Received 11 October 2023, accepted 22 October 2023, date of publication 26 October 2023, date of current version 1 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3327820

RESEARCH ARTICLE

Distribution Aware Testing Framework for Deep Neural Networks

DEMET DEMIR[1], AYSU BETIN CAN[1], AND ELIF SURER[2,3], (Member, IEEE)

[1]Department of Information Systems, Graduate School of Informatics, Middle East Technical University, 06800 Ankara, Turkey
[2]Department of Modeling and Simulation, Graduate School of Informatics, Middle East Technical University, 06800 Ankara, Turkey
[3]Neuroscience and Neurotechnology Center of Excellence, Middle East Technical University, 06800 Ankara, Turkey

Corresponding author: Demet Demir (demet.demir@metu.edu.tr)

ABSTRACT The increasing use of deep learning (DL) in safety-critical applications highlights the critical need for systematic and effective testing to ensure system reliability and quality. In this context, researchers have conducted various DL testing studies to identify weaknesses in Deep Neural Network (DNN) models, including exploring test coverage, generating challenging test inputs, and test selection. In this study, we propose a generic DNN testing framework that takes into consideration the distribution of test data and prioritizes them based on their potential to cause incorrect predictions by the tested DNN model. We evaluated the proposed framework using the image classification as a use case. We conducted empirical evaluations by implementing each phase with carefully chosen methods. We employed Variational Autoencoders to identify and eliminate out-of-distribution data from the test datasets. Additionally, we prioritize test data that increase uncertainty in the model, as these cases are more likely to reveal potential faults. The elimination of out-of-distribution data enables a more focused analysis to uncover the sources of DNN failures while using prioritized test data reduces the cost of test data labeling. Furthermore, we explored the use of post-hoc explainability methods to identify the cause of incorrect predictions, a process similar to debugging. This study can be a prelude to incorporating explainability methods into the model development process after testing.

INDEX TERMS Data distribution, deep learning testing, explainability, test selection and prioritization, uncertainty.

I. INTRODUCTION

Deep Learning (DL) [1] has achieved great success in real-world applications such as image recognition, speech recognition, and machine translation, thanks to advancements in deep learning algorithms, improved hardware capability, increased data volume, and ease of data access [2]. DL systems are adopted in safety and security-critical systems like autonomous driving vehicles [3], medical treatments [4], robotics [5], and malware detection [6] based on their ability to match and sometimes surpass human performance in complex tasks.

Despite their outstanding experimental results, real-world deployments of DL systems may produce unexpected or incorrect behavior. These vulnerabilities may result from insufficient training data, inadequate training procedures, the inability to generalize in the deployment environment, or other systematic errors. Failures in safety-critical systems may result in the loss of human life; therefore, the use of DL systems in such applications necessitates the assurance of their safety and trustworthiness.

Testing is the primary instrument for ensuring the reliability of software. The automotive (ISO 26262 [7]) and avionics (DO-178 [8]) standards outline the testing criteria that must be followed for certification of safety-critical software. However, since DL systems are inherently different in nature from programmed software, these principles do not directly apply to them. DL systems differ from programmed software due to their development process: 1) DL systems do not have well-defined specifications; they learn how to behave from training data that are representative of the problem domain, 2) during the training of a Deep Neural

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana.

Network (DNN) in a DL system, its behavior is encoded to its parameters, and learning continues until the desired level of accuracy is achieved, and 3) after training, it is expected that the learned behavior would generalize to new valid inputs other than the training data. Thus, the behavior of the software is not programmed by the developer in accordance with predefined specifications; instead, it is learned from training data. Also, unlike programmed software, DNNs lack a clear decision path from which the execution flow may be easily followed.

Due to these characteristics, conventional software testing methodologies cannot be applied directly to DL systems. Typically, in DNN model development, the whole labeled dataset is divided into training, validation, and test datasets. Test data are not used during the training of the model. DNN testing is performed with this reserved dataset from the labeled data at hand, with testing accuracy serving as a quality metric for DNNs. In the rest of this paper, we use the term *original test data* for this test dataset used in the model development. Note that the original test dataset consists of a limited number of labeled data.

There is a growing need to test DNNs more thoroughly to improve the trustworthiness of models before they are used in critical safety systems in the real world. In the last decade, many academic studies have been conducted in the field of DL testing to define a more systematic testing process and better assess the reliability of DL systems.

In this study, we propose a DNN testing framework with three phases: Test Input Generation, Test Input Selection, and Test Results Interpretation. The objective of this framework is to evaluate the reliability of a DNN model by subjecting it to a diverse set of test inputs while considering the distribution of the test dataset. Labeling of test data and examining the test results are processes that require manual human effort in DNN testing. Our goal is to ensure that human labor is used effectively by focusing on test inputs that are likely to be more informative and beneficial for improving the model's performance and robustness. Therefore, we give prioritization to test data that cause the DNN model to make incorrect predictions. While doing this, we make sure that the test data are coming from a similar data distribution to the dataset on which the model is trained.

The test dataset plays a crucial role in thoroughly testing a DNN model to ensure its robustness and reliability. The first phase of our test framework concentrates on establishing the test dataset. Data diversity and comprehensiveness are extremely important factors in testing. Further testing beyond the original test data can be realized by collecting new data from the real world or by generating them synthetically. Synthetic data generation methods are commonly used in recent DL testing studies to increase the number and diversification of test data [9], [10], [11], [12]. In some of these studies, test data generation is guided by the increase in the coverage of the DNN model structure, such as neuron coverage [13], [14], [15], [16], [17], [18]. Neuron coverage for a test suite is the ratio of the number of neurons whose output exceeds a predefined threshold value to the total number of neurons in the DNN. The main idea in the studies that aim to generate test data that increases neuron coverage is that the activation of new neurons that have not been activated before can be helpful to explore different parts of the DNN model and identify erroneous behaviors that have not yet been discovered. It is later shown that, despite their high coverage scores, some of these approaches generate a high number of invalid data that are not recognizable even by humans [19]. Even if the synthetically generated data are understandable to humans, they could differ from the dataset that is used to train the DNN model. For instance, if the DNN model is trained to recognize handwritten digits, the digit dataset written in Times New Roman style is a different data distribution for the model. As a second issue, although a high number of test data can be generated automatically, these generated test data need to be labeled before being used in testing, and labeling is performed manually most of the time. These problems are not unique to the synthetically generated test data but also apply to large amounts of unlabeled test data gathered from the real world.

Test selection can be used to overcome these problems. By concentrating the DNN test on selected and prioritized data from a larger dataset, effective test selection reduces the cost of testing in two ways—first, by reducing the cost of labeling additional test data (produced or gathered) and second, by reducing the effort required to examine the results of tests.

Test selection methods in the literature generally ignore the data distribution shifts within the test dataset. The training inputs of a DNN come from a data distribution, and the DNN adopts itself to this distribution while optimizing its parameters to solve the problem defined in this context. Data distribution-shift refers to a phenomenon in which the statistical properties of training data and test data are different. There are mainly two types of data distribution shifts: covariate shift and semantic shift [20]. In covariate data shift, the label space remains the same across two datasets. Examples of this type of shift include style changes or perturbations due to adversarial attacks. Adversarial attacks are a class of techniques in which small, imperceptible manipulations are performed on the input data to deceive the DNN model and produce incorrect output [21]. However, in the case of semantic shift, the labels of the data change, and datasets may have different label spaces. Covariate data shifts are expected to be handled by the DNN model within the scope of generalization. However, the degree of shift is important for the model to handle this shift. As the degree of shift increases, it becomes more difficult to make correct predictions for this data by the model since they differentiate from the training dataset.

Out-of-distribution (OOD) data are defined as data that come from a different data distribution than training data distribution, while in-distribution data are used inversely. OOD data detection is a challenging problem in DL systems, and several approaches have been proposed [20]. OOD detection is studied for semantic data shifts with datasets that have different data label spaces. To date, only a limited number of studies have been conducted on distribution-sensitive DL testing, which only started to be researched in recent years. These recent studies demonstrate the need for additional research on this topic. These studies can be categorized into two groups: evaluation of OOD data impact on DNN testing [22], [23] and application of distribution awareness in test data generation [11], [12], [19]. Berend et al. [22] carried out a study that evaluates the relationship between DL testing criteria and OOD data ratio in the test suite constructed with the use of corresponding testing criteria. Furthermore, they added the selected OOD data to the training dataset and retrained the model. They investigated the impact of OOD data in the retraining dataset on improving the robustness of the DNN model. Hu et al. [23] conducted a comprehensive empirical study to assess the efficiency of the DL test data selection metrics in terms of accuracy improvement after retraining the model using the selected test data. They employed many different generated test datasets with varying percentages of in-distribution and OOD data in their experiments. On the other hand, studies that fall in the second group [11], [12], [19] use generative models with the objective of creating solely in-distribution test data.

We think that the DNN model's compliance with the test requirements should be validated first. The model's success in settings not included in the training set, i.e., the evaluation of the model's generalization scope, should be examined afterward. Based on this idea, we integrate Out-of-Distribution detection into the Test Selection phase. With this method, the data with a covariate shift above a predefined threshold is identified as OOD and removed from the test dataset. Therefore, our study can be evaluated differently from these two categories.

In the second stage of Test Selection, we prioritize the test data according to their ability to reveal faults. The fundamental objective of DL testing is to detect the weaknesses of the tested DNN model, hence, the most valuable test cases are those for which the model makes incorrect predictions. Recently, studies have been conducted to analyze metrics in terms of test selection. These metrics have been compared based on fault detection as well as adversarial input detection and retraining performance gain [23], [24], [25], [26], [27], [28], [29], [30].

In most of the studies in the literature, after identifying the test data that result in incorrect predictions, they are added to the training dataset, and the model is retrained to improve its accuracy on these data samples [23], [24], [27], [28], [29]. As an alternative method, we argue that if the model's reasoning for the prediction can be understood, a wiser decision can be made than retraining the model by adding all the incorrectly predicted data to the training dataset.

For this purpose, in the Test Results Interpretation phase, we employ post-hoc explainability methods for investigating the incorrectly classified test data. An explanation is an answer to why questions [31]. In machine learning (ML), the explanation is defined as the details or reasons that are given by the model to make its functioning clear or easy to understand for its audience [32]. The ML models can be categorized into two groups according to the way they provide it: models that are interpretable by design (transparent models) and those that can be explained by external techniques. As DNN models are black box models [33] and not transparent, post-hoc explainability techniques are used. Post-hoc explainability techniques refer to techniques that are used to understand why the model reached a particular prediction without modifying the model's architecture and training process.

We conducted experiments with image classification datasets by choosing methods to be employed at each phase of the proposed framework. In the test input generation phase, we employed data augmentation techniques, adversarial attacks, and generative models, which are most used in the DL testing literature. Data augmentation is increasing the size of the dataset and diversifying it by applying transformations and modifications to the original data without collecting additional real-world samples. In the second phase, we utilized a generative model for OOD data detection, namely Variational Autoencoder (VAE). For prioritizing test data, we utilized state-of-the-art uncertainty quantification approaches and metrics and evaluated their fault detection capability. DL uncertainty is the degree of how much the model is unsure of its prediction. A classification model outputs probability values for each class, which are also known as confidence. DNN models can make incorrect predictions with overconfidence. In order to address this problem, uncertainty estimation is used to provide additional insight into the predictions of the models. Uncertainty is measured with uncertainty quantification methods which are used to estimate and quantify the uncertainty associated with the particular predictions of the models. In uncertainty-based test selection, test data with higher uncertainty values are prioritized. For uncertainty-based test selection, we employ the Deep Ensemble (DE) method [34] and Variational Inference with the Flipout (VI-F) [35] method, along with the commonly assessed Margin, Entropy, and Least Confidence metrics. To the best of our knowledge, this is the first study to investigate DE and VI-F uncertainty quantification methods for test selection in DL systems. We also compared the effectiveness of these prioritization techniques with state-of-the-art test selection approaches.

In the last phase, we used the visualization methods, which aim to provide an explanation for the model's decision by highlighting regions of the input image that significantly influence the DNN's output. After identifying the test data that led to incorrect predictions by the DL model, we analyzed them using visualization techniques. For test data visualization, Grad-Cam [36], Grad-Cam++ [37], and Score-Cam [38] were investigated. The objective of visualization is to determine whether it is possible to understand the reason behind the model's incorrect predictions. The details of methods employed in each phase are given in Section V-B.

With the experiments, we evaluated the effectiveness of uncertainty-based test selection metrics with a data distribution-aware point of view, assessed their performance in prioritizing fault-revealing test data, and investigated the use of explainability methods in testing. We answered the following research questions:

RQ1: Which test data generation method is more likely to generate OOD data?

RQ2: How effective are uncertainty-based test selection metrics for prioritizing the fault-revealing data instances in datasets generated with different techniques and identified as in-distribution data?

RQ3: Does the existence of OOD data in the test dataset impact the success of the test selection methods?

RQ4: Is it possible to use post-explainability methods in testing to understand the root cause of test failures?

In our experiments, we used two popular image classification datasets (i.e., MNIST and CIFAR-10) with two well-known DNN models for each dataset. We generated new test data employing three test data generation techniques. We observed that each technique produces a substantial amount of OOD data that should be considered in testing. Then, we prioritized the generated test data according to nine test prioritization strategies and observed that uncertainty-based prioritization aids in identifying the fault-revealing data with an average percentage of fault detection (APFD) scores between 0.69 to 0.92. Interestingly, our results reveal that test prioritization strategies that employ Variational Inference with the Flipout method have the lowest performance on all evaluated test datasets in the category of uncertainty-based test prioritization methods. Also, with this study, we took the first step toward incorporating DL explainability techniques into the testing process and conducted a sample analysis. As a result of our analysis, we got promising results regarding the use of visualization techniques to understand the rationale behind the model decision for test failures.

Our contributions with this study can be summarized as follows:

- We proposed a DNN test framework that employs out-of-distribution detection for test data and eliminates OOD data in the test dataset. In this manner, the testing of the DNN model is carried out with a test dataset that comes from a similar data distribution to the training dataset.
- We compared the state-of-the-art uncertainty quantification methods and metrics in terms of test data prioritization. This evaluation was performed with several test datasets generated by various test data generation methods, and the competence of the metrics in different contexts was also compared.
- We assessed the use of visual post-explanation methods for the explainability of the model's decision logic for incorrectly predicted test data and the proposal of actions for improving the model's performance.

II. BACKGROUND AND RELATED WORK

A. DL TESTING

Deep Learning is a data-driven paradigm that employs deep neural networks (DNN) to learn how to make decisions from the data. In recent years, there has been a growing interest in DL testing, and the number of testing techniques proposed has increased exponentially, mainly contributing to i) test data generation, (ii) test adequacy evaluation, (iii) test selection, and (iv) test oracle determination [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [39].

