Black-Box Testing of Deep Neural Networks through Test Case Diversity

Zohreh Aghababaeyan, Manel Abdellatif, Lionel Briand, Ramesh S, and Mojtaba Bagherzadeh

Abstract—Deep Neural Networks (DNNs) have been extensively used in many areas including image processing, medical diagnostics and autonomous driving. However, DNNs can exhibit erroneous behaviours that may lead to critical errors, especially when used in safety-critical systems. Inspired by testing techniques for traditional software systems, researchers have proposed neuron coverage criteria, as an analogy to source code coverage, to guide the testing of DNNs. Despite very active research on DNN coverage, several recent studies have questioned the usefulness of such criteria in guiding DNN testing. Further, from a practical standpoint, these criteria are white-box as they require access to the internals or training data of DNNs, which is often not feasible or convenient. Measuring such coverage requires executing DNNs with candidate inputs to guide testing, which is not an option in many practical contexts. In this paper, we investigate diversity metrics as an alternative to white-box coverage criteria. For the previously mentioned reasons, we require such metrics to be black-box and not rely on the execution and outputs of DNNs under test. To this end, we first select and adapt three diversity metrics and study, in a controlled manner, their capacity to measure actual diversity in input sets. We then analyze their statistical association with fault detection using four datasets and five DNNs. We further compare diversity with state-of-the-art white-box coverage criteria. As a mechanism to enable such analysis, we also propose a novel way to estimate fault detection in DNNs. Our experiments show that relying on the diversity of image features embedded in test input sets is a more reliable indicator than coverage criteria to effectively guide DNN testing. Indeed, we found that one of our selected black-box diversity metrics far outperforms existing coverage criteria in terms of fault-revealing capability and computational time. Results also confirm the suspicions that state-of-the-art coverage criteria are not adequate to guide the construction of test input sets to detect as many faults as possible using natural inputs.

Index Terms—Deep Neural Network, Test, Diversity, Coverage, Faults.

✦

1 INTRODUCTION

Over the last decade, Deep Neural Networks (DNNs) have achieved successful performance in many domains, such as image processing [1], [2], medical diagnostics [3], [4], [5], speech recognition [6] and autonomous driving [7], [8]. Similar to traditional software components, DNN models often exhibit erroneous behaviours that may lead to potentially critical errors. Therefore, like traditional software, DNNs need to be tested effectively to ensure their reliability and safety.

In the software testing context, code coverage criteria (e.g. branch coverage, statement coverage) are used to guide the generation of test cases and assess the completeness of test suites [9]. While full coverage does not ensure functional

- *Zohreh Aghababaeyan is with the School of EECS, University of Ottawa, Ottawa, Canada.*
  E-mail: zagha052@uottawa.ca
- *Manel Abdellatif is with the Software and Information Technology Engineering Department, École de Technologie Supérieure, Montreal, Canada. She contributed to this work mainly during her postdoctoral fellowship at the School of EECS, University of Ottawa, Ottawa, Canada.*
  E-mail: Manel.abdellatif@etsmtl.ca
- *Lionel Briand is with the School of EECS, University of Ottawa, Ottawa, Canada, and also with the SnT Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg.*
  E-mail: Lbriand@uottawa.ca
- *Mojtaba Bagherzadeh is with the School of EECS, University of Ottawa, Ottawa, Canada.*
  E-mail: Mbagherz@cisco.com
- *Ramesh S is with the Department of Research and Development, General Motors, Warren, MI, USA.*
  E-mail: Ramesh.s@gm.com

correctness, high coverage increases stakeholders' confidence in the testing results because it triggers more code execution paths. Inspired by code coverage, several coverage criteria have been introduced to measure the adequacy of test data in the context of DNNs [10], [11], [12], [13], [14]. Neuron coverage measures the extent to which neurons in a DNN are activated based on certain input data. Intuitively, test inputs with higher neuron coverage are desirable. However, reaching high neuron coverage with a few test inputs is usually easy to achieve [14], [15] and the usefulness of such coverage is therefore questionable. Furthermore, defining coverage in DNNs is not as straightforward as testing traditional software because in the latter the code logic is explicit but in DNNs that logic is not represented explicitly. Although more sophisticated coverage criteria have been proposed, several articles have criticized the use of such coverage to guide the testing of DNN models [16], [17], [18].

