A Survey on Test Input Selection and Prioritization for Deep Neural Networks

Shengrong Wang[1], Dongcheng Li[2,*], Hui Li[1], Man Zhao[1], and W. Eric Wong[3]
[1]School of Computer Science, China University of Geosciences, Wuhan, China
[2] Department of Computer Science, California State Polytechnic University - Humboldt, Arcata, USA
[3]Department of Computer Science, University of Texas at Dallas, Richardson, USA
wsr@cug.edu.cn, dl313@humboldt.edu, huili@vip.sina.com, zhaoman@cug.edu.cn, ewong@utdallas.edu
*corresponding author

Abstract—With the breakthrough advancements of deep neural network technology in applications such as image processing, autonomous driving, and speech recognition, the testing of deep neural network models becomes crucial to ensure their performance and reliability. The selection of test inputs is a critical step in the testing process. Prioritizing test inputs and selecting the most impactful ones can help improve testing efficiency to identify potential problems and deficiencies in the model as early as possible, given the large size of the test dataset and the high cost of annotation. In order to gain a deeper understanding of the research progress in the field of test input selection for deep neural networks, this paper conducts a survey of academic papers in the field over the past years. A systematic review of existing research outcomes is presented, focusing on the criteria, methods, and priority ranking for the selection of test inputs in deep neural networks. Additionally, the paper offers insights into future challenges for test input selection in deep neural networks.

Keywords-deep neural networks testing; test inputs; test input selection; selection criteria; test input prioritization

1. INTRODUCTION

Deep neural networks (DNNs) , commonly employed models in deep learning systems, have played a crucial role in numerous application domains [1][2], including facial recognition [3], autonomous driving [4], image processing [5], and software engineering [6][7][8][9]. Despite the great success of Deep Neural Network (DNN) models in many application areas, existing research indicates that DNN models still exhibit numerous potential issues. These issues often lead to unpredictable errors in deep learning systems. Therefore, detecting erroneous behavior in Deep Neural Networks and evaluating model performance have become major bottlenecks in the development of deep neural network models. Researchers have proposed many testing techniques [10][11][12][13][14][15][16][17][18][19] to address this challenge.

There are many key differences between deep neural networks and traditional software. Traditional software is typically designed and coded based on explicitly defined rules and logic, allowing developers to understand the program's behavior by tracing the execution paths of the code. In contrast, deep neural networks are trained on large-scale datasets, and their behavior is influenced by the training data. The internal workings of deep neural networks are often challenging to interpret. Therefore, testing deep neural networks is often regarded as black-box testing, involving considerations of data diversity, model robustness, and interpretability as specific challenges. Developers frequently utilize extensive datasets to retrain DNN models, correcting their incorrect behavior and improving overall performance.

Testing deep neural networks is an effective method for ensuring their quality [20]. However, due to the variety and size of test inputs, as well as the high cost of labeling test inputs, the testing process still has serious efficiency issues. It took more than 49,000 people and 9 years to label the ImageNet [21] dataset containing millions of images. Therefore, in situations with limited annotation budgets, identifying and selecting the most representative and meaningful test inputs from large-scale unlabeled datasets is crucial for enhancing the effectiveness and testing efficiency of deep neural network models [22].

To gain insights into the issues of DNN test input selection and prioritization, this study compiles relevant papers in this domain from 2016 to 2023 for review and analysis. The paper search primarily utilized keywords such as 'Deep Neural Networks Testing,' 'Test Input Selection,' 'Test Input Prioritization,' and 'Test Input Metrics.' Influential search engines including Google Scholar, Web of Science, Ei Village, CNKI, and others; both domestic and international were employed for retrieval. By briefly reviewing the titles and abstracts of the papers, an initial screening was conducted based on predetermined criteria to eliminate papers irrelevant to the research question. Subsequently, an in-depth investigation into associated works, citations, and references in papers identified as highly pertinent was carried out to acquire additional literature with higher relevance. This study ultimately identified 85 papers relevant to the research question. Figure 1 provides a statistical overview of the publication distribution of these papers across different years. This study consolidates recent research findings in the field of test input selection and prioritization and classifies and summarizes their main ideas and research approaches. The article also identifies the challenges and problems faced in this research area of test input selection. The remainder of this article is organized in the following manner: Section 2 describes the criteria for test input selection; Section 3 briefly discusses two existing test input selection methods; Section 4 organizes the various approaches to test input prioritization;

2835-2823/24/$31.00 ©2024 IEEE
DOI 10.1109/ISSSR61934.2024.00035

Section 5 discusses the current challenges of the technique; finally, Section 6 and Section 7 present the threats and conclusions of this study, respectively.

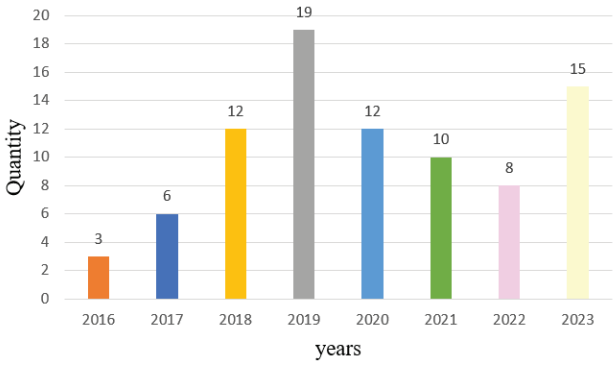

Figure 1. The publication distribution of the literature across different years

2. Criteria for Test Input Selection

In the testing of deep neural networks, the criteria for selecting test inputs are crucial as they directly impact the evaluation and improvement of model performance. In order to achieve effective testing of deep neural networks, researchers have proposed a variety of test input selection criteria for deep neural networks from different perspectives, which can play a role in test input selection and prioritization.

2.1. Model Coverage

Deep neural networks use a deep architecture in neural networks consisting of many layers, each containing a large number of neurons [23]. The DNN model coverage criterion evaluates the adequacy of the test inputs for neural network testing by calculating their coverage of the model. Test coverage criteria can be categorized into structural and non-structural coverage based on whether or not the coverage of structural elements is considered during testing.

- Structural Coverage

Structural coverage considers the structural elements of coverage, mainly referring to the coverage of various neurons [24].

Researchers from Columbia University in the USA, including Pei et al. [25], designed and proposed the first white-box framework, DeepXplore, for systematic testing of deep neural networks. They initially introduced the concept of Neuron Coverage (NC), defined as the ratio of the number of neurons activated by the test inputs to the total number of neurons in the DNN. This measure is used to assess how much of the DNN's logic is covered and executed by a set of test inputs. Building on the research of Pei et al., Ma et al. [26] proposed a set of multi-granularity testing criteria for DNN testing, DeepGauge. This criterion measures, for a given set of test inputs, the extent to which it covers the main functions and boundary conditions of the model, and introduces five coverage criteria including k-multi-section Neuron Coverage, neuron boundary coverage, top-k neuron coverage, strong neuron activation coverage, and top-k Neuron Patterns, at the neuron level and the layer level, respectively [27]. Tian et al. [28] designed and implemented DeepInspect, a white-box testing tool for DNNs, to automatically detect confusions and biases in DNN-driven image classification applications. DeepInspect defines metrics such as Neuron Activation Probability Vector Distance (NAPVD) and Average Bias, based on the paths of activated neurons in the model.