Several studies have been carried out to transfer the knowledge and experience gained in the field of software testing to DL testing without increasing the need for labeled data. Metamorphic relations and cross-referencing are mostly preferred as test oracles since they eliminate the need for manual labeling. In the metamorphic test method, the software function is examined, and metamorphic relations existing in the function are determined. The metamorphic relationship can be defined as the relationship between the input changes and corresponding output changes. When using metamorphic relationships in software testing, it is common to transform the input data so that the expected output remains the same. With the same approach, metamorphic relationships are determined for DL systems as well. For example, for an image classification DNN system, input image changes by linear and affine transformations or adversarial attacks can be called metamorphic transformations since the expected output remains the same despite the changed input data. In DL testing, metamorphic transformations are frequently utilized to generate test data [9], [13], [14], [17].

Existing synthetic test generation methods do not consider the likelihood of generated data distribution similarity to that of training data. Brend et al. [22] conducted an empirical study to evaluate the relationship between the test coverage criteria, mutation operators, and data distribution. They emphasize the importance of distribution awareness in DL testing techniques for more effective testing. It is essential to identify the reason for test failures, which may be related to the DNN model or data attributes.

B. DISTRIBUTION-AWARE TESTING

Several techniques have been developed for detecting OOD data in DL, and in a recent review study, they were categorized as classification-based, density-based, and distance-based methods [20]. In the category of density-based methods, generative models are used to learn the training data distribution and to estimate the likelihood that new input comes from the same distribution as the training data. Variational Autoencoders (VAE) are employed to detect OOD data with this approach.

Dola et al. [19] conducted a study that evaluates the validity of the test data generated by previously proposed DL testing techniques (DeepXplore [14], DLFuzz [15], and DeepConcolic [18]) and showed that these techniques generate a high percentage of out-of-distribution data. They used VAEs for both detection of OOD data and the generation of new in-in-distribution test data. Kang et al. [11] also used VAEs in a search-based DNN testing strategy to generate new test data images. They applied differential testing using multiple DNN models and guided the test data generation by searching test data that cause distinct output values from the models. Similarly, Byun et al. [12] proposed a distribution-aware test case generation and selection method that employs a Conditional Variational Auto Encoder (CVAE). We use VAEs to identify OOD data in test datasets and remove them, as opposed to generating new instances of test data.

C. DL UNCERTAINTY

DL uncertainty is categorized as aleatoric uncertainty and epistemic uncertainty [40]. Measuring the uncertainty of the DNN model is important to estimate the probability of incorrect predictions accurately [41]. In the classification DNN models, the confidence of the model is determined by the output probability for the predicted class. The confidence value is not always sufficient to measure the model's uncertainty since, in some cases, DL systems make incorrect predictions with high confidence values, such as in adversarial samples [42]. Quantifying the DNN model's uncertainty is an open research problem, and several techniques and metrics have been proposed so far [43], [44], [45], [46].

D. TEST SELECTION

Test selection in DL testing is conducted by prioritizing a subset of data from a large test dataset with different objectives such as fault detection, adversarial input detection, or model improvement with retraining [11], [23], [24], [25], [26], [27], [28], [29], [30], [47]. Although most of the test data selection studies are domain agnostic, particular studies have been conducted for domains like automotive and medical [48], [49].

Feng et al. proposed the DeepGini [28], which is a statistical approach for test prioritization. For a classification problem, when the prediction probabilities for different classes are close to each other, the likelihood that the model is uncertain about its prediction and, as a result, the possibility for misclassification is high. Originating from the impurity in statistics, a metric similar to entropy but easier to calculate is suggested with DeepGini.

$$\text{DeepGini} = 1 - \sum_{i} {p_i}^2$$
$$p_i : \text{predicted probability for class } i \qquad (1)$$

Shen et al. introduced multiple-boundary clustering and prioritization (MCP) [29], a test case selection method based on defining the boundaries of classes and prioritizing data from these boundaries. This method clusters the test data into boundary areas according to their first and second predicted classes and selects samples evenly from all boundary areas.

Kim et al. [50] proposed Likelihood-based Surprise Adequacy (LSA) and Distance-based Surprise Adequacy (DSA) metrics for measuring the adequacy of a DNN test suite using the surprise effect of each test input. The surprise effect is calculated according to how far the test input is from the training dataset. Byun et al. [27] used these surprise definitions and proposed surprise-based test input prioritization, which gives higher priority to test inputs with high surprise values. LSA calculates the estimated density of the neuron activation values belonging to a test input with respect to the activation trace of the training dataset using Kernel Density Estimation. Activation trace is defined as the vector of activation values of neurons in a selected layer for an input. DSA uses the Euclidean distance to compare the activation trace of the test input with the activation trace of the training dataset to calculate how different the test input is from the training data set.

As an alternative strategy, Zhang et al. [51] and Yan et al. [26] utilized neuron activation values to prioritize test data.

Empirical evaluations are also carried out to compare the effectiveness of test selection metrics. Ma et al. [24] compared a set of uncertainty metrics with the previously proposed DNN test coverage metrics and concluded that uncertainty metrics are better in prioritizing the test data in terms of identifying the ones that the DNN model will misclassify. Their experiments utilized the Monte Carlo (MC) Dropout Bayesian [52] approximation to estimate the uncertainty. Shi et al. [30] performed the most comprehensive empirical study so far to evaluate the test selection metrics with the objective of their fault-revealing capability. They compared 11 test case prioritization metrics derived from the model's prediction probability and categorized them into four groups: information surprise adequacy, confidence dispersion, mutation uncertainty, and mutation rate. In a recent study with a distribution-aware point of view, Hue et al. [23] compared the effectiveness of test prioritization metrics on test datasets that were created with a mixture of different ratios of out-of-distribution test data and in-distribution test data.

In test selection studies, new test data were generated and added to the original test dataset for use in experiments to assess the effectiveness of the test selection strategy. While adversarial attacks were mainly used to generate new test data, image transformations were also employed in some studies.

Differently from previous test prioritization studies, we use uncertainty metrics with deep ensembles and variational inference with flipout methods. In addition, our methodology eliminates the OOD data from the test dataset and then prioritizes the fault-revealing test data.

E. DL EXPLAINABILITY

With the increasing complexity of DL models, their decision-making processes became less understandable and this trend highlighted the need for explainability methods. To understand the decisions of DNNs, they can be designed and trained to produce an explanation along with their output or post-hoc explanation methods can be used on a trained DNN model without making any modification to its architecture or training process [53]. Various post-hoc explainability approaches have emerged in the last decade, including feature attribution based techniques [36], [54], [55], [56], [57], example-based techniques [58], [59] and rule-based techniques [60], [61]. The goal of feature attribution based methods is to determine the significance or contribution of each individual input feature (such as the words in a text or the pixels in an image) to the model's predictions. Gradient-based feature attribution methods, use the gradients obtained during the backpropagation process to identify which features in the input affect the prediction of the model [36], [37], [57]. However, perturbation-based approaches apply perturbations to particular input regions to determine how they affect the output, such as adding blur, noise or totally blocking a region [54], [62], [63].

With the advancement in the development of explainability methods, utilization of these methods to understand the cause of failures in DL models is explored in academia. Especially for the natural language processing problem, studies using post-hoc explainability methods or using self-explainable models have been conducted [64]. In these studies, mostly Naive Bayes models or Logistic Regression models were used. The use of explainability methods in the image classification problem is studied by Adebayo et al. [65] using an 8-layer convolutional neural network. Their study was to examine the effectiveness of visual post-hoc model explanations in diagnosing model errors and debugging. The bugs were categorized into three types: data contamination bugs, model contamination bugs, and test-time contamination bugs. Overall, their findings indicate that feature attribution methods can effectively identify the spatial spurious correlation bug but do not provide conclusive assistance in distinguishing mislabeled examples from normal ones. Regarding model contamination, they observed that certain feature attributions remain invariant to the parameters of the higher layers in deep learning models, suggesting limited effectiveness in diagnosing model contamination bugs. Furthermore, they discovered that attributions for out-of-domain inputs closely resemble attributions obtained from an in-domain model. In contrast to this study, we concentrated on instances where the model was unable to generate accurate predictions because there were not sufficient data that contain specific features.

III. PRELIMINARIES

Bayesian Neural Network (BNN) is a class of neural networks where weights for each layer are represented as distributions instead of concrete values. The variance in the predictions of the BNN model for the same test input with several samplings from these weight distributions is used to measure the uncertainty.

The BNN is initialized with a prior distribution $p(w)$, and the posterior weight distribution is estimated using Bayesian inference in the training process. The training process aims to infer posterior distribution $p(w|X, Y)$. The training dataset is represented with $X = \{x_1, x_2, \ldots x_N\}$ as the input data, and $Y = \{y_1, y_2, \ldots y_N\}$ as the corresponding output data.

$$p(w|X, Y) = \frac{p(Y|X, w)p(w)}{p(Y|X)} \tag{2}$$

However, training of BNN is computationally hard to implement for networks with deep layers. For this reason, approximation methods are used for BNNs. Variational inference (VI) is an approach for learning an approximate posterior distribution over the neural network's weights instead of an exact one. In VI, the objective is to estimate a variational parameter θ that parametrizes a variational distribution $q_\theta(w)$, which is close to the exact posterior distribution. For this purpose, the minimization of Kullback-Leibler (KL) [66] divergence between the approximate and exact posterior distribution is aimed. The KL divergence is used to measure the closeness of two distributions:

$$KL(q_\theta(w)||p(w|X, Y)) = \int q_\theta(w) log \frac{q_\theta(w)}{p(w|X, Y)} \tag{3}$$

However, the KL divergence cannot be minimized directly since it depends on the exact value of posterior distribution, so evidence lower bound (ELBO) [67], which is equal to it up to a constant value, is used. Maximization of evidence lower bound (ELBO) has the same effect of minimizing KL divergence:

$$L_{VI}(\theta) = \int q_\theta(w) log p(Y|X, w) dw - KL(q_\theta(w)||p(w)) \tag{4}$$

During the training of BNN, stochastic sampling is made from weight parameters of $q_\theta(w)$ and $p(w)$ distributions, with the objective of minimizing the KL-divergence between these two distributions. The $q_\theta(w)$ is defined as a data distribution $q(\theta) = N(\mu, \sigma)$ with having mean μ and variance σ. In weight perturbation-based variational inference methods, sampling from the distribution is made using a weight perturbation variable ϵ at each sampling with $w = \mu + \sigma \odot \epsilon$.

The Flipout [35] method samples weight perturbations for each sample in the training minibatch with a flip-the-coin effect $w = \mu \pm \sigma \odot \epsilon$. With flip the coin, the sign of perturbation that will be used in weight perturbation is decided stochastically. This flipout process yields decorrelated gradients and low variance in the gradient descent algorithm that results in achieving faster convergence during training.

Deep Ensembles are proposed as an alternative to BNNs to estimate the uncertainty of the DNN models [34]. They are easy to implement and require less hyperparameter tuning. In the Deep Ensemble method, multiple DNNs having the same architecture are trained separately. Each DNN learns a distinct set of hyperparameters by conducting the training with randomly initialized model weights. After the models have been trained, the uncertainty metric values for test data are calculated. For this purpose, each test data instance is fed to the trained models, and then metric values calculated from each model are ensembled.

IV. METHODOLOGY

Fig. 1 provides a high-level overview of our testing framework. The phases in this approach are essential in testing DNN models rigorously. Each phase of this framework, we believe, may be applied to any kind of DNN model. The methods used in each phase can be chosen based on the properties of the DNN model and its input data.

The goal of the Test Input Generation phase is to create a diverse and comprehensive test dataset. Since the original test dataset used in model development is limited, enhancing this dataset is needed to increase trust in the model. The test dataset can be enlarged by collecting test data from the real world or by generating them synthetically. Although the generation of realistic test data synthetically is challenging, especially for high dimensional data such as image data, their automated behavior makes them good choices. The original test data can be used as seed inputs to data generation methods to create new data. According to selected test generation methods, test data for representing real-world scenarios or malicious intention attacks can be generated, and the reliability and robustness of DNN models can be evaluated.

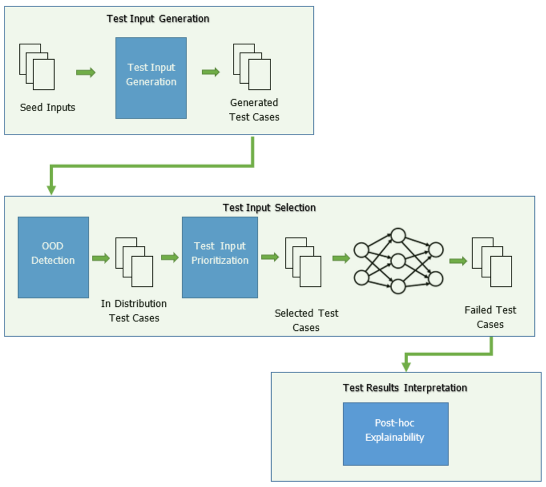


FIGURE 1. Overview of our study.

With limited resources and time, testing the DNN model with all generated test inputs may not be feasible since labeling all the data is a costly process. In the Test Selection phase, the most critical and representative test inputs that will provide useful insights into the DNN model's weaknesses are identified.

We consider test selection in two stages: 1) identifying a subset of data from an unlabeled dataset and 2) prioritizing the most useful data within that subset for testing.

When a DNN produces an incorrect prediction for an input, the question arises as to whether the error is due to a weakness of the DNN model or whether the input data are invalid or from a different data distribution than the training data. It is important to differentiate between the different sources of incorrect predictions. The distribution of training data determines what data can or cannot be correctly predicted by the DNN. By distinguishing between the mispredictions caused by in-distribution or out-of-distribution data, developers can gain insights into the DNN's performance and identify specific areas for improvement.

With our proposed framework, our aim is primarily to focus on the reliability of the DNN model on the data it is intended to work on. These are data coming from similar data distribution with the training dataset, that is, in-distribution data. The performance of the DNN model may be tested for different levels of data distribution shifts after testing with the in-distribution test dataset and trusting the reliability of the DNN models on this dataset. This approach is analogous to performing functional testing on a traditional software program first, followed by robustness tests. It can be misleading to test the DNN model with a dataset that does not resemble the training data without being sure of the model's reliability with these data.

Therefore, in the first stage of test selection, we employ Out-of-Distribution (OOD) data detection method. We apply out-of-distribution detection to test data and only use the ones not classified as OOD. By doing so, we aim to evaluate the DNN model using a more targeted test dataset. As a result, unnecessary effort is not spent in labeling and investigating incorrect predictions for unintended data.

After eliminating OOD data instances from the generated test dataset, we prioritize them by giving higher priority to test data that are more likely to be incorrectly predicted by the DNN model under test. This prioritization will allow labeling efforts to be spent first on test data with a higher priority and then, if necessary, on test data with a lower priority. Additionally, testing the model with prioritized test data will help to reduce the time required to identify the model's weaknesses. The final step is the analysis of the incorrect predictions of the model under test. Because DNNs are black boxes, interpreting their behavior is difficult. Understanding why a DNN made a specific prediction can be difficult with complex DNN models. At this phase, to gain insight into the cause of the model's misprediction for the test data, we use post-hoc methods in DL explainability.