In traditional software systems, testers rely on coverage metrics because they assume that (1) inputs covering the same part of the source code are homogeneous (i.e. either all or none of these inputs trigger a failure), and (2) the inputs used in testing should be diverse to ensure high coverage [16]. However, these assumptions break down in DNN testing because (1) unlike code coverage, neuron coverage does not fully exercise the implicit logic embedded in DNNs; (2) the homogeneity assumption is broken with adversarial inputs; and (3) increasing the diversity of inputs does not necessarily increase DNN coverage [16]. Further, most coverage studies rely on adversarial inputs to validate their proposed criteria [12], [13], [14], [10], [11]. However,

these inputs are mostly unrealistic and used to study the robustness of the DNN model instead of its accuracy. While state-of-the-art coverage criteria have been largely validated with artificial inputs generated based on adversarial methods, their claimed sensitivity to adversarial inputs does not necessarily mean that they relate to the fault detection capability of natural test input sets. This is confirmed by various studies [16], [18] that have failed to find a significant correlation between coverage and the number of misclassified inputs in a natural test input set, despite a positive correlation in the presence of adversarial test inputs. Consequently, coverage criteria may be ineffective in guiding DNN testing to increase the fault-detection capability of natural test input sets. Further, another study [17] found that retraining DNN models with new input sets that improve coverage does not increase the robustness of the model to adversarial attacks.

Furthermore, coverage criteria require full access to the internals of the DNN state or training data, both of which are often not available to testers, especially when the DNN model is proprietary and provided by a third party. Thus, in our project, we focus on black-box input diversity metrics to provide guidance on how to assess test suites or select test cases for DNNs. We target diversity because it has been successfully used in testing software systems [19], [20], [21]. Intuitively, relying on diverse test inputs should increase the exploration of the fault space and thus increase the fault detection capability of a given test input set. Further, we target black-box metrics that do not require executing test inputs on the DNN under test since this is a strong practical impediment in many application contexts, such as when dealing with large models and large databases of unlabeled inputs. We also target black-box metrics that are model-independent and do not rely on the outputs of DNNs under test because they cannot be trusted when the models are not accurate [22]. Based on these requirements, we propose and investigate black-box diversity metrics for DNNs that rely on inputs' features, investigate their relationships with coverage metrics and analyze their association with fault detection. In other words, this paper focuses on the fundamental assumptions related to the relationship between testing criteria (i.e. coverage and diversity metrics) and faults in DNNs. However, this paper does not investigate how these testing criteria might be used for specific testing scenarios such as the selection, minimization or generation of test sets. Nonetheless, investigating the relationship between DNN faults and testing criteria is an essential step for selecting proper criteria, independent of any specific purpose.

In traditional software systems, some of the inputs causing failures are usually very close to each other [23], [24]. Similarly, it has been observed that many mispredicted inputs in DNNs fail due to the same causes [25]. Counting such inputs to assess the fault detection capability of a test suite is therefore misleading. However, the notion of fault, though rather straightforward in regular software, is elusive in DNNs. For this reason, we rely on a clustering-based fault estimation approach to group similar mispredicted inputs based on their features and misprediction behaviour [25]. We assume that each cluster corresponds to a fault because similar mispredicted inputs belonging to the same cluster are assumed to be mispredicted for similar

reasons. To assess test suites for DNNs, we use and adapt three diversity metrics. As we evaluate datasets composed of images, commonly used as inputs in many DNNs (e.g. the perception layer of cyber-physical systems), we rely on a feature extraction model to extract features from images that will be used to compute the diversity of test input sets. We evaluate the selected metrics in terms of their capability to measure actual diversity based on extracted features. We then analyze their associations with fault detection in DNNs using four widely used datasets and five different DNN models. We further study state-of-the-art white-box coverage metrics and their associations with diversity and fault detection.