Inspired by the MC/DC coverage criteria in the field of passive software testing, Sun et al. [29] proposed four DNN test input metrics (symbol-symbol coverage, distance-symbol coverage, symbol-value coverage, and distance-value coverage) based on the variation of neuron activation values between neighboring layers; they studied how this variation affects the outputs of the neurons of the latter layer and implemented them in the DeepCover tool. Based on the interaction of test inputs, Ma et al. [30] continued their exploration in the field of deep neural network testing and proposed a deep neural network coverage criterion based on combinatorial testing, which discretizes the output values of neurons in a deep neural network, takes into account the neuron activation states and configurations of the DNN, and defines the neuron-activation configurations as well as two coverage criteria: the t-way combinatorial sparse coverage and the t-way combinatorial dense coverage.

Sekhon and Fleming [31] propose that an ideal coverage criterion must ensure completeness, meaning that all parts of the internal decision logic of a DNN have been tested by at least one input. Therefore, they introduce a coverage criterion consisting of two factors—the conditional impact of each neuron on the values of the next-layer neurons and the combination of values in one layer. This criterion takes into account the conditional decision relationships between adjacent layers and the combination of neuron values within the same layer. In addition, the impact of different combinations of neurons in the preceding layers on the neurons in the subsequent layers of a DNN model varies [32]. Based on the idea of path-oriented testing, Wang et al. [33] proposed DeepPath, a series of path-driven testing methods for DNN testing. DeepPath treats a single neuron in the model as a node and neuronal connections between different layers as paths. They proposed three path coverage metrics, l-SAP, l-OAP and l-FSP to calculate the coverage of a deep neural network. Zhou et al. [34] proposed DeepCon, a novel contribution coverage. DeepCon defines the term contribution as the combination of a neuron's output and the weights of the connections it emits and uses contribution coverage to measure the test adequacy of a DNN. A new coverage metric, NLC (NeuraL coverage), was proposed by Yuan et al [35]. NLC treats a single DNN layer as the basic unit of computation and accurately describes how DNNs understand inputs by approximating the distribution. Ji et al. [36] proposed the first causality-aware DNN coverage criterion, CC (Causal Coverage), which evaluates test inputs by quantifying the extent to which the input provides new causality to the test DNN.

- Non-Structural Coverage

In practical testing, the most classical non-structured coverage is surprising coverage, which treats the difference in DNN behavior between the test input and the training data as the surprise adequacy (SA) of the test inputs [24].

Kim et al. [19] proposed SADL, a fine-grained test adequacy framework for calculating the "surprise value" of test inputs relative to training inputs, for DNN models. They introduced two metrics: surprise adequacy (SA) and surprise coverage (SC). SA quantifies the relative surprise of each input with respect to the training data, and SC is used to measure the coverage of the range of discrete input surprises rather than the count of neurons with specific activation features. Hu et al. [37] utilized the prediction loss and coverage criteria of DNN as the criteria for test input selection. Through this approach, the generated test inputs not only induce misclassifications in the neural network but also achieve high neuron coverage. Du et al. [38] proposed two test coverage criteria for RNN networks based on abstract state-transition models, state-level and transition-level, to capture their dynamic state-transition behaviors based on the proposed DeepCruiser, an automated test framework capable of systematically generating large-scale test inputs. Table 1 summarizes the information about the existing influential model coverage criteria.

Table 1. Summary and comparison of model coverage criteria

Coverage Criteria	Framework/Tool	Advantages	Disadvantages	Year	Open Source
Neuronal Coverage [25]	DeepXplore	Simple calculation.	broad granularity	2017	Yes
k-multi-section Neuron Coverage, Neuron Boundary Coverage, Strong Neuron Activation Coverage, Top-k Neuron Coverage, Top-k Neuron Patterns [26]	DeepGauge	Wider coverage.	Calculation complexity	2018	No
NAPVD, Average Bias [27]	DeepInspect	Automatic detection of confusion and deviation errors.	Calculation complexity	2019	Yes
Symbol-Symbol Coverage, Distance-Symbol Coverage, Symbol-Value Coverage, Distance-Value Coverage [29]	DeepCover	A more complete picture of the state inside the neural network.	Lower generalizability	2018	No
t-way Combination Sparse Coverage, t-way Combination Dense Coverage, (p, t)-Completeness [30]	DeepCT	The combinatorial relationship of neuron output values is considered.	High calculation cost	2018	Yes
l-length Strong Activated Path Coverage, l-length Output Activated Path Coverage, l-length Full State Path Coverage [33]	DeepPath	Comprehensive consideration of the connections between neurons.	Complex implementation	2018	Yes
Contribution Coverage [34]	DeepCon	Comprehensive coverage	Complexity of implementation	2021	No
Neural Coverage [35]	/	NLC-guided input mutations yield better and more diverse error behaviors.	Large time cost	2022	Yes
Causal Coverage [36]	/	Practical and efficient	Calculation complexity	2023	No
Surprise Coverage [19]	SADL	Capable of accurately capturing unexpected values entered.	Dependent datasets	2019	Yes
State Level Coverage, Transition Level Coverage [38]	DeepCruiser	Enables refined and simplified testing	Calculation complexity	2018	No

2.2. Model Robustness

In practice, DNN models may be maliciously attacked by users thus making wrong judgments. Robustness is a key property of DNN models in the face of diverse and changing data, and researchers have proposed a variety of robustness metrics as test input selection criteria for measuring the degree to which a DNN performs stably under different inputs. Some related works [39][40] provide a more detailed analysis of the performance stability of deep neural networks.

The model's accurate prediction of test inputs is one of the crucial markers when evaluating model performance. The metric of accurate prediction, i.e., the percentage of test samples that the model successfully classifies correctly, is widely defined as prediction accuracy. It directly reflects the model's classification ability and performance and is therefore considered a key factor in assessing model robustness. Hu et al. [41] proposed a new technique called Aries to select representative input data, and they argued that the model should have similar prediction accuracy on data with similar distances from the decision boundary. In addition to the model's predictive accuracy, Ling et al. [42] argue that confidence in predicting incorrect samples can further assess the robustness of the model. In their study, they proposed two different criteria. Adversarial class average confidence is defined as the average predicted confidence for incorrect

classes, while true class average confidence is obtained by averaging the confidence in true classes. This further assesses the extent to which malicious attacks deviate from real samples. In practical decision systems, classification models not only strive to provide accurate predictions but also need to indicate the likelihood of the predictions being incorrect. Guo et al. [43] employed Calibration to assess the consistency between model predicted confidence and actual accuracy. They plotted a confidence-accuracy curve, and the closer it is to the perfect diagonal line, the more reliable the model's predictions are.