V. EXPERIMENTS

We conducted experiments to evaluate the proposed methodology. We have chosen image classification as the use case and used two well-known image classification datasets with two distinct DNN models for these datasets in our experiments (Section V-A). We implemented each phase of the framework for the selected use case (Section V-B) and reported the evaluation results (Section V-C).

A. TEST DATASETS AND DL MODELS

We selected two popular image classification datasets, MNIST [68] and CIFAR-10 [69]. MNIST is a handwritten

TABLE 1. Datasets and DNN models.

Dataset	Original Training Data Size	Original Test Data Size	Class #	DNN Model	Layers	Parameters	Accuracy for Original Test Data(%)
MNIST	60000	10000	10	LeNet-5	7	61.7 K	99.15
				LeNet-1	5	3246	98.2
CIFAR-10	50000	10000	10	ResNet-32	33	0.46 M	92.10
				VGG-16	21	20.5 M	90.33

[*] Layers include convolutional, pooling, and fully-connected layers.

TABLE 2. Test data generation methods.

Test Data Generation Type	Method	Applied Datasets
Image Transformation	Blur0% - 10% random blurring	MNIST/CIFAR-10
	Brightness0% - 50% random brightness change	MNIST/CIFAR-10
	Translate0% - 30% random translation	MNIST/CIFAR-10
	Rotate0% - 10% random rotation	MNIST/CIFAR-10
Adversarial Attacks	Fast Gradient Sign Method (FGSM)	MNIST/CIFAR-10
	Carlini & Wagner Attacks (C&W)	MNIST/CIFAR-10
	Projected Gradient Descent	MNIST/CIFAR-10
Generative Models	Generative Adversarial Network(Multi-perceptron layers)	MNIST
	Deep Convolutional Generative Adversarial Network	MNIST
	Auxiliary Classifier Generative Adversarial Network	MNIST
	Diffusion Projected GAN	CIFAR-10
	Diffusion StyleGAN2	CIFAR-10

digit dataset with a resolution of 28×28 pixels. Images are labeled from 0 to 9. The dataset contains 60,000 training and 10,000 test images. We selected the LeNet-5 and LeNet-1 [70] networks from the LeNet family as the model under test and trained the model with the whole MNIST training dataset for 25 epochs.

CIFAR-10 is a ten-class image dataset with 50,000 training and 10,000 test images. The images are colored with a resolution of 32×32 pixels. For CIFAR-10, we trained a ResNet-32 [71] model for 200 epochs and a VGG-16 [72] model for 150 epochs and used them as the model under test in the experiments. Real-time data augmentation is applied to training images during training by randomly flipping or shifting the images in horizontal or vertical directions. We implemented the ResNet-32 [71] architecture proposed for CIFAR-10 and the VGG-16 [72] architecture with an input layer accepting data having a size of 32×32x3.

These datasets and models are widely used by the research community and provide a solid foundation for evaluating the efficacy of test selection. The high accuracy of these models makes them good candidates for assessing the fault-revealing capabilities of test selection methods. Since there will be less misclassified data with these models, it will be easier to observe the success of the test selection in the experiments. Table 1 summarizes each studied dataset, its corresponding DNN model, and the accuracy attained for the original test dataset after training.

B. IMPLEMENTATION OF THE FRAMEWORK

This section provides implementation details for each phase of the framework. We selected the methods and strategies used in each phase by considering the properties of image classification models and datasets. Some of the methods used already exist in the literature but have not been used in the context of DNN testing, such as using Deep Ensemble and Variational Inference to prioritize test data or using explainability methods to evaluate test results.

1) TEST INPUT GENERATION IMPLEMENTATION

We employed three types of image generation techniques in the Test Input Generation phase to have a diverse test dataset: Image Transformations, Adversarial Attacks, and Generative Models. These are the most common methods used in the literature for generating test data, and each method has unique properties. The selected image generation methods are listed in Table 2.

In the first group, we use pixel transformations and affine transformations. These transformations are generally used to mimic real-world conditions like changing lighting conditions in the environment or the rotation of a camera in an autonomous vehicle. Pixel transformations change the values of pixels in the image, whereas affine transformations move the pixels of the image. We select translation and rotation as affine transformation and blurring and brightness change as pixel transformation. New test datasets are generated for each type of transformation by applying the transformation to seed inputs that are randomly selected from the original test dataset.

Adversarial attacks are decisive methods that try to fool the DNN models by introducing small, artificially crafted perturbations to images. In this study, the following adversarial attack techniques are employed: Fast Gradient Sign Method (FGSM) [21], Carlini & Wagner (C&W) [73], and Projected Gradient Descent (PGD) [74]. We use Cleverhans

[75] library for applying adversarial attack methods to images.

In the last group of image generation techniques, we use the generative adversarial networks (GANs), which were devised by Goodfellow et al. [76]. We trained three different types of generative models for the MNIST dataset: Generative Adversarial Network (GAN [76]), Auxiliary Classifier GAN (ACGAN [77]), and Deep Convolutional GAN (DCGAN [78]). Then we used their generator networks to generate new synthetic images.

For the CIFAR-10 dataset, since it is more challenging to generate realistic synthetic images and developing a GAN model is not our core focus, we preferred to use pre-trained GAN models. There are several successful studies conducted on this topic. While choosing CIFAR-10 GAN models, we considered the reported Fréchet Inception Distance (FID) and Inception Score values which are associated with the realism of the generated images. In this context, we employed Diffusion Projected GAN and Diffusion StyleGAN2 [79] models to generate CIFAR-10 data.

While generating the test datasets, we aimed to obtain images that are as realistic as possible, and that can represent images under different conditions.

2) TEST INPUT SELECTION IMPLEMENTATION

Test input selection is performed in two stages. At first, OOD data are eliminated from the generated test datasets then they are prioritized according to their uncertainty.

a: OUT OF DISTRIBUTION DETECTION

We employed Variational Autoencoders (VAEs) [80], a type of generative model, for OOD data detection. The objective of generative models is to learn the distribution of the data and generate new data instances from this distribution. For OOD data detection, the VAE model is trained in an unsupervised manner with the training dataset, and it learns the intrinsic properties of the data. Then, if data from a different data distribution than the one that VAE was trained on is fed to the VAE model, it is expected that the reconstruction probability of the model will be lower and allow it to be distinguished as OOD data. Unsupervised training creates ease of use for VAEs and saves labeling effort.

We performed OOD data detection by using a VAE with probabilistic encoder and decoder networks as proposed by An et al. [81]. We train a variational autoencoder with the training dataset, and with this model, we calculate the reconstruction probabilities for the original test dataset first. The reconstruction probability is the probability of generating input data from the distribution parameters that the decoder network produces as output for this input data. Next, we choose a dataset distinct from the one used to train the VAE model as the OOD dataset. The selected OOD dataset is expected to have an identical image size to the training dataset. We then calculate the reconstruction probability values for this OOD dataset. Using the reconstruction probability values of the original test dataset and OOD dataset, a threshold value is empirically determined to differentiate the OOD data from the in-distribution data. The original test dataset and OOD dataset instances are classified as out-of-distribution or in-distribution data according to several threshold values, and the number of correct/incorrect classifications is determined for each threshold value. We use the F1-score metric, which is a harmonic mean of precision and recall, to determine the threshold value that best distinguishes OOD data from in-distribution data.

Then, we performed OOD data detection on generated test data using the threshold value with the highest F1-score value. The data instances classified as OOD are eliminated from the generated test datasets.

b: TEST DATA PRIORITIZATION

We employed uncertainty-based prioritization to identify the fault-revealing inputs in the next stage of Test Selection. The use of uncertainty in test input selection is based on the idea that model predictions are more likely to be incorrect for inputs with higher uncertainty values.

We evaluate three metrics with three methods for measuring the DNN model's uncertainty. In the first method, we calculate the metric values for each test data using only the output of the model under test. In this study, we use the "Model Under Test (MUT)" expression, which is adapted from the "System Under Test (SUT)" a term used in software engineering testing, to represent the DNN model that is evaluated with the test dataset(s). As the second and third methods, we employ Deep Ensemble (DE) and Variational Inference (VI) with Flipout methods to prioritize the test data based on the prediction uncertainty of the model. Deep Ensemble (DE) is a sampling-based method, whereas Variational Inference (VI) is a Bayesian approximation method used to estimate uncertainty. In these two methods, we calculate metric values by aggregating multiple output values for the same test data, unlike the use of the MUT in which the metric values are calculated using only a single output value for each test data. These methods are among the state-of-the-art methods proposed to be successful in uncertainty quantification [41], [45].

For Deep Ensemble, we train several DNN models having the same architecture as the tested model, initializing their parameters with different initial values. After training the DNN models, the same test data is fed into all the ensemble models. The uncertainty metrics are calculated from the probability distribution output of each model. Then, the mean of each uncertainty metric is calculated, and this mean value is used in the prioritization of the test data.

For Variational Inference (VI), we utilize Flipout [35], which is a weight perturbation-based method. In Tensorflow 2, Flipout is implemented as a DNN layer and provided by the Tensorflow Probability Library. In our study, we have implemented Flipout with the use of this library. Following the works [41], [82], we created new models with Flipout layers corresponding to each tested DNN model for Variational Inference. The difference in these new models is the replacement of convolutional and/or dense layers with their Flipout counterparts in TPL. We replaced each dense and convolutional layer with Flipout layers in the LeNet-5 and LeNet-1 models. For ResNet-32, we replaced only the last convolutional layer in each residual block with a Flipout layer, and for VGG-16, we replaced all convolutional layers with Flipout layers.

After training the model with Flipout layers, a Monte Carlo approach is implemented by conducting multiple forward passes for the same input and aggregating the results of these forward passes. During these passes, uncertainty metric values for each test data are calculated, and these values are aggregated in the same way as in the Deep Ensembles method and used in the prioritization of the test data.

Uncertainty metrics are used to provide a quantifiable measure for the uncertainty of the model associated with its prediction. We selected the metrics that are calculated using the prediction probabilities of DNN models for the classes. We utilize Least Confidence, Margin, and Entropy as uncertainty metrics.

Least Confidence: Confidence is the basic metric for determining how challenging an input is for the model. The final layer of a DNN used for classification problems outputs a probability distribution over the classes. These values are the probability scores for each class, with values ranging from 0 to 1.0. The probability scores for all classes add up to 1.0. If the model's prediction probability is close to 1 for a class and close to 0 for other classes, it is said to have high confidence in its prediction. In the worst-case scenario, if the model's probability scores for different classes are nearly equal, then the model is said to be unconfident in its prediction. As a result, maximum likelihood values for classes are used as a measure of confidence for the input; the higher the Least Confidence value, the higher the uncertainty of the model.

$$\text{Least Confidence} = 1 - max(p_i) \qquad p_i : \text{predicted probability for class } i \tag{5}$$

Entropy: Entropy, a widely used metric in information theory, is also used to measure the uncertainty of DNN models. In information theory, entropy represents the amount of information required to encode a distribution. In the scope of model uncertainty, entropy is adopted to represent the information in the last layer. The higher entropy values signify more uncertainty. Entropy has the highest value when the model is less certain about its prediction and its output probability distribution is uniform (all classes have the same probability score). Entropy is lowest when the model is more certain about its prediction and the model's output probability distribution is peaked (the probability score of all classes except one is 0, and this class has a probability score of 1.0).

$$\text{Entropy} = -\sum_i p_i log p_i \qquad p_i : \text{predicted probability for class } i \tag{6}$$

Margin: Margin is the difference between the probabilities of the first and second most probable classes. A higher margin signifies that there is a clear separation between the highest prediction probability and the second one. If the probability values of the two classes with the highest probability scores are close to each other, it means the model is having difficulty making a decision between these two classes. In this case, the margin value will be low as well. The smaller margin signifies more uncertainty.

$$\text{Margin} = p_k - p_j \qquad k : \text{class having maximum probability}, \quad j : \text{second most probable class} \tag{7}$$

For each generated test dataset, all metric values are calculated employing the three methods. Test data with higher uncertainty score values are given higher priority, and test datasets are ordered according to these priorities. As a result, with the use of three different metrics in three different methods, we have datasets sorted according to nine test prioritization strategies.

We compared these prioritization strategies with the state-of-art test selection approaches Likelihood-based Surprise Adequacy (LSA), Distance-based Surprise Adequacy (DSA) [27], DeepGini [28], and multiple-boundary clustering and prioritization (MCP) [29]. These approaches have been frequently employed for comparison in DNN test data prioritization studies.

To create the activation traces for both DSA and LSA, we used the activation values of neurons in the layer before the last layer of the DNNs.

Additionally, to evaluate the success of the uncertainty-based test selection, test datasets are sorted by random selection, and experiment results gathered with random ordering are used as the minimum performance baseline.

Random: The test dataset is sorted by drawing test data at random from the test dataset. Each test data has an equal probability of being selected.

Then, we evaluate these test data prioritization strategies' capability in identifying fault-revealing test data. The success of test data prioritization in terms of finding test data instances that cause misprediction is assessed in three ways:

Average Percentage of Fault Detected Evaluation: The Average Percentage of Fault Detected (APFD) value is used in testing to measure the fault-revealing capability of test selection methods. Let T be a test dataset with n test samples. The model makes m incorrect predictions for this dataset, and w_i is the order of test data that causes the i^{th} misprediction. The score of APFD is defined as follows:

$$\text{APFD} = 1 - \frac{\sum_{i=1}^{m} w_i}{nm} + \frac{1}{2n} \tag{8}$$

APFD metric is used to compare the fault detection speed of test selection methods and metrics [26], [28] [83]. The range of APFD is between 0 and 1. Higher APFD values indicate faster fault detection rates.

Fault Detection Ratio Evaluation: The fault detection ratio is another performance metric that measures the success of test selection criteria. When test data are randomly selected, the percentage of mispredictions is linear with the percentage of test data selected. The test selection metric is considered to be more successful when fewer test data are required to identify all mispredictions.

Statistical Correlation between Misprediction and Metrics: The statistical correlation between the mispredictions for the selected test data and test selection metric values is another indicator of the success of test data prioritization. We conduct a Point-Biserial correlation and Spearman correlation tests to evaluate the correlation between the metric values for test data and the correctness of predictions. In order to measure correlation, predictions of the model are encoded as 0 if the model makes an incorrect prediction and 1 otherwise. The higher correlation values indicate the more successful metric in fault detection.