Based on our experiments, we show that diversity metrics, and geometric diversity (GD) [26] in particular, though black-box and without the use of any DNN internal information, far outperform existing coverage criteria in terms of fault-revealing capability and computational time. We also show that state-of-the-art coverage metrics are not correlated to faults or diversity in natural test input sets.

Overall, the main contributions of our paper are as follows:

- We propose and study the use of black-box diversity metrics to guide the testing of DNN models. We show that geometric diversity is the best option to guide the testing of DNN models because it is positively correlated to faults in subsets.
- We introduce and validate a clustering-based approach to estimate faults in DNNs as test input sets typically contain many similar mispredicted inputs caused by the same problems in the DNN model. We explain why this is a requirement to evaluate any test set evaluation criterion.
- We study state-of-the-art coverage criteria and show that there is no correlation between coverage and faults in DNN models. Further, coverage is not correlated with diversity in input sets. Our results question the reliability of coverage, as it is currently defined, to guide DNN testing if the objective is to detect as many faults as possible.

The remainder of the paper is structured as follows. Section 2 presents our approach and describes the selected diversity metrics. Section 3 presents our empirical evaluation and results. Section 4 discusses the implications of our results and our recommendations for guiding the testing of DNN models. Section 5 describes the threats to the validity of our study. Sections 6 and 7 contrast our work with related work and conclude the paper, respectively.

2 APPROACH

A central problem in software testing, especially when test oracles (verdicts) are not automated, is the selection of a small set of test cases that sufficiently exercise a software system. Intuitively, testers should select a set of diverse test cases because selecting similar test cases does not bring extra benefits to fault detection. In this paper, we study diversity metrics with the ultimate aim of using them to guide DNN testing, relying on the best diversity metric in both the capacity to uncover erroneous behavior and computational complexity. We target black-box diversity metrics that are

model-independent because we cannot rely on DNNs output when the models are not accurate [16]. We also target black-box metrics that do not require executing the model with all inputs because it would impede their application when working with large models and datasets. Therefore, we use and adapt three diversity metrics that have been applied widely in other contexts and are based on inputs. We rely on a feature extraction model to extract features from images that we use to compute diversity. In section 3, we will first evaluate the selected metrics in terms of their capability to measure the actual diversity of a test input set. Then we will study their relationships with state-of-the-art white-box coverage metrics and analyze their associations with fault detection in DNNs.

In this section, we describe the feature extraction method and the diversity metrics that we used, and detail the evaluation process in the following section.

2.1 Feature Extraction

For diversity to account for the content of images, we need to extract features from each input image in the test input set. Consequently, we rely on VGG-16 [27], which is one of the most-used and accurate state-of-the-art feature extraction models [28], [29]. It is a pre-trained convolutional neural network model and consists of 16 weight layers, including 13 convolutional layers with a filter size of 3×3, and three fully connected layers. The model is trained on ImageNet [1], which is a dataset of over 14 million labeled images belonging to 22,000 categories.

We use VGG-16 to extract the features of images. A feature is an activation value on the layer after the last convolutional layer of the VGG-16 model. A set of features can characterize semantic elements such as shapes and colors. We extract the features in the test input set S and build the related feature matrix Vs where (1) each row of the matrix corresponds to the feature vector of an input in the test set, and (2) each column corresponds to a feature.

After generating the feature matrix, we normalize it by applying *Min-Max normalization* per feature, which is one of the most common and simple ways to normalize data. For each feature in Vs, the maximum and minimum values of that feature are transformed to one and zero, respectively, and every other value is transformed to a real value between zero and one. The *Min-Max normalization* is defined as follows. For every feature in the feature matrix Vs where $j \in [1, 2, ..., m]$ and m is the number of features, the normalized feature Vs'_j is calculated as follows:

$$Vs'_j(i) = \frac{Vs_j(i) - min(Vs_j)}{max(Vs_j) - min(Vs_j)} \quad (1)$$

We normalize the feature matrix (1) to make the computation of the selected diversity metrics more scalable, and (2) to eliminate the dominance effect of features with large value ranges.

