\mathcal{B}_{\mathcal{N},X}$ as:* $\mathrm{WFdCov}\left(\mathcal{B}_{\mathcal{N},X}\right) \overset{def}{=}$

$$\sum_{(\!|f^\sharp_{i,j}|\!) \in V^+_{\mathcal{N},X}} w_{(\!|f^\sharp_{i,j}|\!)} \cdot \frac{\left|\left\{ \begin{array}{c|c} \left(f^{\sharp k}_{i,j}, F^\sharp_{i-1}\right) \in & \mathcal{CP}_i\left(f^{\sharp k}_{i,j} \middle| F^\sharp_{i-1}\right) \geq \varepsilon \\ \mathbb{F}^\sharp_{i,j} \times \mathbb{F}^\sharp_{i-1} & \vee \quad \mathcal{P}_i\left(f^{\sharp k}_{i,j}\right) < \varepsilon \end{array} \right\}\right|}{\left|\mathbb{F}^\sharp_{i,j} \times \mathbb{F}^\sharp_{i-1}\right|} \tag{4}$$



$f_{1,0} < 2$	$2 \le f_{1,0} \le 8$	$f_{1,0} > 8$	**weight**
0.518	0.150	0.332	0.0318

$f_{1,1} < -1$	$-1 \le f_{1,1} \le 6$	$f_{1,1} > 6$	**weight**
0.129	0.776	0.095	0.0992

		$f_{2,0} < 3$	$3 \le f_{2,0} \le 5$	$f_{2,0} > 5$
$f_{1,0} < 2$	$f_{1,1} < -1$	0.44	0.24	0.32
$2 \le f_{1,0} \le 8$	$f_{1,1} < -1$	0.75	0.25	0.00
...	...	...	...	...
$f_{1,0} > 8$	$f_{1,1} > 6$	0.16	0.75	0.09
marginal distribution		0.343	0.207	0.450
node weight		0.1976		

		$f_{2,1} < 2$	$2 \le f_{2,1} \le 9$	$f_{2,1} > 9$
$f_{1,0} < 2$	$f_{1,1} < -1$	0.60	0.10	0.30
$f_{1,0} < 2$	$-1 \le f_{1,1} \le 6$	0.16	0.75	0.09
...	...	...	...	...
$f_{1,0} > 8$	$f_{1,1} > 6$	0.50	5.00	0.0
marginal distribution		0.243	0.521	0.236
node weight		0.2284		

		$\boldsymbol{f_{3,0} < 3}$	$\boldsymbol{3 \le f_{3,0} \le 5}$	$\boldsymbol{f_{3,0} > 5}$
$f_{2,0} < 3$	$f_{2,1} < 2$	0.90	0.05	0.05
$f_{2,0} < 3$	$2 \le f_{2,1} \le 9$	0.10	0.70	0.20
$f_{2,0} < 3$	$f_{2,1} > 9$	0.80	0.15	0.05
$3 \le f_{2,0} \le 5$	$f_{2,1} < 2$	0.20	0.80	0.00
$3 \le f2,0 \le 5$	$2 \le f2,1 \le 9$	0.35	0.34	0.31
$3 \le f_{2,0} \le 5$	$f_{2,1} > 9$	0.40	0.30	0.30
$f_{2,0} > 5$	$f_{2,1} < 2$	0.38	0.53	0.09
$f_{2,0} > 5$	$2 \le f_{2,1} \le 9$	0.45	0.03	0.52
$f_{2,0} > 5$	$f_{2,1} > 9$	0.47	0.02	0.51
marginal distribution		0.453	0.323	0.224
node weight		0.1730		