Zhou and Patel [44], taking into account the security implications of adversarial vulnerabilities on deep neural network models, introduced the concept of "difficulty." They evaluated the model's robustness score by assessing the difference in accuracy between clean and perturbed samples within a certain perturbation range, thereby gauging the model's robustness. Mangal et al. [45] pointed out that most existing definitions of robustness primarily focus on the worst-input scenario of adversarial inputs. They proposed a probability-based robustness metric: Probabilistic Robustness. In practical non-adversarial scenarios, Probabilistic Robustness is a more efficient, rational, and computationally cost-effective metric. It is defined as the probability that the neural network's output remains within a certain range when the difference in inputs is constrained to a small range, and this probability should be greater than or equal to a specific threshold of $1-\varepsilon$. In order to more accurately measure the robustness of DNN models, Weng et al. [46] provided a theoretical foundation for transforming robustness analysis into a local Lipschitz constant estimation problem. They introduced a novel robustness metric called CLEVER.

Katz et al. [47] from Stanford University provided an explanation for the concept of adversarial robustness, which refers to a model's ability to accurately classify samples generated through small perturbations. Based on this, many scholars have designed a series of DNN model robustness metrics based on the definition of adversarial robustness, which, unlike CLEVER, have lower computational complexity and are easier to apply in practice. In adversarial samples, even a small perturbation can lead to incorrect labeling. Bastani et al. [48] proposed two metrics to measure DNN robustness. The first metric assesses the frequency of adversarial samples, while the second metric quantifies the severity of these adversarial samples. Carlini et al. [49] pointed out significant challenges in evaluating defense methods against adversarial samples. They provided a practical guide on how to assess the robustness of deep neural network models.

Cheng et al. [50] argued that the activation value of a single neuron has a weak correlation with the overall output of the neural network. Therefore, they proposed eight measurement criteria, covering four aspects: robustness, interpretability, completeness, and correctness. However, experimental validation on mainstream datasets was not conducted. Chen et al. [51], through studying the distribution of neuron outputs in DNN models, discovered that the behavioral patterns of neurons vary for different types of DNN inputs. Therefore, they extracted neuron behavioral patterns of DNNs under different adversarial attack techniques as criteria for test input selection.

2.3. Model Uncertainty

In DNN testing, the uncertainty of a model can be expressed by uncertainty quantification metrics such as entropy, confidence, etc., which are commonly used to assess the uncertainty of a model's predictions for specific inputs [52]. The greater the uncertainty of the model with respect to candidate inputs, the more likely the inputs are to trigger erroneous behavior, and thus the performance of the model under uncertainty can be better probed by selecting test inputs that introduce higher levels of uncertainty in the selection of test inputs.

Information entropy is an important measure of system uncertainty, and the uncertainty of a random variable is positively correlated with its information entropy, i.e., the higher the information entropy, the greater the uncertainty of the random variable. In deep learning, information entropy is used to measure the degree of uncertainty in model output prediction. Gal and Ghahramani [53] proposed a Dropout-based Bayesian deep neural network to estimate the model's prediction uncertainty by performing multiple sampling predictions with Dropout and aggregating the predictions. In addition, Kendall and Gal [54] proposed an uncertainty measure that incorporates information entropy and model output entropy for uncertainty estimation in regression problems.

Ma et al. [55] suggest performing test input selection based on the criterion of model uncertainty, where model uncertainty can guide the selection of information input data. Van et al. [56] proposed a method called deterministic uncertainty quantization, which is a simple way to obtain uncertainty in a single forward pass using deep neural networks. Bao et al. [57] proposed a lightweight k-NN prediction smoothing-based test input selection criterion for deep neural networks to improve the effectiveness of existing simple test input selection methods, which takes into account not only the uncertainty of the DNN model on the test inputs themselves, but also the uncertainty of the model on its neighbors. Aghababaeyan et al. [58] proposed DeepGD, a black-box multi-targeted test selection method for deep neural network models. DeepGD not only selects test inputs with high uncertainty scores to trigger as many error-predicting inputs as possible, but also maximizes the probability of revealing obvious errors in a DNN model by selecting different error-predicting inputs.

2.4. Test Input Diversity

The criterion of diversity of test inputs for deep neural networks is used to measure the diversity of the test dataset used in the testing process of deep neural networks. By evaluating the diversity of the test input set, it can be ensured

that the selected test input set does not suffer from insufficient sample coverage, imbalance of sample classes, or similarity of sample data, which is crucial for the testing of deep neural networks.

In practical applications, geometric diversity is widely employed in the field of test input selection. This criterion measures the extent of coverage of the sample space by calculating the Euclidean distance between samples in the test input set. This method assists researchers in quickly assessing the diversity of the test input set. Normalized compression distance is a similarity measure based on Kolmogorov complexity and information distance, which helps to compare data of different dimensions and scales and transform them into relatively consistent metrics to compare the similarity between two objects. In general, the more diverse the test input set, the larger the value of the normalized compression distance. The standard deviation is a statistic used to measure the breadth of distribution or the degree of fluctuation in a data set. In measuring the diversity of a test input set, the standard deviation can be used to indicate the degree of variation among the samples in the input set. Calculating the standard deviation can help the researcher understand the distribution among the samples in the test input set and thus assess the diversity of the input set. As with standardized compressed distance, a larger value of standard deviation represents a more diverse input set.

Chen et al. [59] proposed the PACE technique, which selects test inputs from two parallel choices, including groups and minority spaces, and combines all the small sets of test inputs selected, allowing for accurate estimation of the accuracy of the entire test set. Considering the necessity of test input sets in deep neural network model testing, Mani et al. [60] proposed four metrics to measure the goodness of a test input set based on the coverage of the data points in the model's feature space, where the equivalence distinction is used to measure the distribution of the test inputs among the classes. To avoid bias in testing models that are subsets of any class, an ideal test input set should contain inputs from each class and ensure that the inputs are evenly distributed across the classes. Center-of-mass localization [60] has been proposed to measure the number of test samples located in the center-of-mass region of the class cluster diffusion. Ideally, the test inputs should be uniformly distributed in the feature space of the model. The center of mass region of a class is the normalized Euclidean distance of all points belonging to a class obtained by calculating the average of all eigenvectors of points belonging to that class and using a radius threshold r to classify whether a test point is in the center of mass region or not. In addition, they proposed a boundary condition [60] to measure the percentage of test inputs that lie near the boundary for every other class relative to the trained class cluster. The region near the boundary is the most chaotic region for the classifier, so testing in this region will provide a robust assessment of the model. Ideally, a maximum number of test points with good distribution near the boundary is required. Based on the extension of boundary conditions, Mani et al. [60] proposed pairwise boundary conditions for measuring the boundary conditions of each pair of classes to check that the boundary conditions of all pairs of classes in the dataset are tested equally.