2.2 Diversity metrics

In this section, we describe the selected diversity metrics: Geometric Diversity [26], [30], Normalized Compression Distance [31], [21], and Standard Deviation.

1. https://image-net.org/index.php

We chose these metrics based on the following criteria. First, we targeted diversity metrics that measure diversity within a subset. We did not consider metrics that measure diversity in relation to another subset (e.g. Kullback-Leibler [32], Jensen-Shannon divergence [33]). Second, we selected diversity metrics that can be applied to our datasets, specifically targeting metrics that can be applied to images. Third, we selected diversity metrics that do not depend on the DNN model under test and do not require the execution of this model with all inputs. Finally, we targeted diversity metrics that are widely used in a variety of other application contexts. For instance, the geometric diversity metric has been used in a variety of machine learning applications such as the selection of training sets with the Determinantal Point Process method [26], data summarization [34] and data clustering [35], [36]. Furthermore, the standard deviation metric is considered to be a common diversity metric that has been successfully applied in different contexts to measure text and image similarity [37]. The Normalized Compression Distance metric has been employed in many application domains such as image processing [31], security [38] and clustering [39], [31]. This metric supports any type of input and has been used recently to guide the selection of diverse input tests for regular software systems [21].

In this section, we will describe each of these metrics and discuss their strengths and limitations.

2.2.1 Geometric Diversity

The geometric diversity metric measures the diversity of the selected inputs [26]. As mentioned previously, this metric is widely used to select diverse input sets with the Determinantal Point Process (DPP) method [26], [30]. DPP is applied to guide the selection of diverse subsets from a fixed ground set [40] and has been used in a variety of machine learning applications for images [26], videos [41], documents [34], recommendation systems [42] and sensor placement [43]. The key characteristic of DPP is that including one item makes including other similar items less likely (i.e. a DPP assigns a greater probability to subsets of items that are diverse). Thus, a DPP value of a subset indicates its diversity, where the higher this value, the more diverse the subset. The key component in DPP is geometric diversity that measures the diversity of an input set in terms of the (hyper)volume spanned by the input feature vectors (feature matrix).

2.2.1.1 Definition

The geometric diversity $G(.)$ is defined as follows. Given a dataset X, a number of inputs n, a number of features m, and feature vectors $V \in R^{n*m}$, the geometric diversity of a subset $S \subseteq X$ is defined as:

$$G(S) = det(Vs * Vs^T) \quad (2)$$

which corresponds to the squared volume of the parallelepiped spanned by the rows of Vs, since they correspond to vectors in the feature space. The larger the volume, the more diverse S is in the feature space, as illustrated in Figure 1. It is expected that very different (similar) images result in very different (similar) feature vectors.
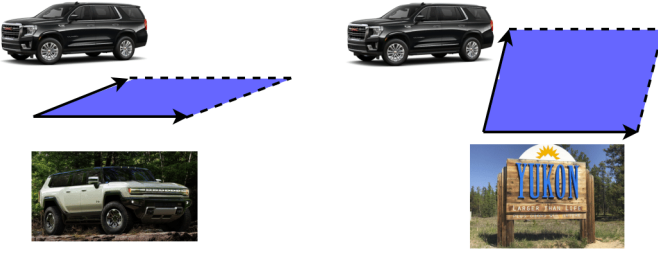
Figure 1: Illustration of the geometric diversity metric

2.2.1.2 Calculation

Geometric diversity takes as input the feature matrix of the test input set, as generated using the feature extraction model. Because geometric diversity relies on the calculation of the determinant of a matrix, we need to handle several challenges related to such processing.