		$f_{3,1} < 3$	$3 \le f_{3,1} \le 7$	$f_{3,1} > 7$
$f_{2,0} < 3$	$f_{2,1} < 2$	0.4	0.6	0.0
$f_{2,0} < 3$	$2 \le f_{2,1} \le 9$	0.1	0.2	0.7
$f_{2,0} < 3$	$f_{2,1} > 9$	0.55	0.06	0.39
$3 \le f_{2,0} \le 5$	$f_{2,1} < 2$	0.15	0.8	0.05
$3 \le f2,0 \le 5$	$2 \le f2,1 \le 9$	0.3	0.3	0.4
$3 \le f_{2,0} \le 5$	$f_{2,1} > 9$	0.2	0.6	0.2
$f_{2,0} > 5$	$f_{2,1} < 2$	0.31	0.34	0.35
$f_{2,0} > 5$	$2 \le f_{2,1} \le 9$	0.9	0.0	0.1
$f_{2,0} > 5$	$f_{2,1} > 9$	0.5	0.4	0.1
marginal distribution		0.380	0.366	0.254
node weight		0.2697		

Figure 2: An abstracted Bayesian network from three CNN's selected hidden layers. Two features are extracted from each layer and discretised into three intervals. The features $f_{1,0}$ and $f_{1,1}$ have marginal tables. Features $f_{3,0}$ and $f_{3,1}$ are illustrated with a complete conditional probability table, while other CPTs have the same length (m^p number of intervals to the number of parents), but are shortened in the diagram. The weight column shows per-node probability.

where $\sum_{(\!|f^\sharp_{i,j}|\!) \in V^+_{\mathcal{N},X}} w_{(\!|f^\sharp_{i,j}|\!)} = 1$.

Here, $V^+_{\mathcal{N},X}$ represents a set of nodes excluding the input layer, for which conditional probability table doesn't exist. Intuitively, in the same manner as the weighted feature coverage, we iterate over all nodes in $V^+_{\mathcal{N},X}$ and calculate a weighted coverage. For each node, we look at its CPT which lives in the space $\mathbb{F}^\sharp_{i,j} \times \mathbb{F}^\sharp_{i-1}$, and look at all of the values larger than ε. In other words, $\mathcal{CP}_i(f^{\sharp k}_{i,j} | F^\sharp_{i-1}) \geq \varepsilon$. For the metric to be independent from the previous feature coverage, the values for which marginal distribution is smaller than ε are also included, i.e. $\mathcal{P}_i(f^{\sharp k}_{i,j}) < \varepsilon$.

Example 3. *Continuing the Example 2, the weighted feature dependence coverage is considered now. The function iterates over last* 4 *out of* 6 *nodes for which CPT exists. For this example, let calculate coverage for the node* $f_{3,0}$, *with* $\varepsilon = 0.01$. *Taking a look into its CPT, there are* 26 *out of* 27 *items with probabilities larger than* 0.01. *Furthermore, all marginal probabilities are larger than* 0.01 *too, which finally means that the coverage is* $26/27$. *Now, we calculate how much will the node amount to the total weighted feature dependence coverage. Because the weights in Figure 2 are normalized to sum to* 1 *for all nodes, they have to be firstly renormalised, so that only the ones with CPT sum to* 1. *Normalisation constant is just the sum over all but first layer weights, which for our example amounts to* 0.8687. *Finally, the contribution of the node* $f_{3,0}$ *to the total coverage amounts to* $0.1730/0.8687 \cdot 26/27 = 0.1917$. *Similarly, for the node* $f_{3,1}$, *there are* 25 *out of* 27 *values with probability larger than* 0.01, *which means the node will contribute to the total coverage as* $0.2697/0.8687 \cdot 25/27 = 0.2875$. *Summing up contribution from all* 4 *nodes with CPT assuming there is one probabilities less than* 0.01 *for each* $f_{2,0}$ *and* $f_{2,1}$, *the final weighted feature dependence coverage amounts to* $0.2190 + 0.2532 + 0.1917 + 0.2875 = 0.9514\%$.

4.3. Generalised Weighted Feature Cov.

To deliver a consistent coverage measure that is based on every probability entry in the BN, the two feature metrics 3 and 4 can be combined to produce the generalised weight feature coverage. This generalised weighted feature metric gives a single, unified coverage. In the simplest approach, one can consider two coverages decoupled from each other, and simply multiply them: $\mathrm{WFCovTot} = \mathrm{WFCov} \times \mathrm{WFdCov}$. This is in most situations sufficient, however the other possibility is to average them on per-node level:

$$\mathrm{WFCovTot}\left(\mathcal{B}_{\mathcal{N},X}\right) \overset{\text{def}}{=}$$