Ratio of Area Under Curve (RAUC): In ideal, the prioritization techniques will assign higher scores to all test data that are incorrectly predicted by the tested DNN model than to test data that are correctly predicted. As a result, we would ideally expect a linear relationship (y=x for x <= total number of incorrectly predicted test data) on a graph where the x-axis represents the number of prioritized test data, and the y-axis represents the number of incorrectly predicted test data in prioritized test data. In practice, this graph may not be linear for a particular test prioritization strategy. RAUC is the ratio of the area under the prioritization technique's curve to the area under the ideal curve [27], [84] [85]. RAUC is calculated based on the budget for prioritized test data. For example, $RAUC_{100}$ is the RAUC value when the first 100 prioritized test data are used in the performance evaluation of the prioritization technique. RAUC has a value between 0 and 1, and larger is better. In our study, we used $RAUC_{300}$, $RAUC_{500}$, $RAUC_{1000}$, and $RAUC_{all}$. $RAUC_{all}$ represents the RAUC value when all test data is used.

3) TEST RESULTS INTERPRETATION IMPLEMENTATION

We employed visualization techniques as a post-hoc DNN explainability method. These techniques highlight the regions of the input that are most influential in the prediction of the DNN model. These visual representations are easy for humans to understand and interpret. Selected visualization techniques are model-agnostic techniques which means that they can be applied to any pre-trained model with some restrictions only. These methods also do not require access to training data or retraining of the model.

The visualization techniques that are employed are as follows:

Grad-Cam [36]: This method is proposed for CNNs and requires the model's final layer activation function to be a differentiable function. The outputs of the DNN model's last convolutional layer are used as feature maps of the input, which are assumed to be high-level representations of the image. The gradients of the model's highest probability score (probability score of the predicted class) with respect to the feature maps are then computed. The gradients indicate how changes in the feature maps affect the probability score of the predicted class. Then, the relative importance of each feature map is determined by taking the global average pooling of the gradients. The class activation map (CAM) is the weighted sum of feature maps with weights based on the importance values. CAM is used to generate a heatmap, which is overlaid on the original input image. The heatmap shows the regions that have the highest influence on the model's prediction.

Grad-Cam++ [37]: This method is an enhancement to Grad-Cam, which better localizes the objects in the input image. Grad-Cam++ uses a weighted average of positive gradients instead of a global average pooling used in Grad-Cam during the calculation of CAM. This approach also presents a solution for the shortcomings of Grad-Cam when there are multiple occurrences of objects belonging to the same class in the same image.

Score-Cam [38]: This method is a perturbation-based algorithm. The feature maps of the input image are produced in the same way as the Grad-Cam. However, unlike Grad-Cam, gradients are not calculated to construct the CAM. Instead, the CAM is calculated from the weighted sum of feature maps. Using the feature maps as masks on the input image, perturbations on the masked regions of the image are made. Then, the increase in the confidence score is calculated, which is defined as the difference between the probability score of the perturbated image and the original image for the predicted class. The weights of feature maps are determined according to the increase in the confidence score. The CAM is used to generate a heatmap, and the heatmap is overlaid on the input image. This method eliminates the need for gradients but necessitates a forward pass for calculating the weight of each feature map.

C. EXPERIMENT RESULTS

We applied each image generation method given in Section V-B1 independently and produced an additional 1000 (one thousand) images with each method. For this purpose, we have selected 1000 seed images with equal image count for each class from the original test datasets randomly and used them as inputs for Image Transformation and Adversarial Attacks methods. In our experiments, we adjusted the parameters used for Image Transformation methods such that the images generated by transformation on the seed images do not need to be manually labeled, and the same label with the seed image can be used. Also, since the adversarial attacks make small perturbations to seed images, although they deceive the DNN models, the labels of the perturbated images do not change. Therefore, we have assigned the same labels to the perturbated images as the originals. The parameters for Image Transformation and Adversarial Attacks are given in Appendix A. Additionally,

TABLE 3. Details of generated datasets.

ID	Generated Test Dataset	Dataset	Data Count	Applied Method
1	Image Transformation	CIFAR-10	1000	Original
			1000	+Blur
			1000	+Brightness
			1000	+Translate
			1000	+Rotate
2	Adversarial Attacks	CIFAR-10	1000	Original
			1000	+Fast Gradient Sign Method (FGSM)
			1000	+Carlini & Wagner Attacks (C&W)
			1000	+Projected Gradient Descent
3	Generative Models	CIFAR-10	1000	Original
			1000	Diffusion Projected GAN
			1000	Diffusion StyleGAN2
4	Image Transformation	MNIST	1000	Original
			1000	+Blur
			1000	+Brightness
			1000	+Translate
			1000	+Rotate
5	Adversarial Attacks	MNIST	1000	Original
			1000	+Fast Gradient Sign Method (FGSM)
			1000	+Carlini & Wagner Attacks (C&W)
			1000	+Projected Gradient Descent
6	Generative Models	MNIST	1000	Original
			1000	Generative Adversarial Network(Multi-perceptron layers)
			1000	Deep Convolutional Generative Adversarial Network
			1000	Auxiliary Classifier Generative Adversarial Network

we used the generator networks of each GAN selected and generated 1000 images from each of them. We manually labeled the images generated using GANs for both MNIST and CIFAR-10. The generated test datasets are listed in Table 3, where "Original" refers to seed data captured from the original test dataset. The "+" symbol in front of the methods in the table indicates that they have been applied on the Original test dataset.

Then, we identified the out-of-distribution images in the generated test datasets as the first step of test selection. For this purpose, we designed two separate VAE networks for MNIST and CIFAR-10 and trained them using the original training datasets. In this way, we aimed for VAEs to learn the distribution of the training datasets. The structure of VAEs is given in Table 4.

We selected the Fashion MNIST [86] dataset as the OOD dataset for MNIST. The Fashion MNIST dataset consists of grayscale images with the same image size as MNIST. Since Fashion MNIST contains clothing images instead of handwritten digit images, its data distribution differs from that of MNIST. Thus, the VAE trained with the MNIST training dataset will produce lower reconstruction probabilities for images from Fashion MNIST. In the same way, we selected the SVHN [87] dataset as the OOD dataset for the CIFAR-10 dataset.

Using VAEs, we obtained the reconstruction probabilities of the images in the original test datasets and OOD datasets. As described in Section V-B2a, the reconstruction probability threshold values that best separate OOD datasets from in-distribution datasets are determined. The best F1-Score values for threshold values are given in Table 5 for MNIST and CIFAR-10.

TABLE 4. Variational autoencoder models used for OOD data detection.

Model Name	Model Structure	Training Epochs
MNIST VAE	Input Layer - 3 Dense - Lambda - 3 Dense	50
CIFAR-10 VAE	Input Layer - 4 Conv2D - Flatten - 3 Dense - Lambda - 2 Dense - 3 Conv2DTranspose - Conv2D - Flatten - 2 Dense	200

TABLE 5. Best F1-score values for VAEs.

Dataset	True Positive	False Positive	F1-Score
MNIST	99.9%	0.29%	0.99
CIFAR-10	83.52%	17.62%	0.83

RQ1. Which test generation method is more likely to generate OOD data?

We classified the generated test images as OOD and in-distribution data according to the chosen reconstruction probability threshold values and respective reconstruction probability of each image in the datasets. Fig. 2 depicts the percentage of OOD data for each dataset.

For the MNIST and CIFAR-10, nearly all images generated by the FGSM and PGD adversarial attack methods are identified as OOD data. C&W attack generates adversarial images with smaller perturbations than FGSM and PGD; consequently, images generated by C&W attack may resemble the distribution of training data. Images generated by the C&W attack are determined to be mostly in-distribution for both datasets (83.5% in CIFAR-10 and 99% in MNIST).

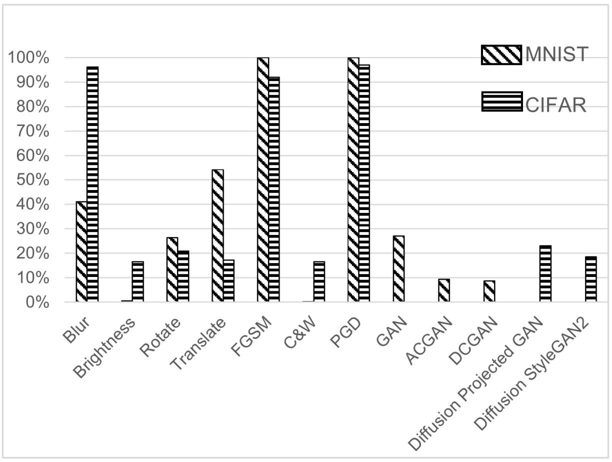


FIGURE 2. OOD data percentages in generated test datasets.

TABLE 6. Accuracy values of models for generated test datasets.

		MNIST	
Model	Generative Models	Image Transformation	Adversarial Attacks
LeNet-5	91.8%	92.4%	41.6%
LeNet-1	90.4%	88.0%	7.0%
		CIFAR-10	
Model	Generative Models	Image Transformation	Adversarial Attacks
ResNet-32	79.4%	77.3%	13.5%
VGG-16	79.8%	76.4%	7.2%

Although the same transformation settings are used for test data in both datasets with Image Transformation methods, as shown in Table 2, the percentage of the OOD data in generated datasets differs between the MNIST and CIFAR-10 datasets. Following adversarial attack methods, images generated with the Translate method have the highest percentage of OOD data for the MNIST, with 54.2%. Translate is an affine transformation that shifts pixels without changing their values. The high OOD ratio of Translate may be due to the fact that the majority of digits in the training images of MNIST are located in the center of the image, whereas Translate generates images with digits located in other positions. In contrast, Blurring is the image transformation technique that produces the most OOD data for the CIFAR-10 dataset. The CIFAR-10 Dataset is colored, and Blurring alters the pixel values in all image channels. We believe that because this transformation is applied randomly to each pixel, it causes the image to deviate from the training data distribution and be classified as OOD data.

In both CIFAR-10 and MNIST datasets, images generated synthetically by Generative Models do not produce a high number of OOD data. During their training process, generative models learn the training data's features and generate images, resulting in in-distribution images.

Result for RQ1: Observations regarding RQ1 indicate that different image generation methods produced a substantial amount of OOD data in the generated test datasets. In particular, adversarial attacks generate a large amount of OOD data, whereas the amount of OOD data generated by Generative Models is moderate. Among the image transformation methods, the test generation method that generates the most OOD data depends on the image dataset properties; there is no single answer.

RQ2: How effective are uncertainty-based test selection metrics for prioritizing the fault-revealing data instances in datasets that are generated with different techniques and identified as in-distribution data?

We eliminated OOD data from generated test images and created three test data groups based on the test generation technique. In Table 6, the accuracy of the respective models under test for the generated test datasets after OOD data elimination is given. Although the OOD detection method identifies most of the images generated by C&W adversarial attacks as in-distribution data, it is clear that these images deceived the DNN models, resulting in a very low level of accuracy of 7% in LeNet-1 and VGG-16.

An additional subset of 1000 images from the original test data are added to each test data group in order to better observe the choice of test selection metrics between data that the model would misclassify and data that would be correctly classified. Since the models had a very high accuracy rate for the original test data, these additional images were considered as a dataset that the model would not have difficulty classifying correctly. In each test data group, we analyzed the performance of test selection methods and metrics in terms of fault identification. For this purpose, the uncertainty metric values of each test dataset are calculated using the MUT, DE models, and VI models separately. Previous research on uncertainty quantification [41] has demonstrated that a sample size of five provides good results for both the DE and VI methods and that increasing the sample size has diminishing positive effects. Therefore, for the DE method, experiments were conducted with a five-member ensemble, and for the VI method, tests were conducted by collecting five samplings from the VI model's pass-throughs.

When calculating metric values with MUT, the predicted probability values of the model for test data are used. For the DE method, each test data is fed into DE models, and the outputs of the models are aggregated and used to determine the value of each metric. Similarly, each test data is fed to the VI model five times, and the output values are aggregated and used to calculate metric values. Then, the test datasets are ordered according to the metric values and fed to the model under test according to this order. Test images are categorized as misclassified or correctly classified based on their ground labels and prediction of the model under test.

We evaluate the fault-revealing capability of the test prioritization strategies based on the Average Percentage

TABLE 7. Average percentage of fault detected values for generated test datasets.

	Test Data Generation Methods					
	Generative Models		Image Transformation		Adversarial Attacks	
Test Prioritization Strategy	LeNet-5	LeNet-1	LeNet-5	LeNet-1	LeNet-5	LeNet-1
Random	0.48565	0.50119	0.53431	0.51787	0.48124	0.49215
MUT - Entropy	0.89260	0.88980	0.91006	0.85751	**0.84513**	0.74349
MUT - Confidence	0.88872	0.88942	0.91028	0.86290	**0.84361**	**0.74565**
MUT - Margin	0.88880	0.88862	0.91002	0.86446	0.84348	**0.74569**
DE - Entropy	**0.90494**	**0.90334**	**0.91810**	**0.88684**	0.80568	0.72215
DE - Confidence	**0.90353**	**0.90368**	**0.91857**	**0.89302**	0.80317	0.72276
DE - Margin	**0.90296**	**0.90308**	**0.91765**	0.89160	0.80555	0.72302
VI - Entropy	0.86774	0.86448	0.87171	0.80019	0.77053	0.67599
VI - Confidence	0.84960	0.86373	0.87369	0.80171	0.76789	0.67567
VI - Margin	0.86066	0.86264	0.87274	0.80130	0.76783	0.67591
LSA	0.87991	0.81456	0.90263	0.85782	0.82452	0.71029
DSA	0.89566	0.85043	0.91233	**0.89281**	0.84250	**0.75189**
DeepGini	0.88877	0.88947	0.91034	0.86145	**0.84465**	0.74443
MCP	0.87212	0.86957	0.87997	0.84186	0.73855	0.62395