Determinant Overflow. The determinant is likely to run into overflow when we work with large feature matrices. The main cause of the problem is that the determinant value is too large to be represented by a real number. To overcome this problem, we follow the recommendations of Celis *et al.* [44] and use the logarithm of this value rather than the determinant itself. We also overcome the determinant overflow problem by using the normalized feature matrix, Vs' thus making the geometric diversity computation more scalable.

Mathematical Limitations. If a matrix contains at least two linearly dependent vectors, its determinant will be equal to zero. Consequently, we cannot calculate the geometric diversity score of an input set that contains duplicate inputs. The feature extraction model predicts features for each test input. If the feature values are the same for two test inputs, we have duplicate inputs. This means the two test images are redundant in terms of this feature extraction model. We therefore have to delete redundant inputs before calculating the diversity score. This kind of pre-processing is acceptable in our context because (1) duplicate inputs that do not add any value to our testing model, and (2) we aim to test the DNN model with a diverse input set to detect faults.

Further, the maximum subset size for which we calculate GD must be less than the number of the features in Vs. This is also due to the mathematical limitations of the determinant and the rank of matrices.

Proof: In linear algebra, the rank of a matrix A of size $n*m$ refers to the number of linearly independent rows or columns in the matrix. Consequently, $Rank(A_{n*m}) <= min(n, m)$, where n is the number of lines in the matrix A, and m is the number of columns. Consider a square matrix B of size $n*n$. By definition, If $Rank(B) < n$ then $Det(B) = 0$. Let us assume that $B = A * A^T$. By definition $Rank(B) = Rank(A * A^T) = Rank(A)$. If $n > m$ then $Rank(B) \leq m < n$. As a result $Det(B) = Det(A * A^T) = 0$.

To mitigate this mathematical limitation, we can select one of the internal layers of the feature extraction model where the number of linearly independent features is equal to or greater than the size of the subset. We propose to use the deepest hidden layer, which provides enough features

because, as noted by Bengio *et al.* [45], [10], deeper layers represent higher-level features of the input. Specifically, we can select hidden layers that are possible candidates for feature extraction because their number of linearly independent features is greater than or equal to the size of the subset. From these candidates, we then select the deepest hidden layer because it is likely to contain the most semantically significant and helpful features for characterizing an input.

2.2.2 Normalized Compression Distance

The Normalized Compression Distance (NCD) is a similarity metric based on the Kolmogorov complexity [46] and information distance [47] where we measure the information required to transform one object into another to assess the similarity between these objects. Because of the complexity in calculating the Kolmogorov complexity, we approximate it by using real-world compressors [31], [48]. This leads to the normalized compression distance [39], which has been extended by Cohen *et al.* [31] to support the calculation of multisets' similarity.

2.2.2.1 Definition

The NCD metric for a multiset S is calculated via an intermediate measure NCD_1 [31], [49], [21]:

$$NCD_1(S) = \frac{C(S) - min_{s \in S}\{C(s)\}}{max_{s \in S}\{C(S \setminus \{s\})\}} \quad (3)$$

$$NCD(S) = max\Big\{NCD_1(S), max_{Y \subset S}\{NCD(Y)\}\Big\} \quad (4)$$

where $C(S)$ denotes the length of S after compression [31], [21].This metric is interpreted by Cohen *et al.* [31] as follows. For example, if a multiset S of strings (inputs) of about 1,000,000 bits each have pairwise information distances of 1,000 bits between each pair of inputs, then those strings can be considered relatively similar. If, on the other hand, a multiset S contains strings of about 1,200 bits each, and each pair of strings in S has a pairwise information distance of 1,000 bits, then we can conclude that the inputs in S are quite diverse [31]. NCD supports any type of input (e.g text, images, execution traces) and has many applications, such as in pattern recognition [50], [51], clustering [31], [39], security [38] and measuring the diversity of test sets [21], [52].