$$\sum_{(\!|f^\sharp_{i,j}|\!) \in V_{\mathcal{N},X}} w_{(\!|f^\sharp_{i,j}|\!)} \cdot \begin{cases} \mathrm{WFCov}_{(\!|f^\sharp_{i,j}|\!)} & \text{if } i = 1\,, \\ TCov & \text{otherwise}\,. \end{cases} \tag{5}$$

where $TCov = \frac{1}{2}\left(\mathrm{WFCov}_{(\!|f^\sharp_{i,j}|\!)} + \mathrm{WFdCov}_{(\!|f^\sharp_{i,j}|\!)}\right)$.

4.4. Coverage Criteria

Weight-based Feature Coverage Criterion. A non-empty set of inputs $X \subset \mathbb{D}_X$ satisfies the *Weight-based feature coverage criterion* that is obtained via the BN abstraction $\mathcal{B}_{\mathcal{N},X}$ iff $\mathrm{WFCov}(\mathcal{B}_{\mathcal{N}},\mathrm{x}) = 1$.

Weight-based Feature-dependence Coverage Criterion. A non-empty set of inputs $X \subset \mathbb{D}_X$ satisfies the *Weight-based feature-dependence coverage criterion* that is obtained via the BN abstraction $\mathcal{B}_{\mathcal{N},X}$ iff $\mathrm{WFdCov}(\mathcal{B}_{\mathcal{N}},\mathrm{x}) = 1$.

4.5. Concolic Test Generation

The weight-based feature metrics are implemented on the DeepConcolic tool [1] and the features weights are used as criteria to direct the Concolic testing algorithm. A detailed description of the test generation procedure is provided in Algorithm 1.

For a given trained DNN $\mathcal{N}$ on a dataset X and the associated abstract BN $\mathcal{B}_{\mathcal{N},X}$, the features weights W_f are calculated for all extracted features. We assume that suitable feature extraction and discretisation have been applied on a training sample X_{train} to obtain the structure of the $\mathcal{B}_{\mathcal{N},X}$. The test generation procedure starts by randomly sampling an initial seed set of test inputs X_0 from X_{test} data set that is correctly classified by $\mathcal{N}$, and initialising the probability tables in the BN to produce $\mathcal{B}_{\mathcal{N},X_0}$. Next, the algorithm identifies the test target intervals $Tar\_invals = \{f^{\sharp}_{i,j}\}$ through analysing the non-epsilon probabilities of the marginal or conditional probability tables in $\mathcal{B}_{\mathcal{N},X_0}$. The non-epsilon is the probabilities that are less than ε and not yet met by the current set of input test cases in X_0. Thus, the Tar_invals consist a set of hidden feature interval(s) that should be elicited by the test input to be generated. The test case generation algorithm then iterates max times according to the following:
First, identifying the $t \in Tar\_invals$ with the highest importance weight in W_f, and selecting a test input $s \in X_0$ based on some heuristics, such as closeness to the targeted interval t. The implemented assumption to find a good-enough candidate input s is searching for an input $s \in X_0$ whose feature value is close to the target interval boundaries. Then, constructing an LP problem based on t and solve the optimisation objective that seeks to minimise the distance between activations of input neurons and s. This problem is formulated as:

$$\text{Minimise: } \|(n_{1,1}, \ldots, n_{1,|l_1|}) - (s_{1,1}, \ldots, s_{1,|l_1|})\|_\infty \tag{6}$$

Where $n_{1,1}, \ldots, n_{1,|l_1|}$ is the set of all input neurons.

[1]The tool is available at https://github.com/TrustAI/DeepConcolic.