Hao et al. [61] proposed MOTS, a multi-objective optimization-based test input selection method, for selecting a more efficient subset of test inputs. MOTS employs a multi-objective optimization algorithm, NSGA-II, which also takes into account the diversity and uncertainty of test inputs. Gerasimou et al. [62] proposed DeepImportance as a systematic approach to testing accompanied by a test adequacy criterion called Importance-Driven (IDC) to measure the semantic diversity of the test input set. Zhu et al. [63] proposed a cluster-based surprise adequacy as a metric for test input generation, aiming to improve the diversity of test inputs.

To summarize, the existing test input selection criteria provide multiple perspectives to assess the quality of test inputs, and researchers can select a test input set that is more suitable for a specific DNN model or test requirement based on these guidelines to discover as many potential problems of the model as possible and greatly improve the testing efficiency.

3. Test Input Selection Methods

Test input selection is designed to filter a representative and challenging set of test inputs from a large and complex input space to ensure a high degree of stability and confidence in the model for real-world applications. In deep learning, test input selection addresses the practical question of which subset of unlabeled data should be labeled to detect errors in deep neural network models [55]. When considering test input selection methods for deep neural networks, it is common in existing research to categorize the methods into two approaches, one focusing on streamlining the test inputs and the other focusing on prioritizing the test inputs.

3.1. Test Input Reduction

Test input streamlining aims to select a small subset from a large number of original inputs to maintain the representativeness and validity of the input set, while the annotator only needs to label the subset of test inputs, which can effectively reduce the complexity of the test and save the cost of annotation and time, but the method needs to discard part of the test inputs. Currently, the main test input refinement methods mainly include random selection methods, cluster analysis methods, and so on. Random selection is a basic method that randomly selects test inputs from the input space. Cluster analysis methods are widely used in test input streamlining. With cluster analysis, similar test inputs are grouped into clusters, and representative samples can subsequently be selected from each cluster, thus reducing the number of test inputs while maintaining diversity.

3.2. Test Input Prioritization

For successful test input selection, test input prioritization is used; in this technique, each test input is assigned a priority. Prioritization is set based on a number of model measurements, rather than a confusing order, and test inputs with a high priority are more valuable for testing DNN models. The test input prioritization technique improves testing efficiency by ensuring that high priority test inputs are prioritized to find as many potential problems with the model as possible. This article focuses on a systematic summary of DNN test input prioritization selection methods for deep learning systems in Section 4.

4. Test Input Prioritization

Test input prioritization is the process of rearranging the order of execution of test inputs in a test set based on pre-determined criteria so that higher priority test inputs are executed earlier in the test execution process than lower priority test inputs. This chapter describes the DNN test input prioritization method for deep learning systems. Its workflow diagram is shown in Figure 2. Firstly, the DNN model and test input set are entered, and then the test input prioritization assignment is performed. Table 2 summarizes the related work, and the test inputs are sorted in descending order according to their priority.

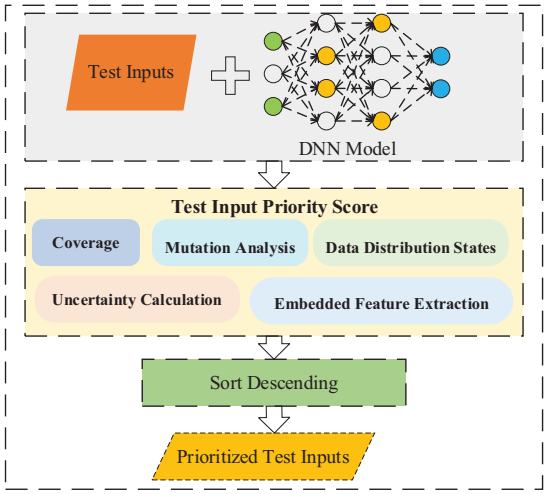

Figure 2. Test input prioritization workflow

4.1. Coverage

Coverage-based test input prioritization method is a basic and commonly used method in deep neural network testing. The main goal is to find input sample inputs that can reach more neurons in the test set, thus covering as much of the entire deep neural network as possible and testing the overall performance or potential problems of the model. The main idea behind the coverage metric is that high coverage of the DNN architecture is a useful metric for testing the validity of inputs [55].

Deep neural network neuron coverage is widely used as a prioritization metric in coverage-based approaches. Ma [26] and others proposed several neuron coverage criteria in their study, including k-multi-region neuron coverage, neuron boundary coverage, top-k neuron coverage, and strong neuron activation coverage. Guo et al. [64] proposed MOON, a white-box test input selection method based on multi-objective optimization, which focuses on the coverage of neurons and uses the neuron spectrum (neuron spectrum) to locate neurons that may lead to incorrect decisions in deep neural network models. Some researchers have considered that neurons in neural networks are not independent but need to be considered in a combinatorial context. Sun et al. [65] proposed the first combinatorial coverage test for deep neural networks, an approach that takes into account the interactions of neurons during activation. Li et al. [66] proposed an effective test for variance reduction via conditional reflection as a generalization of structural coverage and used a distributional approximation technique based on cross-entropy minimization for deep neural networks.

4.2. Uncertainty Calculation

Prioritization of test inputs using uncertainty as a basis is a strategy for prioritizing test inputs based on their uncertainty metrics. This uncertainty can originate from the uncertainty of the test inputs themselves, or from the uncertainty of the effect that the test inputs have on the behavior of the model. The core of this approach is that we need to give higher priority to those inputs that have higher uncertainty.

Back in 2019, Byun et al. [67] investigated this issue. They evaluated the validity of three such semantic metrics measuring prioritization (including confidence, uncertainty, and unexpected values) and compared their error-revealing ability and retraining effectiveness. Feng et al. [16] proposed DeepGini, a deep neural network-based technique for prioritizing test inputs from a statistical perspective, which is based on statistical theory and transforms the problem of calculating the probability of misclassification into calculating the ensemble index. In contrast to traditional white-box testing methods, DeepGini does not need to record a large amount of information about neurons in order to compute the coverage rate and instead relies on the output variables of the deep neural network in order to determine the prioritization of test inputs. Based on [16], in 2022, Weiss and Tonella [68] replicated the study of Feng et al. and found that DeepGini outperforms various types of SURPRISE and neuronal coverage metrics in terms of execution time, test prioritization, and active learning efficiency. Wang et al. [69] proposed a method based on crossing-layer dissection, Dissector, which distinguishes between external and internal inputs by collecting evidence of prediction uniqueness and works in a model-aware and lightweight manner. Pan et al. [70] designed a test input prioritization system based on Gini impurity, where the system models a series of related operations as a directed acyclic graph. Measurements are calculated based on the Gini impurity formula and used as a basis for comparison when prioritizing test inputs.