2.2.2.2 Calculation

We have re-implemented the NCD metric for multisets based on the original paper [21]. NCD takes the normalized feature matrix of an input set and measures its diversity score. It takes values in the range $[0, 1]$. The more diverse the input set, the larger the NCD score. However, one limitation of this metric is its high computational cost, such that its application on large input sets becomes prohibitive [31], [21]. NCD is highly sensitive to the used compression tool [21]. Different compression tools determine various performance aspects of NCD, such as computation time, used memory and compression distance. Following the recommendations of existing papers on NCD [21], [31], [38], we tried different compression tools like *Lzm*, *Bzip2* and *Zlip*. We tested their efficiency in computational cost and correctness in generating diversity scores. We evaluated the correctness of the diversity scores by controlling the actual diversity of input

sets in terms of features and compared the corresponding NCD scores. We compared the NCD score of input sets with similar images to other sets with different images. The NCD score was expected to increase when the input set was more diverse in terms of features. The best results were obtained with *Bzip2*, which we used in our experiments.

2.2.3 Standard Deviation

Standard deviation (STD) is a statistical measure of how far from the mean a group of data points is, determined by calculating the square root of the variance.

2.2.3.1 Definition

STD is a straightforward measure of the diversity of a test input set based on the statistical variation of the inputs' features. We define the STD metric as the norm of the standard deviation of each feature in the test input set. Formally, we define the STD of an input set S of size n as follows:

$$STD(S) = \left\| \left(\sqrt{\sum_{i=1}^{n} \frac{Vs_{i,j} - \mu_j}{n}}, 1 \le j \le m \right) \right\| \quad (5)$$

where Vs is the feature matrix of the input set S, m is the number of features, and μ_j is the mean value of feature j in Vs.

2.2.3.2 Calculation

To calculate STD for an input set S, we first extract the feature matrix for S and normalize it. Then we calculate the norm of the standard deviations of each feature in the matrix to measure the diversity of the input set. The higher the STD, the more diverse the input set. One of the limitations of the standard deviation is its dependence on the mean, which introduces unwanted bias in some cases. To explain this, we use two same-size subsets, A and B, where (1) in subset A we have two sets of similar inputs and these two sets are far from each other in the features space, and (2) in subset B all inputs are different from one another. The variance of the inputs in subset A with respect to the mean could be larger than the one in subset B. In such a case, $STD(A)$ would be larger than $STD(B)$ though subset B is more diverse than A, as the latter only contains two truly distinct groups of inputs.

3 EMPIRICAL EVALUATION

This section describes the empirical evaluation of our approach, including research questions, datasets, DNN models, experiments and results.

3.1 Research Questions

Our empirical evaluation is designed to answer the following research questions.

- **RQ1. To what extent do the selected diversity metrics measure actual diversity in input sets?** We want to assess, in a controlled way, the reliability of the selected diversity metrics for measuring the actual diversity of an input set in terms of the features the images contain. Only the metrics that reliably reflect changes in image diversity will be retained for the next research questions.

- **RQ2. How does diversity relate to fault detection?** Similar to other studies in different contexts [21], [53], [54], we aim to investigate the correlation between diversity and faults to assess whether diverse input sets lead to higher fault coverage. We do not investigate in this research question the correlation between diversity and the number of mispredicted inputs, as this is misleading. Many mispredictions result from the same problems in the DNN model and are therefore redundant. This is similar to failures in regular software. In classification problems, for example, guiding the selection of test inputs to maximize misprediction rates (the number of mispredicted inputs / total number of inputs) could thus be misleading. However, the notion of fault in DNN models is not as straightforward as it is in regular software, where we can identify statements responsible for failures. Therefore, to investigate this research question, we need to first define a mechanism to compare how effective test sets are in detecting faults in DNNs, so that we can then investigate the relationship between diversity and faults.