Algorithm 1 Test Dataset Generation

```
Input:
$\mathcal{N} \leftarrow$ DNN under test
$X \leftarrow$ data set
$\mathcal{B}_{\mathcal{N},X_{train}} \leftarrow$ abstract BN
$W_f \leftarrow$ features sensitivity weights
Output: test inputs $X_0$, coverage
 1: $X_0 \leftarrow$ sampling initial seed test inputs from $X_{test}$
 2: $\mathcal{B}_{\mathcal{N},X_0} \leftarrow$ initialising the BN prob. tables with $X_0$
 3: $Tar\_invals \leftarrow$ intervals with prob $\leq \varepsilon$
 4: for $i = 1$ to max iterations do
 5:   $t \leftarrow Tar\_invals$ with highest weight in $W_f$
 6:   select a test input $s \in X_0$
 7:   construct an LP problem based on $t$
 8:   solve the optimisation objective:
      min $\|(n_{1,1}, \ldots, n_{1,|l_1|}) - (s_{1,1}, \ldots, s_{1,|l_1|})\|_\infty$
 9:   $s' = (n_{1,1}, \ldots, n_{1,|l_1|})$
10:   if $s'$ passes the oracle then
11:     $s' \leftarrow$ newly generated test input
12:     if $f_{\mathcal{N}}(s') = f_{\mathcal{N}}(s)$ then
13:       $X_0 \leftarrow X_0 \cup \{s'\}$
14:       update $\mathcal{B}_{\mathcal{N},X_0}$ probabilities
15:       update coverage
16:     else
17:       $s' \leftarrow$ adversarial input
18:     end if
19:   end if
20: end for
```

After solving the LP problem and extracting the newly generated test input s' from values of input neurons: $s' = (n_{1,1}, \ldots, n_{1,|l_1|})$, the algorithm will check two properties of the new input s'. Does the s' pass the oracle, i.e., is it structurally close enough to s w.r.t the L_∞ norm? If yes, then, does the s' output the same classification label of s, in other words, is $f_{\mathcal{N}}(s') == f_{\mathcal{N}}(s)$? If yes, then, the s' is considered a valid input and added to the test input $X_0 = X_0 \cup \{s'\}$. Otherwise, the s' is considered *adversarial* for $\mathcal{N}$, as s' is both deemed close enough to s from which it is derived, and it is not assigned the same classification label as s by $\mathcal{N}$. Accordingly, the probabilities in $\mathcal{B}_{\mathcal{N},X_0}$ are updated to account for the new test s' and then recalculate the coverage. The test case generation continues if the test criteria obtained via the new $\mathcal{B}_{\mathcal{N},X_0}$ is not yet satisfied.

Note that, the new test s' may not actually improve reported coverage if it is just "closer" to the target interval than s but does not hit it. The expectation is that, s' will later be selected to generate a new input s'' according to the same process, and eventually the target interval might be reached.

5. Evaluation

This section reports on the experimental analysis conducted to evaluate the performance of the suggested coverage metrics and the usability of the weight in guiding the adapted Concolic test case generation. The first set of analyses examined the quality of the existing BN-based feature metrics originated by Berthier et al. [2]. Then, the efficiency of the developed feature weights was tested and compared to the previous coverage results.The research questions that were investigated are:

1. **RQ1:** Do existing BN-based testing metrics guarantee covering a model's critical parts and directing their test generation algorithm to target the most relevant features?
2. **RQ2:** Does the proposed coverage metrics deliver a reliable testing measure in terms of reporting the coverage *prioritising* the important internal representation of the model?

5.1. Datasets and Models

Two trained CNN models have been trained for the experiments: the first one targets the Fashion-MNIST classification problem with 89.03% validation accuracy, and the second model targets the CIFAR-10 dataset with 81.00% validation accuracy. The models are reasonably sized, with more than 10 layers, including blocks of convolutional and max-pooling layers, followed by a series of dense layers. Three different layers with various functionality are chosen for the testing to fairly cover all types of layers. For the two models, the considered layers are the convolutional ReLU, 2d max pooling, and dense ReLU. Note that our proposed testing approach is applicable to any size of neural network since it is based on an abstracted model that performs a dimension reduction on any number of desired layers.

5.2. Experimental Setup

In the following experiments, the high-level criteria is used to investigate how a test dataset exercises a set of hidden features that has been learned from the training dataset and internally represented by any layer of the CNNs. Therefore, the reliance will be placed on the latent features learned by the trained CNN models. Multiple strategies for linear dimensionality reduction and discretisation of each feature component were applied to construct various BN abstraction scheme. Two linear feature extraction techniques were selected: PCA and ICA, with two to five numbers of extracted features for each of the abstracted layers. The Kernel Density Estimation (KDE) and uniform-based discretisation are considered, with varying numbers of the uniform partitions bins that are: one, three, and five. Finally, the extended Concolic testing tool is run on both DNNs models with a maximum 100 iterations per run. Each run is initialised with uniformly drawn test sets X_0 of 10 and 100 correctly classified inputs.