Sun et al. [71] proposed Robust Test Selection (RTS), a robust test selection method for DNNs, which divides unlabeled test

inputs into noisy, successful, and suspicious sets. RTS then assigns different selection priorities to each of these sets and ultimately prioritizes test inputs in each of the separated sets using a test metric based on a probabilistic hierarchy matrix. In practical applications, different inputs have different noise sensitivities. Therefore, Zhang et al. [72] proposed a test input prioritization technique based on noise sensitivity analysis to select inputs based on their noise sensitivity.

4.3. Embedded Feature Extraction

Prioritization of test inputs can be based on embedded feature extraction, which involves converting the test inputs into vector or embedded form, calculating the similarity or distance between them in some kind of metric space, and then prioritizing the test inputs based on the values obtained.

Zhang et al. [73] proposed a test input prioritization method for DNN classifiers that assigns high priority to those inputs that may lead to misclassification. Prioritization is calculated based on the activation patterns of neurons obtained from the training set and activated neurons collected from certain inputs. Related studies have also been done by Yan et al. [74]. Tao et al. [75] proposed a fault localization-based prioritization method TPFL for deep neural network test inputs based on the idea of fault localization technique based on spectrum. TPFL firstly performs dynamic spectral analysis of each neuron in the DNN and then uses neuron spectra to identify suspicious neurons that cause the DNN to make an incorrect decision; finally, based on the test inputs that make the suspicious neuron fully active, TPFL concludes that this may be a revealing incorrect input, and therefore the input should have a higher priority. Weng et al. [76] proposed DeepView, a novel instance-level test prioritization tool that aims to improve model performance by calculating the ability to localize and classify object detection instances to prioritize instances in the dataset that may lead to model errors.

Chen et al. [77] proposed the concept of an activation graph from the perspective of neuron spatial relations and designed an activation graph-based test input prioritization method, ActGraph, by extracting the higher-order node features of the activation graph for prioritization, which explains the differences between the test inputs in order to solve the scenario-constrained problem. Li et al. [78] proposed a new test input prioritization technique, TestRank, which ranks unlabeled test data based on the likelihood of failure. Unlike traditional methods, TestRank exploits the intrinsic and contextual properties of unlabeled test data when prioritizing them. In order to overcome the limitations of existing test input prioritization methods in the three aspects of certifiability, validity and generality, Zheng et al. [79] proposed CertPri, a test input prioritization technique designed based on the mobile cost perspective of test inputs in the DNN feature space.

4.4. Mutation Analysis

The prioritization of test inputs can also be based on the degree of mutation applied to the test inputs. This method determines the priority of a test input by measuring its difference or similarity to other test inputs. In the field of traditional software testing input prioritization, researchers have proposed mutation-based test input prioritization methods [80][81][82]. In the testing of deep neural networks, the purpose of mutation is to apply mutations to multiple models with different structures, enriching the features of the models. This results in obtaining diverse predictions from multiple models for the same test input, effectively evaluating the test input [55].

Prioritization of test inputs can also be done based on the degree of variability of the test inputs, which is prioritized by measuring the degree of difference or similarity between the test inputs and other test inputs. Typically in practice, test inputs with a high degree of variability are given a higher priority because they are more likely to reveal errors or instabilities in the model. Wang et al. [10] proposed a deep neural network (DNN) test input prioritization method called PRIMA, based on intelligent mutation analysis. PRIMA obtains multiple mutation results from a series of designed models and input mutation rules for each test input. It further incorporates machine learning for sorting, intelligently combining these mutation results to achieve effective test input prioritization.

In general, DNN test inputs are independent of each other, while GNN test inputs are usually represented as graphs with complex relationships between each test. Dang et al. [83] introduced a test input prioritization technique for Graph Neural Networks (GNN) named GraphPrior. GraphPrior generates mutation models and evaluates test inputs based on two methods: kill-based and feature-based, utilizing mutation results. Finally, it prioritizes all test inputs according to their scores. Wei et al. [84] proposed an efficient and effective test input prioritization technique, EffiMAP, which incorporates predictive mutation analysis capabilities. It predicts whether a model mutation is killed by a test input based on information extracted from the test input's execution trace. Wang et al. [85] proposed a method for runtime detection of adversarial samples. The core idea of this method is to apply random mutations on a DNN. When given a test input to the deep neural network, it determines whether the input could be an adversarial example.

4.5. Data Distribution States

The prioritization method of test inputs can be determined based on the state of data distribution; the basic idea is to determine the importance and coverage of each test input with respect to the whole input set by analyzing the data distribution of the test inputs in detail. The researchers can assign appropriate priorities to the test inputs based on their distribution in the input set.

Gao et al. [86] proposed measuring the behavioral diversity of DNN test inputs based on the differences between model outputs. The objective is to select a subset from a large pool of unlabeled datasets. Shen et al. [87] proposed a technique that clusters test inputs into boundary regions of multiple

boundaries of a deep learning model and assigns priority to uniformly select inputs from all boundary regions. Al-Qadasi et al. [88] proposed the DeepAbstraction technique, which prioritizes test inputs more likely to expose errors across the entire unlabeled test dataset.

The existing methods in current research often employ simple statistical techniques or are based on kernel density estimation. However, these methods may struggle to capture the complex characteristics and nonlinear correlations within data distributions. Therefore, future research could consider utilizing more sophisticated probabilistic models.

Table 2. Summary and comparison of test input priority sorting methods

	Priority Criteria	Data Requirements	Advantages	Disadvantages	Applicable Scenarios
Coverage	Coverage of different parts of the DNN model by test inputs.	Detailed neural network structure is required.	Comprehensive consideration of different layers and components of the neural network.	Computational complexity is high, and the overhead is significant.	Regression or classification models
Uncertainty Calculation	The degree of uncertainty in the model's output caused by test inputs.	The model's output results are required.	Simple and direct.	Difficulties in setting uncertainty thresholds.	Regression or classification models
Data Distribution States	Data distribution status of test inputs.	Building the data distribution.	Considering the global distribution state of the test inputs.	Difficulties in constructing the data distribution.	classification models
Embedded Feature Extraction	Similarity or distance between test inputs.	Feature vectors need to be extracted.	High generality.	Feature extraction is challenging, and computation is complex.	Regression or classification models
Mutation Analysis	The degree of variation in test inputs.	Test inputs that undergo variation.	The test space is large.	Difficulty in formulating mutation rules.	Regression or classification models

5. DISCUSSION

Impact of Different Test Input Selection Criteria. The model coverage criterion focuses on whether the test set covers the internal structure of the model and helps to comprehensively assess the model's performance. The model robustness criterion focuses on the robustness of the model to anomalous inputs and can reveal model vulnerabilities and deficiencies. The Model Uncertainty criterion focuses on the model's predictive confidence in the inputs and expands the test set to cover edge inputs. The test input diversity criterion ensures that the test set covers different data distributions and scenarios to comprehensively assess model performance.