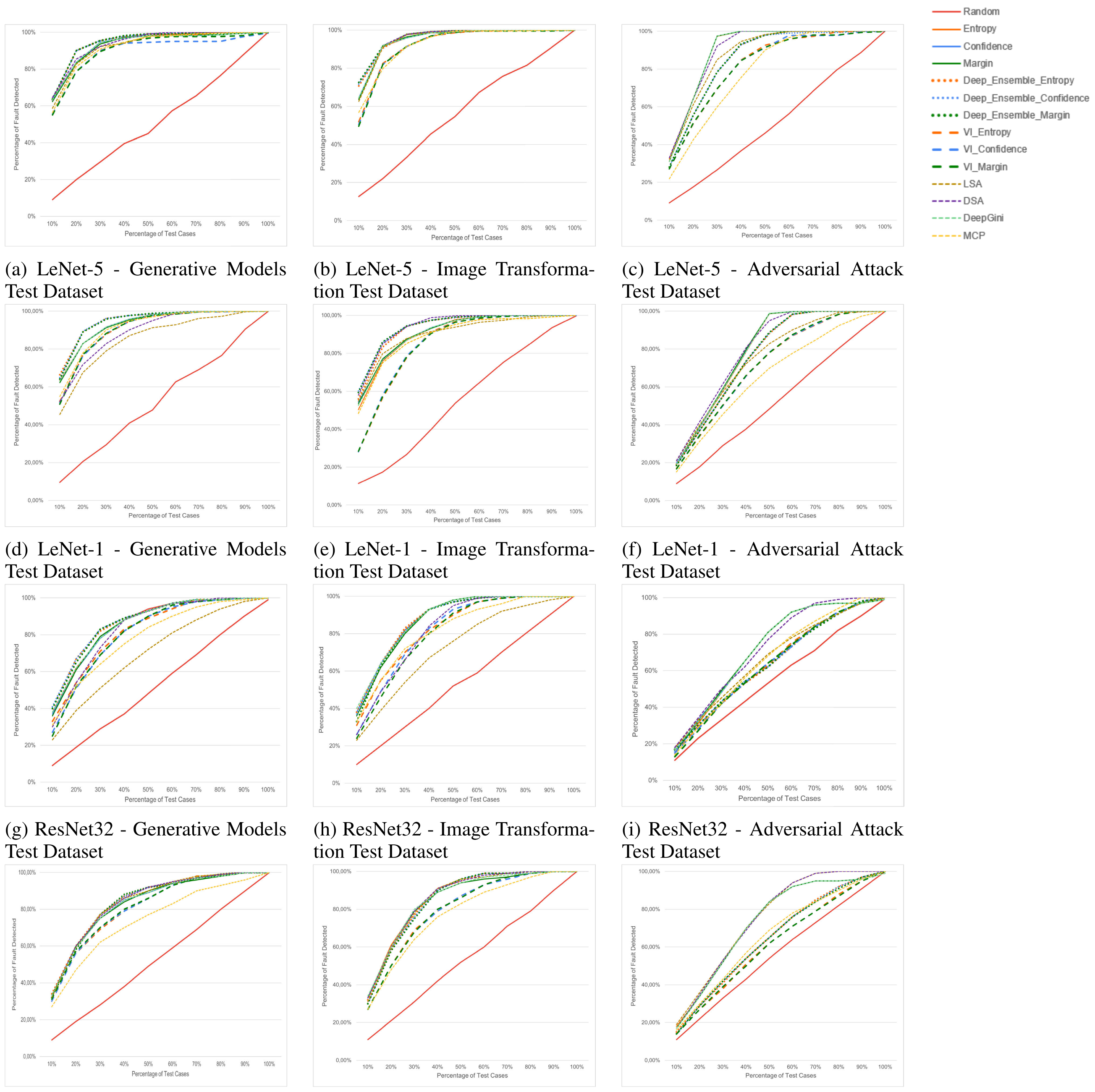
(a) MNIST

	Test Data Generation Methods					
	Generative Models		Image Transformation		Adversarial Attacks	
Test Prioritization Strategy	ResNet-32	VGG16	ResNet-32	VGG16	ResNet-32	VGG16
Random	0.49061	0.49142	0.50166	0.50730	0.51963	0.52276
MUT - Entropy	0.81193	0.78317	0.82685	0.79941	**0.67568**	**0.68741**
MUT - Confidence	0.81012	0.78321	0.82524	0.79881	0.67481	0.68659
MUT - Margin	0.80750	0.78334	0.82275	0.79839	0.67383	0.68634
DE - Entropy	**0.82078**	**0.79434**	**0.82992**	**0.80229**	0.60304	0.60578
DE - Confidence	**0.82286**	**0.79235**	**0.82695**	0.80055	0.60330	0.60497
DE - Margin	**0.82045**	**0.79180**	0.81849	0.79853	0.60250	0.60493
VI - Entropy	0.77477	0.76601	0.77590	0.75195	0.60219	0.57658
VI - Confidence	0.76287	0.76570	0.76942	0.75134	0.59932	0.57666
VI - Margin	0.76269	0.76580	0.75475	0.75059	0.59778	0.57642
LSA	0.66072	0.78717	0.67856	**0.80361**	0.62552	**0.70690**
DSA	0.78548	0.79076	0.77192	**0.80682**	**0.67683**	**0.70323**
DeepGini	0.81133	0.78326	**0.82840**	0.80161	**0.67635**	0.68704
MCP	0.74190	0.69680	0.77082	0.72965	0.62116	0.61416

(b) CIFAR-10

of Fault-Detected (APFD) values for each test dataset. As described in Section V-B2b, the APFD value represents the fault detection speed of the test dataset. The APFD values for each test data group are given in Table 7. The results of experiments are listed according to the test prioritization strategies, which are identified with method-metric tuple values, e.g., we use MUT-Entropy naming for the experiments conducted using Entropy metric with employing probability output of the model under test or use DE-Confidence for the experiments conducted using Least Confidence metric obtained from the outputs of deep ensemble models. The best three values for each dataset are shown in bold. The closer to 1.0 the APFD values are, the more successful the prioritization of fault-revealing test data. APFD values for CIFAR-10 are smaller than those for the MNIST dataset; likewise, the accuracy of the model under test for CIFAR-10 is less than for MNIST.

Table 7 also shows a comparison of the techniques with the state-of-the-art test selection approaches in the last four rows. We used LSA, DSA, DeepGini, and MCP approaches to prioritize the generated test datasets and incorporated their results for the performance comparison. Table 7 shows that except for the adversarial attack test datasets, the DE method yields high APFD values ranging from 0.887 to 0.919 for MNIST and 0.792 to 0.830 for CIFAR-10. However, for adversarial attack test datasets, metrics acquired using the MUT, DeepGini, LSA, and DSA produce the best results. The APFD values for MUT-Entropy, MUT-Confidence, and MUT-Margin are very close to each other for the adversarial attack test datasets, with a difference of less than 0.007. The adversarial attack test datasets are generated using the MUT as input to attack methods, so the generated images are specialized according to the parameter values of the MUT to fool the network. Therefore, these images are unlikely to deceive the models used in DE or VI-F methods since they have different parameter values, and the uncertainty metric values acquired with these methods do not provide adequate guidance for fault detection. To measure the uncertainty, DeepGini uses a formula similar to MUT-Entropy; both employ the probability score of each class in the output. As expected, they have similar APFD values, with an average difference of 0.016. DSA has APFD values that are among the top three APFD values in test datasets created with adversarial attacks. From this, we can judge that, despite the



(a) LeNet-5 - Generative Models Test Dataset

(b) LeNet-5 - Image Transformation Test Dataset

(c) LeNet-5 - Adversarial Attack Test Dataset

(d) LeNet-1 - Generative Models Test Dataset

(e) LeNet-1 - Image Transformation Test Dataset

(f) LeNet-1 - Adversarial Attack Test Dataset

(g) ResNet32 - Generative Models Test Dataset

(h) ResNet32 - Image Transformation Test Dataset

(i) ResNet32 - Adversarial Attack Test Dataset

(j) VGG16 - Generative Models Test Dataset

(k) VGG16 - Image Transformation Test Dataset

(l) VGG16 - Adversarial Attack Test Dataset

FIGURE 3. Fault detection ratios according to the size of generated test datasets.

small perturbations generated by the adversarial attacks, the activation traces of the corresponding data differ from the activation traces of the training data, and this is detected by DSA.

Fig. 3 shows the Fault Detection Ratio values for each test prioritization strategy relative to the test dataset size. The generated test datasets are sorted by metric values and divided into ten bins, with 10 percent of the combined test data contained in each bin. The first bin contains test data instances with the highest priority, while the last bin contains images with the lowest priority. As the size of the test dataset increases from 10% to 100%, so does the ratio of detected faults. The prioritization metric that assigns high-priority values to the test data that will be misclassified by the DNN model will have a higher fault detection ratio for the same size of test data.

The detailed percentage of faults detected for each type of generated test dataset is shown in Fig. 3. It is evident that

TABLE 8. Ratio of area under curve values for generated test datasets (MNIST).

Test Data Generation Methods	Test Prioritization Strategy	$RAUC_{300}$		$RAUC_{500}$		$RAUC_{1000}$		$RAUC_{All}$	
		LeNet-5	LeNet-1	LeNet-5	LeNet-1	LeNet-5	LeNet-1	LeNet-5	LeNet-1
Generative Models	MUT - Entropy	**0.50448**	0.57334	**0.53510**	0.58931	0.65030	0.6823	0.86259	0.86870
	MUT - Confidence	0.50323	0.56855	0.53386	0.58551	0.64964	0.68059	0.86019	0.86806
	MUT - Margin	0.50301	0.56601	0.53292	0.58313	0.64902	0.67963	0.86034	0.86702
	DE - Entropy	0.49643	**0.57721**	**0.53990**	**0.59864**	**0.67647**	**0.70582**	**0.88182**	**0.89042**
	DE - Confidence	0.48624	**0.59223**	0.53452	**0.60391**	**0.67414**	**0.70765**	**0.88080**	**0.89105**
	DE - Margin	0.47313	**0.58049**	0.52801	**0.59534**	**0.67143**	**0.70300**	**0.87995**	**0.88986**
	VI - Entropy	0.38622	0.49216	0.43765	0.49728	0.56964	0.59200	0.82159	0.82568
	VI - Confidence	0.37884	0.4642	0.43272	0.48062	0.56716	0.58413	0.80994	0.82394
	VI - Margin	0.38020	0.44851	0.43310	0.47079	0.56734	0.57880	0.81737	0.82206
	LSA	0.47342	0.50127	0.49618	0.46985	0.61620	0.53343	0.84242	0.75966
	DSA	**0.50964**	0.53685	**0.54561**	0.52376	0.66493	0.59103	0.86692	0.80587
	DeepGini	**0.50441**	0.56847	0.53438	0.58591	0.64986	0.68090	0.86024	0.86816
	MCP	0.43399	0.51757	0.47214	0.5164	0.59053	0.60594	0.82981	0.83423
Image Transformation	MUT - Entropy	0.51541	0.47572	0.54306	0.48114	0.67219	0.57976	0.88733	0.82751
	MUT - Confidence	0.52078	0.55677	0.54642	0.54421	0.67423	0.61714	0.88783	0.83825
	MUT - Margin	0.51479	0.58207	0.54314	0.56495	0.67275	0.62788	0.88736	0.84148
	DE - Entropy	**0.56199**	0.65117	**0.59606**	0.61153	**0.71093**	0.66689	**0.90179**	0.87651
	DE - Confidence	**0.56437**	**0.68653**	**0.59972**	**0.64611**	**0.71326**	**0.70058**	**0.90258**	**0.88826**
	DE - Margin	**0.54622**	0.61540	**0.58720**	0.60433	**0.70659**	**0.68474**	**0.90075**	**0.88510**
	VI - Entropy	0.29076	0.25568	0.35228	0.25807	0.51329	0.35369	0.81927	0.72567
	VI - Confidence	0.33394	0.29407	0.37565	0.28478	0.52419	0.3657	0.82251	0.72879
	VI - Margin	0.34250	0.26373	0.38295	0.27344	0.52869	0.36362	0.82328	0.72825
	LSA	0.40995	**0.68616**	0.47229	**0.61579**	0.62375	0.65507	0.87309	**0.83979**
	DSA	0.52160	**0.71412**	0.54349	**0.65083**	0.67009	**0.69261**	0.89016	0.88676
	DeepGini	0.52440	0.51643	0.54799	0.51815	0.67484	0.60497	0.88798	0.83526
	MCP	0.42154	0.50876	0.45707	0.50222	0.57631	0.57209	0.83636	0.80961
Adversarial Attacks	MUT - Entropy	**0.95622**	0.89497	**0.95619**	0.90063	**0.97180**	0.91456	**0.98582**	0.95360
	MUT - Confidence	0.90093	**0.93863**	0.92872	**0.92758**	0.96310	**0.92614**	**0.98197**	**0.95890**
	MUT - Margin	0.89151	0.83913	0.92573	0.88172	0.96232	0.91713	0.98159	**0.95762**
	DE - Entropy	0.84088	0.92076	0.83553	0.8966	0.84024	0.87479	0.90866	0.91323
	DE - Confidence	0.83500	0.92224	0.83309	0.89504	0.83960	0.87514	0.90629	0.91424
	DE - Margin	0.83175	0.90241	0.83175	0.88783	0.83923	0.87356	0.90824	0.91438
	VI - Entropy	0.79997	0.80397	0.78588	0.80686	0.77046	0.79575	0.85079	0.83491
	VI - Confidence	0.80433	0.76050	0.78753	0.78774	0.77025	0.78978	0.84954	0.83343
	VI - Margin	0.80522	0.75815	0.78805	0.78606	0.77076	0.78878	0.84979	0.83360
	LSA	**0.94048**	**0.99051**	0.92765	**0.97362**	0.91213	**0.92072**	0.94621	0.90679
	DSA	**0.98274**	**0.99567**	**0.97177**	**0.99064**	**0.96509**	**0.97398**	0.98051	**0.97636**
	DeepGini	0.93891	0.90417	**0.94386**	0.91020	**0.96697**	0.92037	**0.98358**	0.95602
	MCP	0.65586	0.71074	0.64871	0.72288	0.65491	0.71586	0.78448	0.75519

when test data are sorted according to uncertainty metrics, faults are detected earlier in all test datasets compared to when data are selected at random. Except for the adversarial attack dataset of CIFAR-10, this difference is very substantial for all datasets. By using the uncertainty metrics with DE, 30% of the test dataset captures more than 95% of the whole faults in test datasets generated by Generative Models and Image Transformations for MNIST. The percentage of test data required to find the equivalent percentage of faults in the CIFAR-10 dataset is 50%. While DE methods have the highest fault detection ratios for test datasets generated by Generative Models and Image Transformations, prioritization with DSA, DeepGini, and metrics acquired with MUT methods have higher fault detection ratios than other methods for datasets generated by Adversarial Attacks. The performances of uncertainty metrics obtained with the VI method and MCP are poorer than those of other approaches. The difference between fault detection rates of prioritization approaches employing VI and the nearest fault detection rate of other approaches changes in the average of 4.03% to 13.06% for MNIST test datasets. In the case of MCP, the difference is between 2.28% and 18.54% on average. The fault detection rate of LSA is also low compared to other methods except for the experiments made with the VGG-16 model. It has the lowest fault detection ratios with the experiments performed with the ResNet-32 model for the test data sets generated by the Generative Model and Image Transformation, and it has more than 10% difference from the nearest approach.

We calculated the Ratio of Area Under Curve value for each generated test dataset with the prioritized test dataset size of 300, 500, and 1000 and with all test data. Tables 8 and 9 show the RAUC values for MNIST and CIFAR-10. The best three values are shown in bold for RAUC values calculated for different test data sizes, with a total of 24 best RAUC values for each generated test

TABLE 9. Ratio of area under curve values for generated test datasets (CIFAR-10).