5.3. Results and Discussion

To analyse the testing outcomes, it is necessary to carefully select and decide how to split different categories. Since the experiment objective is to demonstrate the increase in coverage over the run time, the primary variables will be the initial coverage, the final coverage, and the time it takes to obtain the final coverage. So, there are two numerical parameters: run-time and coverage. Other parameters are categorical: initial or final; ICA or PCA; initial test sizes. Therefore, plotting the result in space of run-time vs. coverage, and have one error point representing each categorical class - initial PCA, initial ICA, final PCA and final ICA, will illustrate the desired intention. For each of those variables, the errors are calculated as following: For the run-time, standard normal error is expected, so the mean and one standard deviation are calculated. This amounts to 68% interval around the mean. For coverage, however, distribution is neither normal nor symmetric. Therefore, median and 68% interval around it is computed, equivalent to the previous case.

RQ1: Coverage Quality Analysis Using Existing Metrics. The plots in Figure 3 show the results of a standard test generation process, for two of our datasets. First and second rows show BFCov and BFxCov metrics respectively. Every column differs in the initial test size $X_0 \in \{10, 100\}$. Each individual plot shows initial and final coverage distributions (their medians and 68% regions), for PCA and ICA methods. The interpretation is that higher median line on coverage, better the median coverage and smaller the errors.

The analysed outcomes illustrate that test generation process consistently enlarges the median of the coverage, which is expected. However, the spread of a distribution stays similar, with a few exceptions. Larger number of runs could improve the precision of results, however, we believe the main reason for such a spread is that only runs with higher initial coverage managed to improve. The ones with low initial coverage were hard to improve and stayed the same. It can be observed that the constant 0.33333333333 initial coverage that appeared frequently in all testing situations, did not increase in most cases (note the minimum coverage -initial and final for all charts is 0.33).

Considering the initial test size, we can see that larger initial test size, *i.e.,*, $X_0 = 100$ consistently results in larger (sometime comparable) coverage. A larger X_0

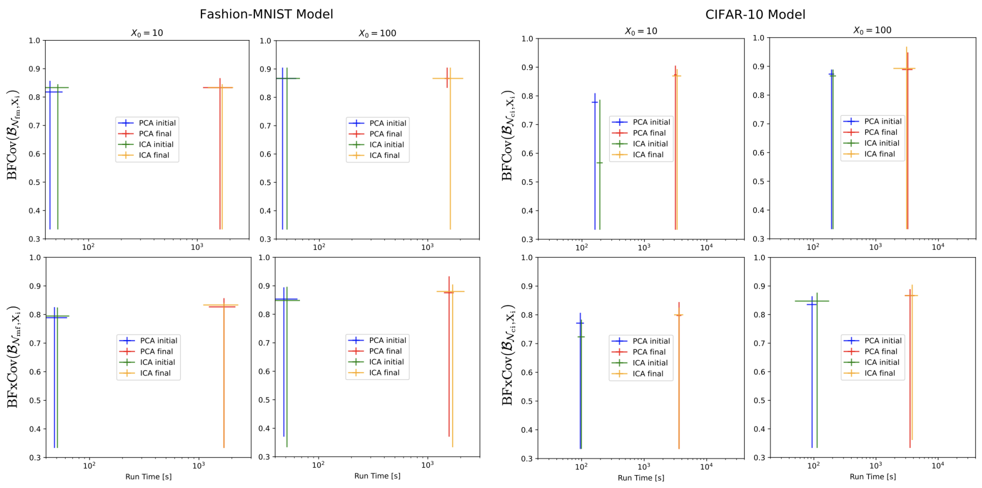


Figure 3: BN-based feature coverage plots show the overall distribution of initial and the respective final coverage of up to 100 iterations of Concolic test case generation. X-axis indicates the run time in seconds (initial and run time). The vertical lines are the coverage, and the horizontal line on the coverage is the median.