Impact of Different Testing Input Prioritization Strategies. Coverage-based methods are able to comprehensively assess the validity of the test set by considering the coverage of the internal structure of the neural network and are particularly suited to ensuring comprehensive testing of the model structure. Methods based on uncertainty calculation focus on the model prediction uncertainty of the test samples, are more effective in mining the performance of the model in edge cases, and are suitable for improving the robustness of the model. Methods based on data distribution state help to evaluate model performance in different data scenarios by considering the distribution of data in the model and are suitable for tasks with diverse data distributions. Methods based on feature extraction focus on the important characteristics of test samples and can select key test samples in a targeted manner, which is suitable for complex tasks and large-scale neural networks. Methods based on mutation analysis help to find the sensitivity of the model to input changes by introducing test samples that make small changes to the model and are suitable for finding the boundary cases of the model.

Challenge. Despite numerous research achievements in the current studies related to the selection and prioritization of test inputs for deep neural networks, challenges persist in this research field due to the increasing scale of deep neural network models and their datasets, coupled with the inherent lack of interpretability in these models.

- As the scale of deep learning models continues to grow and datasets involved become increasingly vast, the challenge of efficiently selecting and prioritizing test inputs within limited time and computational resources becomes more formidable.
- Deep learning models are often regarded as "black boxes." The challenge lies in introducing interpretability into the process of test input selection and prioritization, aiding researchers and practitioners in better understanding the behavior and performance of neural network models. This will be a crucial challenge.
- Different application scenarios and specific domains may have varying requirements for test input needs and prioritization. The future challenge lies in designing flexible test input selection and prioritization algorithms that can be personalized and customized based on the specific demands corresponding to a given scenario.

Threat To Validity. In conducting this literature review, certain potential threats to the validity and comprehensiveness of the analysis were considered. These threats include publication bias, as the literature search might not have captured all relevant studies, language bias, as only papers published in English were included, and selection bias, as the criteria used for paper inclusion might introduce subjectivity.

Additionally, the temporal bias could be a concern, as newer studies may not have been adequately represented in the review. Despite these potential threats, efforts were made to mitigate them through a systematic search strategy, inclusion criteria, and a comprehensive evaluation of the selected papers. The acknowledgment of these threats is essential for a nuanced interpretation of the findings and suggestions for future research.

6. CONCLUSION

With the widespread application of deep neural networks in various critical domains, testing deep neural network models is crucial to ensure model performance, robustness, and reliability. Test input selection plays a central role in the testing process. This article provides a detailed review of test input selection and prioritization for deep neural networks. Firstly, it introduces the fundamental concepts and significance of testing deep neural networks, emphasizing the importance of test input selection. Secondly, it summarizes and categorizes existing criteria for test input selection in current research. It then provides a brief overview of two different test input selection methods. The current study subsequently focuses on various methods for test input prioritization, categorizing them based on criteria such as uncertainty, data distribution status, embedding features, and more. Finally, it discusses the challenges in this research area and outlines future work, concluding with a reliability analysis and summary.

REFERENCES

[1] Sudha, V. and Vijendran, A.S., 2024. OSD-DNN: Oil Spill Detection using Deep Neural Networks. International Journal of Performability Engineering, 20(2), pp. 57-67.

[2] Bhandari, R., Singla, S., Sharma, P. and Kang, S.S., 2024. AINIS: An Intelligent Network Intrusion System. International Journal of Performability Engineering, 20(1), 24-31.

[3] Sun, Y., Chen, Y., Wang, X. and Tang, X., 2014. Deep learning face representation by joint identification-verification. Advances in neural information processing systems, 27.

[4] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J. and Zhang, X., 2016. End to end learning for self-driving cars. arxiv preprint arxiv:1604.07316.

[5] Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N. and Terzopoulos, D., 2021. Image segmentation using deep learning: A survey. IEEE transactions on pattern analysis and machine intelligence, 44(7), pp.3523-3542.

[6] Chen, J., He, X., Lin, Q., Zhang, H., Hao, D., Gao, F., Xu, Z., Dang, Y. and Zhang, D., 2019, November. Continuous incident triage for large-scale online service systems. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 364-375). IEEE.

[7] Chen, J., He, X., Lin, Q., Xu, Y., Zhang, H., Hao, D., Gao, F., Xu, Z., Dang, Y. and Zhang, D., 2019, May. An empirical investigation of incident triage for online service systems. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 111-120). IEEE.

[8] Chen, J., Zhang, S., He, X., Lin, Q., Zhang, H., Hao, D., Kang, Y., Gao, F., Xu, Z., Dang, Y. and Zhang, D., 2020, December. How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (pp. 373-384).

[9] Yang, L., Chen, J., Wang, Z., Wang, W., Jiang, J., Dong, X. and Zhang, W., 2021, May. Semi-supervised log-based anomaly detection via probabilistic label estimation. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 1448-1460). IEEE.

[10] Wang, Z., You, H., Chen, J., Zhang, Y., Dong, X. and Zhang, W., 2021, May. Prioritizing test inputs for deep neural networks via mutation analysis. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 397-409). IEEE.

[11] Zhang, M., Zhang, Y., Zhang, L., Liu, C. and Khurshid, S., 2018, September. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (pp. 132-142).

[12] Dunn, I., Pouget, H., Kroening, D. and Melham, T., 2021, July. Exposing previously undetectable faults in deep neural networks. In Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 56-66).

[13] Ji, P., Feng, Y., Liu, J., Zhao, Z. and Xu, B., 2021, November. Automated testing for machine translation via constituency invariance. In 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 468-479). IEEE.

[14] Odena, A., Olsson, C., Andersen, D. and Goodfellow, I., 2019, May. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In International Conference on Machine Learning (pp. 4901-4911). PMLR.

[15] Ma, S., Liu, Y., Lee, W.C., Zhang, X. and Grama, A., 2018, October. MODE: automated neural network model debugging via state differential analysis and input selection. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 175-186).

[16] Feng, Y., Shi, Q., Gao, X., Wan, J., Fang, C. and Chen, Z., 2020, July. 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 (pp. 177-188).

[17] Zhou, J., Li, F., Dong, J., Zhang, H. and Hao, D., 2020, October. Cost-effective testing of a deep learning model through input reduction. In 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE) (pp. 289-300). IEEE.

[18] Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J. and See, S., 2019, July. 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 (pp. 146-157).

[19] Kim, J., Feldt, R. and Yoo, S., 2019, May. Guiding deep learning system testing using surprise adequacy. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 1039-1049). IEEE.

[20] Zhang, J.M., Harman, M., Ma, L. and Liu, Y., 2020. Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering, 48(1), pp.1-36.

[21] Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M. and Tonella, P., 2020. Testing machine learning based systems: a systematic mapping. Empirical Software Engineering, 25, pp.5193-5254.