- **RQ3. How does coverage relate to fault detection?** Similar to diversity, we aim to assess the association between state-of-the-art coverage metrics and faults. This enables us to compare black-box diversity and white-box coverage in selecting test sets with high fault-revealing power. Note that recent studies questioned the use of coverage metrics to assess DNN test inputs [16], [55]. Most state-of-the-art coverage metrics strongly rely on artificial inputs generated based on adversarial methods [12], [13], [14], [10], [11]. However, their positive correlation with the presence of adversarial inputs does not necessarily mean that they are efficient enough to reveal the fault detection capability of natural test input sets. Several studies [16], [18] failed to find a strong correlation between coverage and misprediction rates when using only natural input sets. Furthermore, coverage metrics showed poor performance in guiding the retraining of DNN models to improve the robustness of the model to adversarial attacks [17], [56]. Therefore, there is still no consensus on which coverage metrics are suitable for different DNN testing-related tasks such as test selection, minimization and generation.

- **RQ4. How do diversity and coverage perform in terms of computation time?** We aim to compare the computation times of selected diversity and coverage metrics. Most importantly, we aim to study how these computation times scale as the sizes of the test sets increase. Excessive computation times may limit applicability, though what is acceptable depends on the context.

- **RQ5. How does diversity relate to coverage?** Though diversity is black-box and therefore has inherent practical advantages, it is interesting to study the correlation between diversity and coverage to determine if they essentially capture the same thing. Though this question can be answered indirectly by some of the previous questions (correlations are transitive), such correlation analysis can provide additional insights to explain and support previous results.

Dataset	Description	DNN Model	Accuracy
MNIST	Handwritten digit images composed of 60,000 images for training and 10,000 images for testing.	LeNet-5	87.85%
		LeNet-1	84.5%
Fashion-MNIST	Grayscale images in 10 different classes of clothes composed of 60,000 images for training and 10,000 images for testing.	LeNet-4	88%
Cifar-10	Object recognition dataset in ten different classes composed of 50,000 images for training and 10,000 images for testing.	A 12-layer ConvNet with max-pooling and dropout layers.	82.93%
		ResNet20	86%
SVHN	A real-world image dataset for recognizing house numbers obtained from Google Street View images. It is composed of 73,257 training images and 26,032 testing images.	LeNet-5	88%

Table 1: Datasets and models used for evaluation

3.2 Subject Datasets and DL Models

Table 1 shows the characteristics of the datasets and models in our experiments. We used four common image recognition datasets, Cifar-10 [57], MNIST [58], Fashion-MNIST [58] and SVHN [59]. We use these datasets with five state-of-the-art DNN models: 12 layers Convolutional Neural Network (12-layer ConvNet), LeNet-1, LeNet-4, LeNet-5 and ResNet20.

Cifar-10 contains 50,000 images for training and 10,000 for testing. These images belong to 10 different classes (e.g cats, dogs, trucks).

We also used MNIST, which contains 70,000 images (60,000 for training and 10,000 for testing). Each of these images represents a handwritten digit and belongs to one of the 10 classes. We included Fashion-MNIST, which contains grayscale images in 10 different classes of clothes. It is composed of 60,000 images for training and 10,000 images for testing. Finally, we use SVHN, a real-world image dataset for recognizing house numbers. It contains 99,289 images where 73,257 are for training and 26,032 are for testing.

For Cifar-10, we used a 12-layer ConvNet and ResNet20 that we trained for 50 and 100 epochs, respectively. For MNIST, we used the LeNet-1 and LeNet-5 models that we trained for 50 epochs. We trained the LeNet-4 model with Fashion-MNIST for 20 epochs. Finally, for SVHN, we used the LeNet-5 model, which we trained for 100 epochs. The different combinations of models and datasets, along with the models' accuracy, are detailed in Table 1.

We selected these datasets and models because they are widely used in the literature [12], [10], [11], [13]. Further, all the inputs in the selected datasets are correctly labelled. These datasets and models are considered good baselines to observe key trends, as they offer a wide range of diverse inputs (in classes and domain concepts) and different models (in terms of internal architecture).