gives the synthesis algorithm more leeway to find candidates from which to derive new inputs that hit target intervals that are not exercised by any test in X_0. For the PCA and ICA, there's no apparent difference between two methods, one exception for the Fashion-MNIST, $X_0 = 100$, BFCov metric, where PCA results in much tighter distribution. As the same is not visible in the BFxCov metric, the significance of this result cannot be assessed. Both BFCov and BFxCov metric generally agree with the level of improvement during the testing. Considering runtime, the charts express that ICA method is slightly more expensive in all situations.

Giving a deeper inspection to the all-finals coverage, a query about 1.00 achieved coverage shows it only obtained twice with the bfc criterion on the CIFAR-10 dataset. Both situations occurred with a 100 initial test size using the ICA with two and three extracted features per layer and the KDE discretisation method. This implied a total of 254 combination traces out of 256 (64 per testing criteria per CNN model) did not satisfy $\mathrm{BFCov}(\mathcal{B}_{\mathcal{N},\mathrm{X}}) = 1$ neither $\mathrm{BFxCov}(\mathcal{B}_{\mathcal{N},\mathrm{X}}) = 1$, after 100 iterations. Observe the final coverage in Figure 3, with red and yellow colours, the average median final coverage is around 0.87, which mean there are 0.13 of the networks remain not tested. What if the not covered features are the vital element of the neural network? There are neither guarantees nor any information about the untested elements. This issue will be evidenced in the following experiments.

RQ2: Weight Features Coverage using Proposed Metrics. The following experiments assess whether the weight-based approach exhibits advantages in improving the BN-based feature coverage. In particular, the study examines whether the weight-based feature metrics will achieve higher coverage with less run time than the original metrics.

Figure 4 shows the results for the weighted coverage, in equivalent arrangement as the previous Figure 3. Comparing the two figures, as can be seen, the minimal starting coverage in the majority of the plots is greater than the value of 0.33, which occurred often in the previous experiment. This expected increase in initial coverage results from the fact that one coverage is being weighted and the other is not. This small growth gives a greater opportunity for the coverage to be improved during the testing. An example from the weight coverage testing experiment that gave 0.3895021093 initial coverage that increased to 0.7261016195 final coverage with 51 new generated inputs. This is evident from the preceding finding, which reported that starting testing with a higher initial coverage has a better chance of increasing. Furthermore, the initial coverage in all plots, except for the CIFAR-10 with $X_0 = 100$, are consistent with initial coverage in Figure 3. That indicate most of features with higher weights are not covered yet.

Considering the final coverage, the charts show a significant improvement in the coverage for the WFCovTot compared to the BFxCov for both datasets and PCA/ICA methods. The reason for this is that the generation process is led by the most important parts of the BN, which have larger weights. A notable observation is that the minimum final coverage increased considerably, which indicates the higher-importance intervals were covered

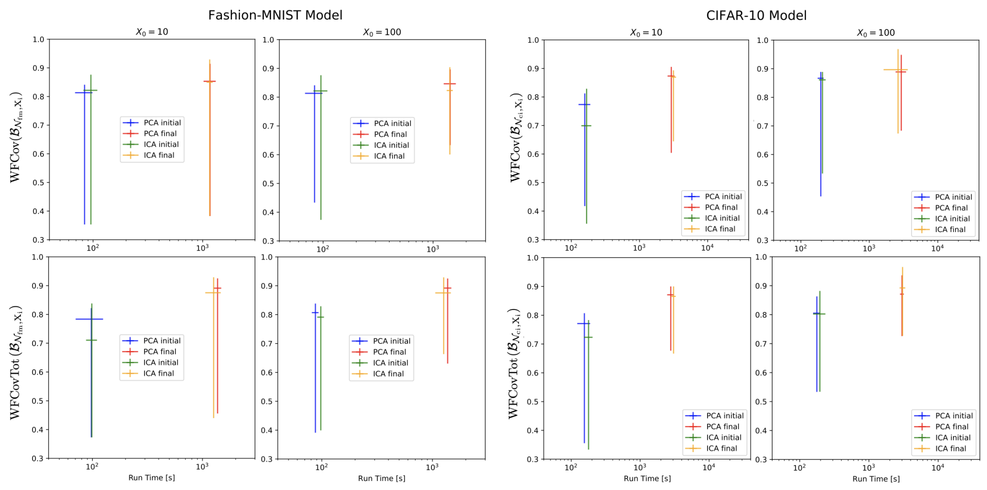


Figure 4: Weight-based feature coverage evaluation with identical caption as Figure 3.