Test Data Generation Methods	Test Prioritization Strategy	$RAUC_{300}$		$RAUC_{500}$		$RAUC_{1000}$		$RAUC_{All}$	
		ResNet-32	VGG16	ResNet-32	VGG16	ResNet-32	VGG16	ResNet-32	VGG16
Generative Models	MUT - Entropy	**0.70469**	**0.57090**	0.62464	**0.54650**	0.65742	**0.60896**	0.80584	0.76840
	MUT- Confidence	0.65143	0.56611	0.60070	0.54494	0.64736	0.60869	0.80202	0.76844
	MUT- Margin	0.59220	0.56179	0.57120	0.54357	0.63440	0.60850	0.79677	0.76873
	DE - Entropy	**0.69538**	**0.57891**	**0.64434**	**0.55709**	**0.68816**	**0.61823**	**0.82264**	**0.78239**
	DE - Confidence	**0.72336**	0.56057	**0.66066**	0.54130	**0.69741**	0.60831	**0.82674**	**0.77829**
	DE - Margin	0.66925	0.53745	**0.63122**	0.53002	**0.68395**	0.60289	**0.82169**	**0.77662**
	VI - Entropy	0.52144	0.54517	0.49054	0.51647	0.54685	0.57180	0.73580	0.73891
	VI - Confidence	0.48609	0.53826	0.46916	0.51351	0.53918	0.57108	0.73240	0.73868
	VI - Margin	0.39990	0.53988	0.40834	0.51458	0.50368	0.57165	0.71976	0.73933
	LSA	0.34284	0.52424	0.34158	0.51125	0.39624	0.59411	0.58795	0.76920
	DSA	0.48091	0.53669	0.47238	0.52788	0.55466	0.60502	0.75583	0.77541
	DeepGini	0.68988	**0.56904**	0.61943	**0.54601**	0.65476	**0.60903**	0.80467	0.76855
	MCP	0.53949	0.46213	0.49424	0.43387	0.53423	0.48795	0.70212	0.65424
Image Transformation	MUT - Entropy	**0.77794**	**0.73261**	**0.74838**	**0.69651**	**0.69707**	**0.65080**	**0.85544**	**0.82287**
	MUT- Confidence	**0.77223**	**0.70870**	**0.73821**	**0.67986**	**0.68613**	**0.64433**	0.84988	0.82159
	MUT- Margin	0.69429	0.68624	0.68390	0.66833	0.65931	0.64006	0.84151	0.82072
	DE - Entropy	0.71960	0.66930	0.71167	0.64707	0.67873	0.62053	**0.85060**	0.82007
	DE - Confidence	0.73236	0.66225	0.70473	0.63491	0.67271	0.61274	0.84776	0.81688
	DE - Margin	0.64113	0.61319	0.63306	0.59808	0.63238	0.59776	0.83435	0.81262
	VI - Entropy	0.60618	0.62210	0.58474	0.58116	0.55732	0.54455	0.76712	0.74524
	VI - Confidence	0.59000	0.59205	0.54973	0.56163	0.51293	0.53658	0.74340	0.74355
	VI - Margin	0.44009	0.56576	0.43896	0.54674	0.46495	0.53085	0.73417	0.74217
	LSA	0.42427	0.69694	0.41997	0.65626	0.40763	0.62130	0.62942	0.81983
	DSA	0.51067	0.68186	0.49700	0.66221	0.49414	0.63719	0.75101	**0.82616**
	DeepGini	**0.78172**	**0.72522**	**0.75289**	**0.68794**	**0.69716**	**0.64684**	**0.85293**	**0.82206**
	MCP	0.66563	0.57252	0.63887	0.54717	0.59583	0.51491	0.76644	0.71215
Adversarial Attacks	MUT - Entropy	**0.84040**	**0.89818**	**0.80094**	**0.87909**	**0.76300**	**0.86718**	**0.82814**	**0.88951**
	MUT- Confidence	0.74322	0.85400	0.73898	0.85030	0.74663	0.85801	0.82337	0.88673
	MUT- Margin	0.64176	0.80756	0.69565	0.82797	0.73655	0.85185	0.82040	0.88522
	DE - Entropy	0.78554	0.74484	0.75436	0.73008	0.68598	0.70135	0.71418	0.73889
	DE - Confidence	0.79762	0.73411	0.75645	0.72002	0.68533	0.69532	0.71449	0.73661
	DE - Margin	0.76995	0.72494	0.73861	0.71734	0.67791	0.69369	0.71206	0.73630
	VI - Entropy	0.70394	0.70785	0.69490	0.67959	0.65840	0.65558	0.70908	0.69565
	VI - Confidence	0.68532	0.71365	0.67261	0.68139	0.64618	0.65562	0.70250	0.69571
	VI - Margin	0.63791	0.71336	0.63743	0.68105	0.63851	0.65475	0.70104	0.69513
	LSA	**0.84279**	**0.95282**	**0.80208**	**0.93390**	0.73335	**0.89469**	0.75564	**0.91070**
	DSA	**0.83092**	0.85886	**0.81037**	**0.86603**	**0.77381**	**0.86604**	**0.82391**	**0.90036**
	DeepGini	0.82713	**0.88489**	0.78380	0.86566	**0.75986**	0.86262	**0.82777**	0.88819
	MCP	0.68027	0.75943	0.66801	0.76141	0.65863	0.73311	0.72991	0.76104

dataset. The prioritization strategies that employ the DE method achieve the best RAUC values for the test dataset generated by Generative Models for MNIST with a ratio of 19/24 and for CIFAR-10 with a ratio of 17/24. Similarly, for the test dataset generated by Image Transformations for MNIST, the prioritization strategies with DE achieved the highest RAUC value ratio of 18/24. On the contrary, for the CIFAR-10 test dataset that is generated by Image Transformations, the MUT-Entropy, MUT-Confidence, and DeepGini have the best results. The distance-based methods (LSA/DSA) and techniques that use uncertainty metrics with MUT outperform the other prioritization techniques in the case of test datasets generated by Adversarial Attacks.

Additionally, the statistical correlation between misclassifications and metric values is used to evaluate the fault detection capability of test prioritization strategies. For this purpose, we encoded the model's prediction for a given test data as 1 for misclassification and 0 for correct classification of the model under test. The correlation degrees between the metric values of the test dataset and the corresponding classification results are calculated for each metric.

We employed the Point-Biserial correlation, which measures the strength of the relationship between a continuous-level variable and a binary variable since metric values are continuous data, while misclassifications are binary data in our situation. In addition, we also conducted the Spearman correlation, which is used to analyze how effectively a monotonic function describes the relationship between two variables.

Table 10 and 11 show that there is a moderate correlation between the uncertainty metrics obtained using the DE method and the misclassifications for the test datasets generated using Generative Models and Image Transformation techniques. Besides, for the test dataset generated with Adversarial Attacks, the correlation coefficient is highest for MUT-Margin, MUT-Confidence, and DeepGini for MNIST.

TABLE 10. Correlation between test prioritization strategies and misclassifications (MNIST).

	Test Data Generation Methods					
	Generative Models		Image Transformation		Adversarial Attacks	
Test Prioritization Strategy	Biserial	Pearson	Biserial	Pearson	Biserial	Pearson
MUT - Entropy	0.43	0.35	0.49	0.35	0.96	0.78
MUT - Confidence	0.40	0.38	0.47	0.37	**0.96**	**0.84**
MUT - Margin	0.40	0.37	0.46	0.37	**0.97**	**0.83**
DE - Entropy	**0.49**	**0.36**	**0.55**	**0.36**	0.41	0.69
DE - Confidence	**0.47**	**0.38**	**0.55**	**0.37**	0.34	0.72
DE - Margin	**0.47**	**0.37**	**0.55**	**0.37**	0.35	0.71
VI - Entropy	0.33	0.33	0.34	0.32	0.29	0.61
VI - Confidence	0.29	0.36	0.33	0.34	0.24	0.62
VI - Margin	0.28	0.34	0.34	0.33	0.24	0.62
LSA	0.41	0.34	0.31	0.35	0.64	0.73
DSA	0.46	0.35	0.46	0.36	0.83	0.77
DeepGini	0.41	0.38	0.49	0.37	**0.97**	**0.84**
MCP	0.37	0.35	0.42	0.35	0.96	0.77

(a) MNIST - LeNet-5

	Test Data Generation Methods					
	Generative Models		Image Transformation		Adversarial Attacks	
Test Prioritization Strategy	Biserial	Pearson	Biserial	Pearson	Biserial	Pearson
MUT - Entropy	0.55	0.38	0.49	0.40	0.85	0.80
MUT - Confidence	0.53	0.38	0.52	0.40	**0.92**	**0.80**
MUT - Margin	0.53	0.38	0.53	0.40	**0.92**	**0.80**
DE - Entropy	**0.57**	**0.39**	**0.57**	**0.43**	0.62	0.73
DE - Confidence	**0.57**	**0.39**	**0.60**	**0.44**	0.55	0.73
DE - Margin	**0.56**	**0.39**	**0.59**	**0.43**	0.57	0.73
VI - Entropy	0.46	0.36	0.32	0.33	0.36	0.58
VI - Confidence	0.42	0.36	0.30	0.34	0.29	0.58
VI - Margin	0.41	0.35	0.30	0.33	0.28	0.58
LSA	0.36	0.31	0.37	0.40	0.38	0.69
DSA	0.40	0.34	0.54	**0.44**	0.80	0.83
DeepGini	0.55	0.38	0.51	0.40	**0.90**	**0.80**
MCP	0.49	0.38	0.51	0.41	0.89	0.81

(b) MNIST - LeNet-1

In the case of CIFAR-10, MUT-Entropy, LSA, DSA, and DeepGini have the highest correlation coefficient values. The correlation results are in harmony with those derived from APFD values.

Result for RQ2: Uncertainty-based test selection is an effective technique for prioritizing the test data in order to identify fault-revealing inputs. The best prioritization results are obtained when uncertainty metrics are used with the Deep Ensemble method for Image Transformation and Generative Model datasets. For test datasets generated by Adversarial Attacks, metrics gathered from Model Under Test give the best results. Additionally, distance-based methods (LSA/DSA) and DeepGini effectively prioritize test datasets generated by Adversarial Attacks. The VI method has the lowest performance in terms of uncertainty-based test prioritization in prioritizing fault-revealing test data in all datasets.

RQ3: Does the existence of OOD data in the test dataset impact the success of the test selection methods?

We repeated the experiments without removing the OOD data from the generated test datasets to determine whether the same metrics and methods effectively prioritize test datasets that contain OOD data. In these experiments, all generated test data were combined and used as a single dataset. In Table 12, the best APFD values from the experiments are given, and in Fig. 4, the fault detection ratios are displayed. The APFD values of test prioritization methods for test datasets with OOD data are lower than those for test datasets without OOD data. DSA has the highest APFD values for test datasets with OOD data, which indicates that the OOD test data have activation traces different from the training data. The uncertainty metrics employed with the Deep Ensemble and Model Under Test are also efficient ways for prioritizing test data in datasets containing OOD data, similar to datasets in which OOD data have been eliminated.

As Fig. 4 shows, the fault detection rate of uncertainty metrics is decreased compared to datasets where OOD data are removed. This decrease is especially significant for the CIFAR-10 dataset, where the performance of uncertainty metrics deteriorates to a level close to random sampling.

As Table 13 shows, the accuracy of the tested models for test datasets containing OOD data also decreased, from 65.5% to 49.3% for CIFAR-10 (ResNet-32) and from 84.1% to 78.4% for MNIST (LeNet-5). This is due to the fact that the

TABLE 11. Correlation between test prioritization strategies and misclassifications (CIFAR-10).

	Test Data Generation Methods					
	Generative Models		Image Transformation		Adversarial Attacks	
Test Prioritization Strategy	Biserial	Pearson	Biserial	Pearson	Biserial	Pearson
MUT - Entropy	0.53	0.46	**0.61**	**0.55**	**0.53**	**0.56**
MUT - Confidence	0.50	0.46	0.59	0.54	0.46	0.56
MUT - Margin	0.48	0.46	0.58	0.53	0.46	0.56
DE - Entropy	**0.56**	**0.48**	**0.60**	**0.54**	0.30	0.33
DE - Confidence	**0.57**	**0.48**	**0.60**	**0.54**	0.29	0.33
DE - Margin	**0.56**	**0.48**	0.58	0.53	0.28	0.33
VI - Entropy	0.43	0.41	0.48	0.46	0.27	0.33
VI - Confidence	0.37	0.39	0.44	0.45	0.23	0.32
VI - Margin	0.36	0.39	0.39	0.42	0.23	0.31
LSA	0.23	0.24	0.28	0.30	0.37	0.40
DSA	0.43	0.42	0.44	0.45	**0.56**	**0.57**
DeepGini	0.51	0.46	**0.60**	**0.54**	**0.52**	**0.57**
MCP	0.43	0.45	0.53	0.53	0.38	0.56

(a) CIFAR-10 - ResNet32

	Test Data Generation Methods					
	Generative Models		Image Transformation		Adversarial Attacks	
Test Prioritization Strategy	Biserial	Pearson	Biserial	Pearson	Biserial	Pearson
MUT - Entropy	0.36	0.43	0.46	0.51	**0.51**	**0.64**
MUT - Confidence	0.31	0.43	0.41	0.51	0.44	0.64
MUT - Margin	0.30	0.43	0.40	0.51	0.44	0.64
DE - Entropy	**0.47**	**0.44**	**0.55**	**0.52**	0.26	0.36
DE - Confidence	**0.45**	**0.44**	**0.53**	**0.52**	0.24	0.36
DE - Margin	0.44	0.44	0.53	0.51	0.24	0.36
VI - Entropy	0.30	0.40	0.33	0.43	0.13	0.26
VI - Confidence	0.27	0.40	0.29	0.43	0.13	0.26
VI - Margin	0.28	0.40	0.29	0.43	0.13	0.26
LSA	0.43	0.43	**0.53**	**0.52**	**0.61**	**0.71**
DSA	**0.46**	**0.44**	**0.56**	**0.52**	**0.67**	**0.70**
DeepGini	0.34	0.43	0.44	0.51	0.49	0.64
MCP	0.26	0.43	0.36	0.51	0.36	0.64

(b) CIFAR-10 - VGG-16

TABLE 12. Average percentage of fault detected values for generated test dataset including OOD data.