[22] Masuda, S., Ono, K., Yasue, T. and Hosokawa, N., 2018, April. A survey of software quality for machine learning applications. In 2018 IEEE International conference on software testing, verification and validation workshops (ICSTW) (pp. 279-284). IEEE.

[23] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y. and Alsaadi, F.E., 2017. A survey of deep neural network architectures and their applications. Neurocomputing, 234, pp.11-26.

[24] Yan, M., Chen, J., Cao, X., Wu, Z., Kang, Y. and Wang, Z., 2023. Revisiting deep neural network test coverage from the test effectiveness perspective. Journal of Software: Evolution and Process, p.e2561.

[25] Pei, K., Cao, Y., Yang, J. and Jana, S., 2017, October. Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles (pp. 1-18).

[26] Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., Liu, Y. and Zhao, J., 2018, September. Deepgauge: Multi-granularity testing criteria for deep learning systems. In Proceedings of the 33rd ACM/IEEE international conference on automated software engineering (pp. 120-131).

[27] Usman, M., Sun, Y., Gopinath, D., Dange, R., Manolache, L. and Păsăreanu, C.S., 2023. An overview of structural coverage metrics for testing neural networks. International Journal on Software Tools for Technology Transfer, 25(3), pp.393-405.

[28] Tian, Y., Zhong, Z., Ordonez, V. and Ray, B., 2019. Testing deep neural network based image classifiers. CoRR, abs/1905.07831.

[29] Sun, Y., Huang, X., Kroening, D., Sharp, J., Hill, M. and Ashmore, R., 2018. Testing deep neural networks. arxiv preprint arxiv:1803.04792.

[30] Ma, L., Juefei-Xu, F., Xue, M., Li, B., Li, L., Liu, Y. and Zhao, J., 2019, February. Deepct: Tomographic combinatorial testing for deep learning systems. In 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 614-618). IEEE.

[31] Sekhon, J. and Fleming, C., 2019, May. Towards improved testing for deep learning. In 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER) (pp. 85-88). IEEE.

[32] Chen, Y., Wang, Z., Wang, D., Fang, C. and Chen, Z., 2019, April. Variable strength combinatorial testing for deep neural networks. In 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 281-284). IEEE.

[33] Wang, D., Wang, Z., Fang, C., Chen, Y. and Chen, Z., 2019, April. Deeppath: Path-driven testing criteria for deep neural networks. In 2019 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 119-120). IEEE.

[34] Zhou, Z., Dou, W., Liu, J., Zhang, C., Wei, J. and Ye, D., 2021, March. Deepcon: Contribution coverage testing for deep learning systems. In 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 189-200). IEEE.

[35] Yuan, Y., Pang, Q. and Wang, S., 2023, May. Revisiting neuron coverage for dnn testing: A layer-wise and distribution-aware criterion. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 1200-1212). IEEE.

[36] Ji, Z., Ma, P., Yuan, Y. and Wang, S., 2023, May. Cc: Causality-aware coverage criterion for deep neural networks. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 1788-1800). IEEE.

[37] Hu, L., Ji, S., Dai, Q. and Zhang, P., 2021, August. Evolutionary Generation of Test Case for Deep Neural Network Based on Coverage Guidance. In 2021 IEEE International Conference on Artificial Intelligence Testing (AITest) (pp. 19-20). IEEE.

[38] Du, X., Xie, X., Li, Y., Ma, L., Zhao, J. and Liu, Y., 2018. Deepcruiser: Automated guided testing for stateful deep learning systems. arxiv preprint arxiv:1812.05339.

[39] Ruan, W., Huang, X. and Kwiatkowska, M., 2018. Reachability analysis of deep neural networks with provable guarantees. arxiv preprint arxiv:1805.02242.

[40] Huang, X., Kwiatkowska, M., Wang, S. and Wu, M., 2017. Safety verification of deep neural networks. In Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30 (pp. 3-29). Springer International Publishing.

[41] Hu, Q., Guo, Y., Xie, X., Cordy, M., Papadakis, M., Ma, L. and Le Traon, Y., 2023, May. Aries: Efficient testing of deep neural networks via labeling-free accuracy estimation. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 1776-1787). IEEE.

[42] Ling, X., Ji, S., Zou, J., Wang, J., Wu, C., Li, B. and Wang, T., 2019, May. Deepsec: A uniform platform for security analysis of deep learning model. In 2019 IEEE Symposium on Security and Privacy (SP) (pp. 673-690). IEEE.

[43] Guo, C., Pleiss, G., Sun, Y. and Weinberger, K.Q., 2017, July. On calibration of modern neural networks. In International conference on machine learning (pp. 1321-1330). PMLR.

[44] Zhou, M. and Patel, V.M., 2022. Enhancing adversarial robustness for deep metric learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 15325-15334).

[45] Mangal, R., Nori, A.V. and Orso, A., 2019, May. Robustness of neural networks: A probabilistic and practical approach. In 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER) (pp. 93-96). IEEE.

[46] Weng, T.W., Zhang, H., Chen, P.Y., Yi, J., Su, D., Gao, Y., Hsieh, C.J. and Daniel, L., 2018. Evaluating the robustness of neural networks: An extreme value theory approach. arxiv preprint arxiv:1801.10578.

[47] Katz, G., Barrett, C., Dill, D.L., Julian, K. and Kochenderfer, M.J., 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30 (pp. 97-117). Springer International Publishing.

[48] Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A. and Criminisi, A., 2016. Measuring neural net robustness with constraints. Advances in neural information processing systems, 29.

[49] Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A. and Kurakin, A., 2019. On evaluating adversarial robustness. arxiv preprint arxiv:1902.06705.

[50] Cheng, C.H., Huang, C.H., Ruess, H. and Yasuoka, H., 2018, October. Towards dependability metrics for neural networks. In 2018 16th ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE) (pp. 1-4). IEEE.

[51] Chen, Y., Wang, Z., Wang, D., Yao, Y. and Chen, Z., 2019, April. Behavior pattern-driven test case selection for deep neural networks. In 2019 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 89-90). IEEE.

[52] Shi, Y., Yin, B., Zheng, Z. and Li, T., 2021, December. An empirical study on test case prioritization metrics for deep neural networks. In 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS) (pp. 157-166). IEEE.

[53] Gal, Y. and Ghahramani, Z., 2016, June. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning (pp. 1050-1059). PMLR.

[54] Kendall, A. and Gal, Y., 2017. What uncertainties do we need in bayesian deep learning for computer vision?. Advances in neural information processing systems, 30.

[55] Ma, W., Papadakis, M., Tsakmalis, A., Cordy, M. and Traon, Y.L., 2021. Test selection for deep learning systems. ACM Transactions on Software Engineering and Methodology (TOSEM), 30(2), pp.1-22.

[56] Van Amersfoort, J., Smith, L., Teh, Y.W. and Gal, Y., 2020, November. Uncertainty estimation using a single deep deterministic neural network. In International conference on machine learning (pp. 9690-9700). PMLR.