with the new input first. This trend is the same for all coverages except for the F-MNIST model with $X_0 = 10$. Finally, considering runtime, weighted coverage takes more time for the initial computation, which spend additional time calculating feature weights. However, convergence is reached slightly faster compared to the non-weighted case. Improvement is of a few percent, thus not so significant.

5.4. Further Results

The above experiments clearly demonstrated the effectiveness of the weighted coverage compared with the basic coverage. Both metrics were able to generate new sets of inputs that achieve high coverage. The plot in Figure 5 shows the growth of the generated test set with respect to the testing iterations. Overall, between 10% to 60% of iterations produce new test cases. However, the WFCov and WFdCov enforced the higher coverage on the more relevant training dataset features. Consider one testing scenario illustrated in Table 1, with the bfc criterion, the testing algorithm was able to generate 25 test inputs, of which two hit desired intervals: the third interval of the first feature extracted from max_pooling2d_1 layer and the fourth interval of the second feature extracted from the same layer. The coverage increased properly in the same proportions. On the other hand, the wfc criterion systemically picked out the interval from Tar_invals set, and the algorithm was able to synthesise new tests for three high-weight intervals before the 100 iterations were over. Note that a full experiment document with clear coverage numbers and feature weights presented in a tabular manner will be uploaded to the arXiv database.

6. Conclusion

This paper introduced a weight-based semantic testing approach that measures how well the DNN is tested by focusing on the important features of the DNN using its abstracted Bayesian network. Investigating the high-level feature weights revealed the network's internal decision mechanism and how it processes the input data. The developed weighted feature metrics achieved higher testing coverage than the original metrics, with an emphasis on covering important learned representations. The test generation algorithm is directed to synthesise new input targeting features with higher importance scores. The conducted experiments empirically validated the applicability and effectiveness of the proposed weight metrics. This serves as a strong argument in favour of increasing the trustworthy performance of the DNN models.

References

[1] S. Yan, G. Tao, X. Liu, J. Zhai, S. Ma, L. Xu, X. Zhang, Correlations between deep neural network model coverage criteria and model quality, in: Proc. of the 28th ACM, 2020, pp. 775–787.

[2] N. Berthier, A. Alshareef, J. Sharp, S. Schewe, X. Huang, Abstraction and symbolic execution of deep neural networks with bayesian approximation of hidden features, arXiv preprint arXiv:2103.03704 (2021).

[3] A. Alshareef, N. Berthier, S. Schewe, X. Huang, Quantifying the importance of latent features in neural networks, in: CEUR Workshop Proceedings, volume 3087, 2022.

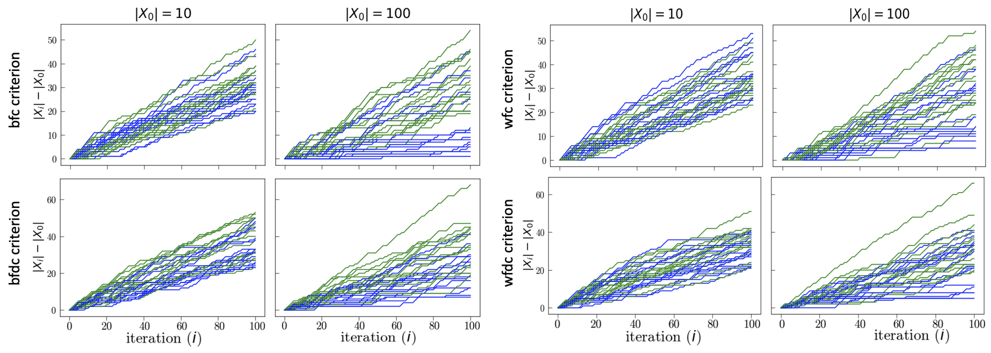


Figure 5: New produced test inputs of up to 100 iterations of test case generation targeting BN-based coverage and weight-based coverage for the Fashion-MNIST model, for two sizes of initial test sets {10, 100}. Each raw specifies the used criterion: bfc, bfdc, wfc, and wfdc. Green and blue lines respectively indicate runs with ICA and PCA feature extractions.