Test Prioritization Strategy	LeNet-5	LeNet-1	ResNet-32	VGG-16
Random	0.50126	0.50031	0.50283	0.50227
MUT - Entropy	0.81596	0.76158	0.59780	0.59575
MUT - Confidence	**0.81683**	0.76998	0.60203	0.59547
MUT - Margin	**0.81716**	0.77179	0.59828	0.59525
DE - Entropy	0.79419	0.77691	**0.64457**	**0.62754**
DE - Confidence	0.79448	**0.78423**	**0.64215**	0.62631
DE - Margin	0.79564	**0.78667**	0.63862	0.62552
VI - Entropy	0.74212	0.70580	0.62723	0.62579
VI - Confidence	0.73505	0.70765	0.62243	0.62552
VI - Margin	0.74167	0.70862	0.61884	0.62509
LSA	0.79834	0.75302	0.59848	**0.68230**
DSA	0.81588	**0.80198**	**0.65949**	**0.68230**
DeepGini	**0.81635**	0.76688	0.60278	0.59559
MCP	0.77444	0.71591	0.60024	0.56802

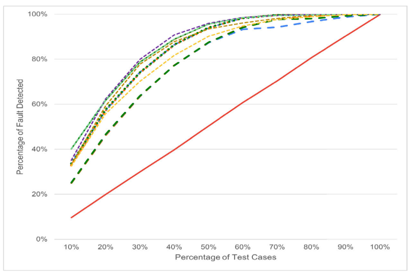
TABLE 13. Accuracy values for generated test dataset including OOD data.

	LeNet-5	LeNet-1	ResNet-32	VGG-16
Accuracy with OOD Data(%)	78.4	70.2	49.3	49.2
Accuracy without OOD Data(%)	84.1	76.3	65.5	63.9

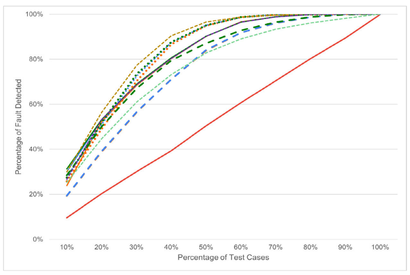
model is more exposed to datasets from distributions different than the one it was trained on.

Result for RQ3: The performance of uncertainty metrics decreases in prioritizing test datasets containing OOD data compared to datasets in which OOD data have been removed.

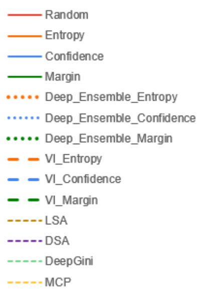
RQ4: Is it possible to use post-explainability methods in testing to understand the cause of test failures? We investigated the misclassifications in test datasets to understand the rationale behind the model's incorrect decision-making. We analyzed the results from the CIFAR-10 dataset with the ResNet-32 model and determined the predicted versus actual classes of misclassified test data for each test dataset generated. Confusion matrices for predicted and correct class labels are displayed in Fig. 5. The highest number of misclassified test data in the Generative Model test dataset are those whose actual class is Dog but was predicted as Cat or vice versa. Again, the highest number of misclassified test data instances in the Image Transformation test dataset are test data instances that are predicted as Cat but actually belong to the Dog class. Also, test images of the Deer class are wrongly predicted as Bird or Frog, whereas test data instances

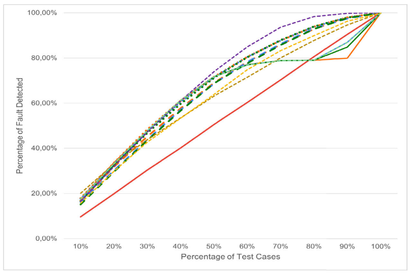


(a) LeNet-5 - Generated Test Dataset with OOD Data

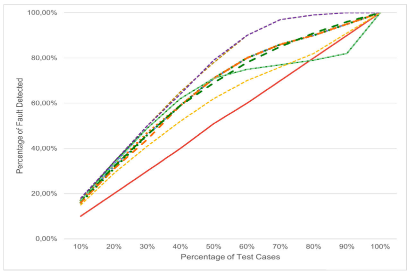


(b) LeNet-1 - Generated Test Dataset with OOD Data



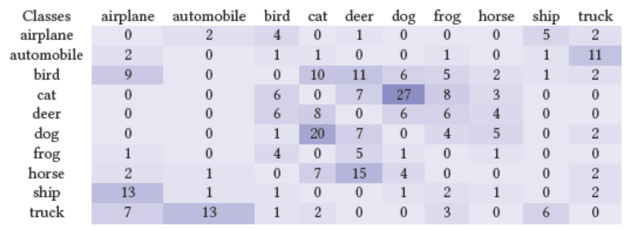


(c) ResNet-32 - Generated Test Dataset with OOD Data



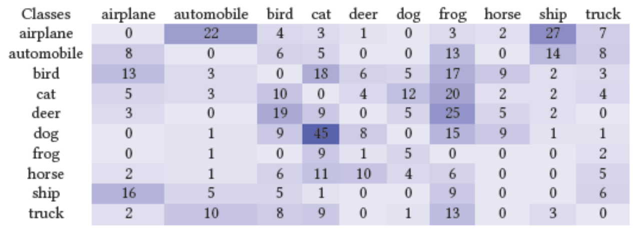
(d) VGG16 - Generated Test Dataset with OOD Data

FIGURE 4. **Fault detection ratios according to the size of generated test dataset including OOD data.**



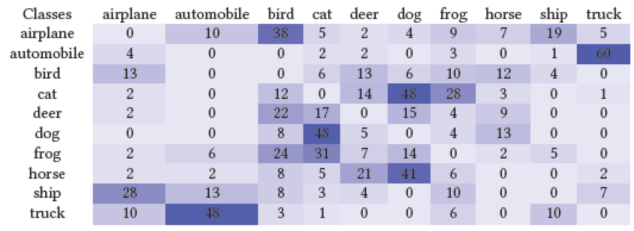
Classes	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0	2	4	0	1	0	0	0	5	2
automobile	2	0	1	1	0	0	1	0	1	11
bird	9	0	0	10	11	6	5	2	1	2
cat	0	0	6	0	7	27	8	3	0	0
deer	0	0	6	8	0	6	6	4	0	0
dog	0	0	1	20	7	0	4	5	0	2
frog	1	0	4	0	5	1	0	1	0	0
horse	2	1	0	7	15	4	0	0	0	2
ship	13	1	1	0	0	1	2	1	0	2
truck	7	13	1	2	0	0	3	0	6	0

(a) Generative Models Test Data



Classes	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0	22	4	3	1	0	3	2	27	7
automobile	8	0	6	5	0	0	13	0	14	8
bird	13	3	0	18	6	5	17	9	2	3
cat	5	3	10	0	4	12	20	2	2	4
deer	3	0	19	9	0	5	25	5	2	0
dog	0	1	9	45	8	0	15	9	1	1
frog	0	1	0	9	1	5	0	0	0	2
horse	2	1	6	11	10	4	6	0	0	5
ship	16	5	5	1	0	0	9	0	0	6
truck	2	10	8	9	0	1	13	0	3	0

(b) Image Transformation Test Data



Classes	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0	10	38	5	2	4	9	7	19	5
automobile	4	0	0	2	2	0	3	0	1	60
bird	13	0	0	6	13	6	10	12	4	0
cat	2	0	12	0	14	48	28	3	0	1
deer	2	0	22	17	0	15	4	9	0	0
dog	0	0	8	48	5	0	4	13	0	0
frog	2	6	24	31	7	14	0	2	5	0
horse	2	2	8	5	21	41	6	0	0	2
ship	28	13	8	3	4	0	10	0	0	7
truck	10	48	3	1	0	0	6	0	10	0

(c) Adversarial Attacks Test Data

FIGURE 5. **Predicted vs correct classes of misclassified test data.**

of the Airplane class are incorrectly classified as Automobile or Ship in the Image Transformation test dataset.

As adversarial attacks are artificial perturbations designed to deceive the model, there is a high number of misclassified test images in all classes, ranging from 64 to 108, as shown in Fig. 5c.

We examined the classification error that occurs most frequently between the Dog and Cat classes in detail. First, we apply the visualization methods to images of Dog and Cat classes from the original test dataset, which are correctly classified. This aims to identify the regions of the images that have the greatest impact on the model's prediction. According to the analysis results, we think that for the Dog class, the face of the dogs in images, specifically the nose region, affects the model's decision. Again, for the Cat class, the face of the cats mostly influences the decision of the model, but not the nose region in this case. In some of the images, the body of the cats was also factored into the decision-making of the model. In Fig. 6 and 7, the outcomes of these analyses conducted with the Grad-CAM, Grad-CAM++, and ScoreCAM techniques are depicted.

Then, we applied visualization techniques to the sample images that have been incorrectly classified as Cat when they actually belong to the Dog class, which is demonstrated in Fig. 8. The analyzed images belong to the Image Transformation test dataset and were generated with Rotate method. In the first row of images, the body of the dog is identified as the image region that affects the decision of the model. For the image on the third row, the highlighted region by visualization methods does not include the nose of the dog, and for the image on the fourth row, the ears of the dog are highlighted, as well as the face of the dog.

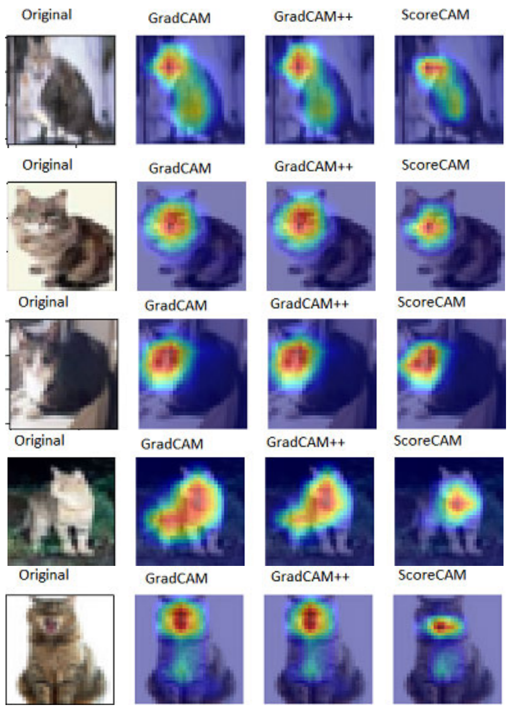


FIGURE 6. Correctly classified Cat images from the original CIFAR-10 test dataset.

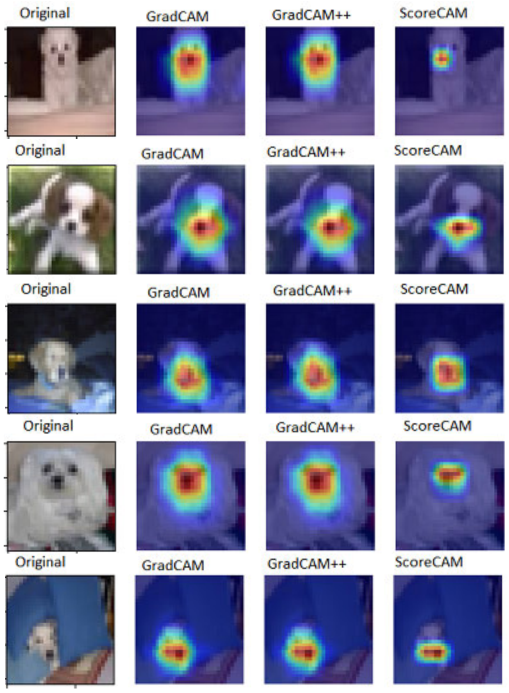


FIGURE 7. Correctly classified Dog images from the original CIFAR-10 test dataset.

Contrary to the images in Fig. 8, although the body of the dog is not seen in the last image row of Fig. 7 since the model recognizes the dog's nose in the image, it is classified as Dog. We hypothesize that when the model fails to identify the nose region in the dog images or when additional patterns that belong to other classes are also identified, the DNN model struggles to correctly classify the image in the Dog class. Images of dogs where the dog's nose does not appear properly or the color of the nose is not black can be considered to be added to the training data to overcome this misclassification case.

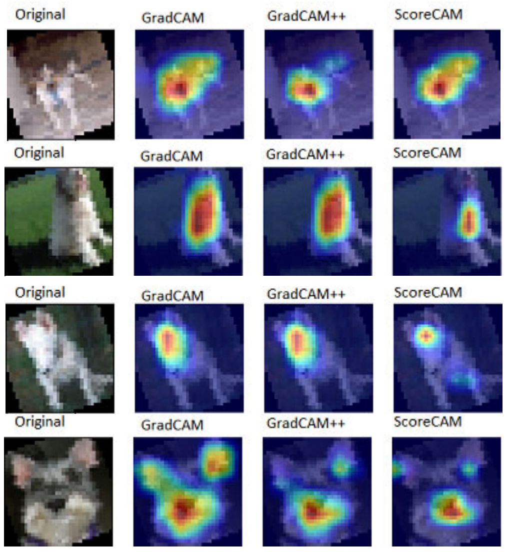


FIGURE 8. Dog images misclassified as Cat.

Result for RQ4: Analysis performed with visualization techniques used in DL explainability may help DL developers/testers to understand the rationale behind the model's decision for the test failures. However, this analysis requires manual and detailed assessment. According to reasonings derived from this analysis, actions to be taken in the DNN training to overcome the weaknesses of the model can be planned.

VI. LIMITATIONS

There is no labeled data in OOD detection to indicate whether data are OOD or not. The ability of an OOD detection method to distinguish between two different datasets with similar properties (e.g., picture resolution, color depth, etc.) is used to evaluate its performance. However, there is no ground truth about under what conditions the differences within the same test dataset, rather than the differences between two independent datasets, should be considered as OOD data. Different OOD detection methods may produce different outcomes when applied to the same datasets, and there is no universally agreed metric for comparing OOD detection approaches. Establishing standardized evaluation protocols and metrics for OOD detection and then benchmarking these methods on various types of datasets may aid in selecting and integrating the best-fitting OOD detection method into the DNN testing process for different problem domains. There is a recent study that provides an initial benchmark for OOD detection methods; however, distinct data sets were used as in-distribution and out-of-distribution sets in this study [88].

The dataset's characteristics are critical for the effectiveness of the OOD detection method. For example, a suitable approach must be chosen to capture data created by adversarial attack methods. While the data generated by adversarial attack methods FGSM and PGD were mostly detected as OOD with the VAE method utilized in the present study, the data generated by the C&W adversarial attack method were not detected as OOD. However, the accuracy values of the models were rather low on the data indicated to be not OOD.

Furthermore, the performance of test prioritizing techniques is affected by the characteristics of test datasets. The approaches that are independent of the tested model, such as the Deep Ensemble method, are insufficient for prioritizing fault-revealing data in test datasets generated with white box adversarial attack methods on the tested model. To prioritize this type of data, distance-based techniques such as LSA/DSA or methods based on the variance of the tested model's outputs can be chosen.

Examining the outputs of post-hoc explainability methods with the aim of determining the cause for the incorrect prediction of the model could assist the developer in uncovering weaknesses in the model or training dataset. However, a manual review is required in this scenario. More in-depth research in this area would be beneficial so that the data acquired with post-hoc procedures could be processed more quickly, reducing the amount of manual effort required.

VII. THREATS TO VALIDITY

The threat to external validity could originate from the selection of datasets and models. We selected two widely used and publicly available datasets (i.e., MNIST and CIFAR-10) and we employed well-known DNN architectures for each dataset. We employed two distinct models for each dataset in order to eliminate model dependence in the findings. Despite this, it is possible that some of our results do not generalize to other datasets and DNN models.