[57] Bao, S., Sha, C., Chen, B., Peng, X. and Zhao, W., 2023, July. In defense of simple techniques for neural network test case selection. In Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 501-513).

[58] Aghababaeyan, Z., Abdellatif, M., Dadkhah, M. and Briand, L., 2023. Deepgd: A multi-objective black-box test selection approach for deep neural networks. ACM Transactions on Software Engineering and Methodology.

[59] Chen, J., Wu, Z., Wang, Z., You, H., Zhang, L. and Yan, M., 2020. Practical accuracy estimation for efficient deep neural network testing. ACM Transactions on Software Engineering and Methodology (TOSEM), 29(4), pp.1-35.

[60] Mani, S., Sankaran, A., Tamilselvam, S. and Sethi, A., 2019. Coverage testing of deep learning models using dataset characterization. arxiv preprint arxiv:1911.07309.

[61] Hao, Y., Huang, Z., Guo, H. and Shen, G., 2023, March. Test input selection for deep neural network enhancement based on multiple-objective optimization. In 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 534-545). IEEE.

[62] Gerasimou, S., Eniser, H.F., Sen, A. and Cakan, A., 2020, June. Importance-driven deep learning system testing. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (pp. 702-713).

[63] Zhu, J., Tao, C., Guo, H. and Ju, Y., 2023, November. DeepTD: Diversity-Guided Deep Neural Network Test Generation. In International Symposium on Dependable Software Engineering: Theories, Tools, and Applications (pp. 419-433). Singapore: Springer Nature Singapore.

[64] Guo, H., Tao, C. and Huang, Z., 2023, October. Multi-Objective White-Box Test Input Selection for Deep Neural

Network Model Enhancement. In 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE) (pp. 521-532). IEEE.

[65] Sun, Y., Wu, M., Ruan, W., Huang, X., Kwiatkowska, M. and Kroening, D., 2018, September. Concolic testing for deep neural networks. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (pp. 109-119).

[66] Li, Z., Ma, X., Xu, C., Cao, C., Xu, J. and Lü, J., 2019, August. Boosting operational dnn testing efficiency through conditioning. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 499-509).

[67] Byun, T., Sharma, V., Vijayakumar, A., Rayadurgam, S. and Cofer, D., 2019, April. Input prioritization for testing neural networks. In 2019 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 63-70). IEEE.

[68] Weiss, M. and Tonella, P., 2022, July. Simple techniques work surprisingly well for neural network test prioritization and active learning (replicability study). In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 139-150).

[69] Wang, H., Xu, J., Xu, C., Ma, X. and Lu, J., 2020, June. Dissector: Input validation for deep learning applications by crossing-layer dissection. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (pp. 727-738).

[70] Pan, Z., Zhou, S., Wang, J., Wang, J., Jia, J. and Feng, Y., 2022, August. Test case prioritization for deep neural networks. In 2022 9th International Conference on Dependable Systems and Their Applications (DSA) (pp. 624-628). IEEE.

[71] Sun, W., Yan, M., Liu, Z. and Lo, D., 2023. Robust Test Selection for Deep Neural Networks. IEEE Transactions on Software Engineering, 49(12), pp.5250-5278.

[72] Zhang, L., Sun, X., Li, Y. and Zhang, Z., 2019. A noise-sensitivity-analysis-based test prioritization technique for deep neural networks. arxiv preprint arxiv:1901.00054.

[73] Zhang, K., Zhang, Y., Zhang, L., Gao, H., Yan, R. and Yan, J., 2020, December. Neuron activation frequency based test case prioritization. In 2020 International Symposium on Theoretical Aspects of Software Engineering (TASE) (pp. 81-88). IEEE.

[74] Yan, R., Chen, Y., Gao, H. and Yan, J., 2022. Test case prioritization with neuron valuation based pattern. Science of Computer Programming, 215, p.102761.

[75] Tao, Y., Tao, C., Guo, H. and Li, B., 2022, November. Tpfl: Test input prioritization for deep neural networks based on fault localization. In International Conference on Advanced Data Mining and Applications (pp. 368-383). Cham: Springer Nature Switzerland.

[76] Weng, S., Feng, Y., Yin, Y. and Liu, J., 2023, August. Prioritizing Testing Instances to Enhance the Robustness of Object Detection Systems. In Proceedings of the 14th Asia-Pacific Symposium on Internetware (pp. 194-204).

[77] Chen, J., Ge, J. and Zheng, H., 2023. ActGraph: prioritization of test cases based on deep neural network activation graph. Automated Software Engineering, 30(2), p.28.

[78] Li, Y., Li, M., Lai, Q., Liu, Y. and Xu, Q., 2021. Testrank: Bringing order into unlabeled test instances for deep learning tasks. Advances in Neural Information Processing Systems, 34, pp.20874-20886.

[79] Zheng, H., Chen, J. and Jin, H., 2023, September. CertPri: certifiable prioritization for deep neural networks via movement cost in feature space. In 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1-13). IEEE.

[80] Chen, J., Wang, G., Hao, D., Xiong, Y., Zhang, H., Zhang, L. and Xie, B., 2018. Coverage prediction for accelerating compiler testing. IEEE Transactions on Software Engineering, 47(2), pp.261-278.

[81] Chen, J., 2018, May. Learning to accelerate compiler testing. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings (pp. 472-475).

[82] Chen, J., Patra, J., Pradel, M., Xiong, Y., Zhang, H., Hao, D. and Zhang, L., 2020. A survey of compiler testing. ACM Computing Surveys (CSUR), 53(1), pp.1-36.

[83] Dang, X., Li, Y., Papadakis, M., Klein, J., Bissyandé, T.F. and Le Traon, Y., 2023. Graphprior: mutation-based test input prioritization for graph neural networks. ACM Transactions on Software Engineering and Methodology, 33(1), pp.1-40.

[84] Wei, Z., Wang, H., Ashraf, I. and Chan, W.K., 2022, December. Predictive mutation analysis of test case prioritization for deep neural networks. In 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS) (pp. 682-693). IEEE.

[85] Wang, J., Dong, G., Sun, J., Wang, X. and Zhang, P., 2019, May. Adversarial sample detection for deep neural network through model mutation testing. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 1245-1256). IEEE.

[86] Gao, X., Feng, Y., Yin, Y., Liu, Z., Chen, Z. and Xu, B., 2022, May. Adaptive test selection for deep neural networks. In Proceedings of the 44th International Conference on Software Engineering (pp. 73-85).

[87] Shen, W., Li, Y., Chen, L., Han, Y., Zhou, Y. and Xu, B., 2020, December. Multiple-boundary clustering and prioritization to promote neural network retraining. In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (pp. 410-422).

[88] Al-Qadasi, H., Wu, C., Falcone, Y. and Bensalem, S., 2022, August. DeepAbstraction: 2-level prioritization for unlabeled test inputs in deep neural networks. In 2022 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 64-71). IEEE.