BN specification	criterion	init_cov	hit_interval	invl_weight	progs_cov	#gen$_{tests}$
PCA-X10-N3-U5	bfc	0.8095	l:max f:0 v:2	—	0.8254	25 + 1 adv.
			l:max f:1 v:3	—	0.8413	
	wfc	0.8409	l:max f:2 v:4	0.0237	0.8646	27 + 3 adv.
			l:act f:0 v:0	0.0159	0.88	
			l:act f:0 v:2	0.0159	0.8964	

Table 1
Improvement of testing coverage (up to 100 iterations) for one BN specification scenario (PCA-X10-N3-U5), conducted on the Fashion-MNIST model. PCA indicates the feature extraction method, X10 indicates $|X_0|$ size, N3 is the number of extracted features per DNN's layer, and U5 is the uniform discretisation with five bins. Two criteria are compared: bfc and wfc.

[4] Y. Bengio, I. Goodfellow, A. Courville, Deep learning, volume 1, MIT press Cambridge, USA, 2017.

[5] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: Automated whitebox testing of deep learning systems, in: proc. of the 26th SOSP, 2017, pp. 1–18.

[6] Y. Tian, K. Pei, S. Jana, B. Ray, Deeptest: Automated testing of deep-neural-network-driven autonomous cars, in: Proc. of the 40th ICSE, 2018, pp. 303–314.

[7] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, J. Sun, Dlfuzz: Differential fuzzing testing of deep learning systems, in: Proc. of the 2018 26th ACM Joint Meeting on ESE Conference and the FSE, 2018, pp. 739–743.

[8] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, R. Ashmore, Testing deep neural networks, arXiv preprint arXiv:1803.04792 (2018).

[9] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, et al., Deepgauge: Multi-granularity testing criteria for deep learning systems, in: Proceedings of the 33rd ACM/IEEE ICASE, 2018, pp. 120–131.

[10] C.-H. Cheng, C.-H. Huang, H. Yasuoka, Quantitative projection coverage for testing ml-enabled autonomous systems, in: Automated Technology for Verification and Analysis: 16th IS, ATVA 2018, Los Angeles, USA, October 7-10, 2018, Proceedings 16, Springer, 2018, pp. 126–142.

[11] M. Wicker, X. Huang, M. Kwiatkowska, Feature-guided black-box safety testing of deep neural networks, in: Tools and Algorithms for the Construction and Analysis of Systems: 24th IC, TACAS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I 24, Springer, 2018, pp. 408–426.

[12] J. Kim, R. Feldt, S. Yoo, Guiding deep learning system testing using surprise adequacy, in: 2019 IEEE/ACM 41st ICSE, IEEE, 2019, pp. 1039–1049.

[13] Z. Li, X. Ma, C. Xu, C. Cao, Structural coverage criteria for neural networks could be misleading, in: 2019 IEEE/ACM 41st ICSE: New Ideas and Emerging Results (ICSE-NIER), IEEE, 2019, pp. 89–92.

[14] Y. Dong, P. Zhang, J. Wang, S. Liu, J. Sun, J. Hao, X. Wang, L. Wang, J. S. Dong, D. Ting, There is limited correlation between coverage and robustness for deep neural networks, arXiv preprint arXiv:1911.05904 (2019).

[15] S. Gerasimou, H. F. Eniser, A. Sen, A. Cakan, Importance-driven deep learning system testing, in: Proceedings of the ACM/IEEE 42nd ICSE, 2020, pp. 702–713.

[16] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, D. Kroening, Concolic testing for deep neural networks, in: Proceedings of the 33rd ACM/IEEE ICASE, 2018, pp. 109–119.