Another threat to validity could originate from the test data generation methods we used. To address this threat, we selected the three popular adversarial attacks in the literature and used their implementation in a well-known library. Additionally, for image transformations APIs supplied in the TensorFlow library were used.

The internal threat to validity lies in the implementation of the test prioritization approaches selected for comparison. To reduce this threat, we adopted the source code of the compared approaches published by their authors.

The threat to construct validity could originate from the measurements we used for assessing the effectiveness of the test prioritization strategies. We selected the average percentage of faults detected, RAUC, and fault detection ratio measurements, all of which are accepted metrics in software testing to evaluate the performance of test suites. Also, we used two different correlation methods for determining the relation between the misclassifications and test prioritization strategies.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a DNN test framework with the consideration of data distribution in DNN models. We demonstrated that popularly used test data generation methods produce large amounts of OOD data. With the addition of an OOD data detector to the testing process, this OOD data can be eliminated from the test dataset, and wasting effort in both labeling and investigating predictions for invalid or unintended test data can be prevented. We conducted uncertainty-based test data prioritization with nine different strategies on three generated test datasets and compared their effectiveness in identifying fault-revealing test data. Experiments demonstrate that while prioritization strategies that employ uncertainty metrics with the Deep Ensemble perform better for Image Transformation and Generative Model datasets, strategies that use metrics gathered from the Model Under Test produces the best results for test datasets generated with Adversarial Attacks. In all datasets, the Variational Inference with the Flipout method has the lowest performance in prioritizing fault-revealing test data.

Additionally, we evaluated the use of post-explainability methods for determining the cause of misclassifications in DNN models, analogues to debugging. CAM images can provide a clue to understanding the decision of the model, but these results are only relevant when assessed by an expert person in the field and the intended functionality of the model. Evaluating the CAM images can aid in detecting the underrepresented or overrepresented datasets that influence the model's decision-making. We concluded that a detailed analysis of the misclassified test data using visualization methods, can help to discover the cause of failures, and rather than adding all misclassified test data to the training dataset, more helpful decisions can be taken for the retraining process.

With the framework proposed in this study, a step has been taken to efficient use of time allocated for testing with the systematic test selection, which enables the elimination of OOD test data and the prioritization of fault-revealing test data. In addition, although promising, more research is needed on the use of post-explainability approaches with test data that cause failure in DNN models and on the retraining of models utilizing the inferences drawn from these analyses. In the future, we will improve our work by using alternative methods in the steps of the proposed testing framework.

APPENDIX A
TEST INPUT GENERATION PARAMETERS

The parameter values used for the adversarial attack functions in the Cleverhans library are given in Table 14.

The transformation values given as ranges in Table 15 are randomly applied to MNIST and CIFAR-10 datasets.

TABLE 14. Adversarial attacks parameters.

Method	Parameter Name	CIFAR-10	MNIST
CarliniWagnerL2	binary_search_steps	1	1
	max_iterations	100	1000
	learning_rate	5e-03	5e-03
	initial_const	10	1
FastGradientMethod2	eps	0.1	0.1
	norm	np.inf	np.inf
projected_gradient_descent	eps	0.3	0.1
	eps_iter	0.02	0.02
	nb_iter	40	40
	norm	np.inf	np.inf

TABLE 15. Image transformation parameters.

Method	Transformation Value
Blur	%0 - %50
Brightness	%0 - %10
Translate	%0 - %30
Rotate	%0 - %10

ACKNOWLEDGMENT

The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24411–24432, 2018.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[4] J. Ker, L. Wang, J. Rao, and T. Lim, "Deep learning applications in medical image analysis," *IEEE Access*, vol. 6, pp. 9375–9389, 2018.

[5] A. Loquercio, A. I. Maqueda, C. R. del-Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[6] Z. Wang, Q. Liu, and Y. Chi, "Review of Android malware detection based on deep learning," *IEEE Access*, vol. 8, pp. 181102–181126, 2020.

[7] *Road Vehicles-Functional Safety*, Standard ISO 26262, International Organization for Standardization, 2011.

[8] L. A. Johnson, "DO-178b: Software considerations in airborne systems and equipment certification," *Crosstalk*, vol. 199, pp. 11–20, Oct. 1998.

[9] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems," in *Proc. 33rd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2018, pp. 132–142.

[10] H. Zhou, W. Li, Z. Kong, J. Guo, Y. Zhang, B. Yu, L. Zhang, and C. Liu, "DeepBillboard: Systematic physical-world testing of autonomous driving systems," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. (ICSE)*, Oct. 2020, pp. 347–358.

[11] S. Kang, R. Feldt, and S. Yoo, "SINVAD: Search-based image space navigation for DNN image classifier test input generation," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Workshops*, Jun. 2020, pp. 521–528.

[12] T. Byun, A. Vijayakumar, S. Rayadurgam, and D. Cofer, "Manifold-based test generation for image classifiers," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Aug. 2020, pp. 15–22.

[13] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng. (ICSE)*, May 2018, pp. 303–314.

[14] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Operating Syst. Princ.*, Oct. 2017, pp. 1–18.

[15] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, "DLFuzz: Differential fuzzing testing of deep learning systems," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Oct. 2018, pp. 739–743.

[16] A. Odena, C. Olsson, D. Andersen, and I. Goodfellow, "TensorFuzz: Debugging neural networks with coverage-guided fuzzing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4901–4911.

[17] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, "DeepHunter: A coverage-guided fuzz testing framework for deep neural networks," in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2019, pp. 146–157.

[18] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "DeepConcolic: Testing and debugging deep neural networks," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Companion Proc. (ICSE-Companion)*, May 2019, pp. 111–114.

[19] S. Dola, M. B. Dwyer, and M. L. Soffa, "Distribution-aware testing of neural networks using generative models," in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE)*, May 2021, pp. 226–237.

[20] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," 2021, *arXiv:2110.11334*.

[21] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[22] D. Berend, X. Xie, L. Ma, L. Zhou, Y. Liu, C. Xu, and J. Zhao, "Cats are not fish: Deep learning testing calls for out-of-distribution awareness," in *Proc. 35th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2020, pp. 1041–1052.

[23] Q. Hu, Y. Guo, M. Cordy, X. Xie, L. Ma, M. Papadakis, and Y. Le Traon, "An empirical study on data distribution-aware test selection for deep learning enhancement," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 4, pp. 1–30, Oct. 2022.

[24] W. Ma, M. Papadakis, A. Tsakmalis, M. Cordy, and Y. L. Traon, "Test selection for deep learning systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 2, pp. 1–22, Apr. 2021.

[25] Y. Li, M. Li, Q. Lai, Y. Liu, and Q. Xu, "TestRank: Bringing order into unlabeled test instances for deep learning tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 20874–20886.

[26] R. Yan, Y. Chen, H. Gao, and J. Yan, "Test case prioritization with neuron valuation based pattern," *Sci. Comput. Program.*, vol. 215, Mar. 2022, Art. no. 102761.

[27] T. Byun, V. Sharma, A. Vijayakumar, S. Rayadurgam, and D. Cofer, "Input prioritization for testing neural networks," in *Proc. IEEE Int. Conf. Artif. Intell. Test. (AITest)*, Apr. 2019, pp. 63–70.

[28] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang, and Z. Chen, "DeepGini: Prioritizing massive tests to enhance the robustness of deep neural networks," in *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2020, pp. 177–188.

[29] W. Shen, Y. Li, L. Chen, Y. Han, Y. Zhou, and B. Xu, "Multiple-boundary clustering and prioritization to promote neural network retraining," in *Proc. 35th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2020, pp. 410–422.

[30] Y. Shi, B. Yin, Z. Zheng, and T. Li, "An empirical study on test case prioritization metrics for deep neural networks," in *Proc. IEEE 21st Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Dec. 2021, pp. 157–166.

[31] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, Feb. 2019.

[32] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020.

[33] F.-L. Fan, J. Xiong, M. Li, and G. Wang, "On interpretability of artificial neural networks: A survey," *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 5, no. 6, pp. 741–760, Nov. 2021.

[34] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6402–6413.

[35] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," 2018, *arXiv:1803.04386*.

[36] R. R Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-CAM: Why did you say that?" 2016, *arXiv:1611.07450*.

[37] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 839–847.

[38] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, "Score-CAM: Score-weighted visual explanations for convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 111–119.

[39] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Trans. Softw. Eng.*, vol. 48, no. 1, pp. 1–36, Jan. 2022.

[40] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5574–5584.

[41] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 13991–14002.

[42] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 427–436.

[43] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Inf. Fusion*, vol. 76, pp. 243–297, Dec. 2021.

[44] M. Henne, A. Schwaiger, K. Roscher, and G. Weiss, "Benchmarking uncertainty estimation methods for deep learning with safety-related metrics," in *Proc. SafeAI@ AAAI*, 2020, pp. 83–90.

[45] J. Caldeira and B. Nord, "Deeply uncertain: Comparing methods of uncertainty quantification in deep learning algorithms," *Mach. Learn., Sci. Technol.*, vol. 2, no. 1, Dec. 2020, Art. no. 015002.

[46] N. Ståhl, G. Falkman, A. Karlsson, and G. Mathiason, "Evaluation of uncertainty quantification in deep learning," in *Proc. Int. Conf. Inf. Process. Manage. Uncertainty Knowl.-Based Syst.* Cham, Switzerland: Springer, 2020, pp. 556–568.

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

[48] J. Bernhard, T. Schulik, M. Schutera, and E. Sax, "Adaptive test case selection for DNN-based perception functions," in *Proc. IEEE Int. Symp. Syst. Eng. (ISSE)*, Sep. 2021, pp. 1–7.

[49] S. Xu, Z. Wang, L. Fan, X. Cai, H. Ji, S.-C. Khoo, and B. B. Gupta, "DeepSuite: A test suite optimizer for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9506–9517, Jul. 2022.

[50] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng. (ICSE)*, May 2019, pp. 1039–1049.

[51] K. Zhang, Y. Zhang, L. Zhang, H. Gao, R. Yan, and J. Yan, "Neuron activation frequency based test case prioritization," in *Proc. Int. Symp. Theor. Aspects Softw. Eng. (TASE)*, Dec. 2020, pp. 81–88.

[52] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[53] G. Ras, N. Xie, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *J. Artif. Intell. Res.*, vol. 73, pp. 329–396, Jan. 2022.

[54] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proc. NAACL-HLT-Demonstrations Session*, 2016, pp. 97–101.

[55] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4765–4774.

[56] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 7, 2017, pp. 3319–3328.

[57] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3145–3153.

[58] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harvard J. Law Technol.*, vol. 31, no. 2, pp. 841–888, 2017.

[59] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee, "Counterfactual visual explanations," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2376–2384.

[60] W. J. Murdoch and A. Szlam, "Automatic rule extraction from long short term memory networks," 2017, *arXiv:1702.02540*.

[61] J. R. Zilke, E. L. Mencía, and F. Janssen, "DeepRED—Rule extraction from deep neural networks," in *Proc. 19th Int. Conf. Discovery Sci.* Cham, Switzerland: Springer, 2016, pp. 457–473.

[62] K. Leino, S. Sen, A. Datta, M. Fredrikson, and L. Li, "Influence-directed explanations for deep convolutional networks," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2018, pp. 1–8.

[63] V. Petsiuk, A. Das, and K. Saenko, "RISE: Randomized input sampling for explanation of black-box models," 2018, *arXiv:1806.07421*.

[64] P. Lertvittayakumjorn and F. Toni, "Explanation-based human debugging of NLP models: A survey," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 1508–1528, Dec. 2021.

[65] J. Adebayo, M. Muelly, I. Liccardi, and B. Kim, "Debugging tests for model explanations," 2020, *arXiv:2011.05429*.

[66] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[67] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, Apr. 2017.

[68] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[69] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf

[70] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[72] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[73] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[74] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[75] N. Papernot et al., "Technical report on the CleverHans v2.1.0 adversarial examples library," 2016, *arXiv:1610.00768*.

[76] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020.

[77] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.

[78] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[79] Z. Wang, H. Zheng, P. He, W. Chen, and M. Zhou, "Diffusion-GAN: Training GANs with diffusion," 2022, *arXiv:2206.02262*.

[80] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[81] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lect. IE*, vol. 2, no. 1, pp. 1–18, 2015.

[82] Z. Nado et al., "Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning," 2021, *arXiv:2106.04015*.

[83] V. Mosin, M. Staron, D. Durisic, F. G. d. O. Neto, S. K. Pandey, and A. C. Koppisetty, "Comparing input prioritization techniques for testing deep learning algorithms," in *Proc. 48th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2022, pp. 76–83.

[84] Z. Wang, H. You, J. Chen, Y. Zhang, X. Dong, and W. Zhang, "Prioritizing test inputs for deep neural networks via mutation analysis," in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng. (ICSE)*, May 2021, pp. 397–409.

[85] Y. Tao, C. Tao, H. Guo, and B. Li, "TPFL: Test input prioritization for deep neural networks based on fault localization," in *Proc. Int. Conf. Adv. Data Mining Appl.* Cham, Switzerland: Springer, 2022, pp. 368–383.

[86] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[87] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, Sierra Nevada, Spain, 2011. [Online]. Available: http://deeplearningworkshopnips2011.files.wordpress.com/2011/12/12.pdf
[88] J. Yang, P. Wang, D. Zou, Z. Zhou, K. Ding, W. Peng, H. Wang, G. Chen, B. Li, Y. Sun, X. Du, K. Zhou, W. Zhang, D. Hendrycks, Y. Li, and Z. Liu, "OpenOOD: Benchmarking generalized out-of-distribution detection," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 32598–32611.

DEMET DEMIR received the B.Sc. and M.Sc. degrees in computer engineering from Middle East Technical University, Ankara, Turkey, in 2000 and 2003, respectively, where she is currently pursuing the Ph.D. degree with the Graduate School of Informatics.

Her research interests include software verification, machine learning, and computer vision.

AYSU BETIN CAN received the bachelor's degree in computer engineering from Middle East Technical University (METU), Ankara, Turkey, in 1999, and the Ph.D. degree in computer science from the University of California at Santa Barbara, Santa Barbara, CA, USA, in 2005.

She is currently an Associate Professor with the Graduate School of Informatics, METU. Her research interests include automated software verification, automated software testing, interface-based modular verification, and specification.

ELIF SURER (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering from Boğaziçi University, Turkey, in 2005 and 2007, respectively, and the Ph.D. degree in bioengineering from the University of Bologna, Italy, in 2011. She is currently an Associate Professor with the Graduate School of Informatics, Department of Modeling and Simulation, Middle East Technical University (METU), Ankara, Turkey. Her research interests include serious games, virtual/augmented reality, human and canine movement analysis, machine learning, and computer vision.