3.3 Evaluation and Results

Before addressing our research questions, one essential issue was how to count faults in DNNs. A misprediction implies the existence of a fault in the DNN. However, identifying faults is not as straightforward as in regular software, where faulty statements that cause failures can be identified. Nevertheless, estimating fault detection effectively is essential to compare coverage and diversity metrics. Simply comparing misprediction rates is misleading as many test inputs are typically mispredicted for the same reasons [25]. Typically, with regular software, a tester does not select input tests to maximize the failure rate (equivalent to the misprediction rate in our context) but rather wants to maximize the number of distinct detected faults. This should be not different with DNNs, where we want to detect the distinct causes of mispredictions.

We illustrate this issue in Figure 2 where we represent an example of a test input set in a two-dimensional feature space. Black dots refer to the inputs correctly predicted by the DNN under test, and red dots represent the mispredicted ones. We select two subsets from the initial set and measure their corresponding misprediction rates. As shown in Figure 2, subset 1 is less diverse than subset 2 but has a higher misprediction rate. However, some of the mispredicted inputs are very similar and somewhat redundant.

As a result, it can be argued that subset 2 is more diverse than subset 1 and is more informative for testing the model because its mispredicted inputs potentially reveal more
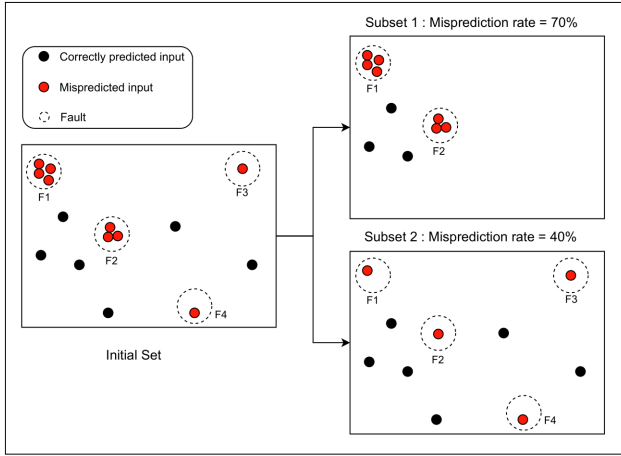
Figure 2: Relying on misprediction rates is misleading

faults in the DNN model. In preliminary experiments (not included in this paper), we evaluated the computation of misprediction rates in test input sets and studied their correlation with diversity and coverage and found no statistically significant correlation for both diversity and coverage metrics. We suspected that accounting for numerous redundant test inputs affected our correlation analysis. In practice, selecting or generating test inputs that trigger failures (i.e mispredictions) is far more useful when these failures are diverse [60]. A test set that repeatedly exposes the same problem in the DNN model is a waste of computational resources, especially when we have a limited testing budget and a high labeling cost for testing data [60]. This is why, similar to other studies comparing the effectiveness of test strategies with regular software, we want here to address the notion of faults detected in DNNs and study their association with diversity and coverage.

3.3.1 Estimating Faults in DNNs

Following a similar approach to the work of Fahmy *et al.* [25] and Attaoui *et al.* [61], we rely on a clustering approach to group similar mispredicted inputs presenting a common set of characteristics that are plausible causes for mispredictions. We approximate the number of detected faults in a DNN through such clustering. Although many mispredicted test inputs are redundant and result from the same causes, we assume that test inputs belonging to different clusters are mispredicted due to distinct problems [25] in the DNN model. This is an approximation but a practical and plausible way to estimate and compare the number of detected faults across coverage and diversity strategies. Although faults can only be addressed by retraining in DNNs, as opposed to debugging, clusters nevertheless capture common causes for mispredictions and are thus comparable to faults in regular software. Figure 3 depicts how faults are counted in DNNs and we describe below each step in detail.

3.3.1.1 Feature Extraction

We start by training our model using the training dataset. We then run our pre-trained model on the test and training datasets to identify all mispredicted inputs. We not only use mispredicted inputs from the test set but also mispredicted inputs from the training dataset to extract the best clusters

and estimate detected faults as accurately as possible. We rely on VGG16 to extract the mispredicted inputs' features and build the corresponding feature matrix as described in section 2.1. We add two extra features to the matrix from the DNN model to capture actual and mispredicted classes (labels) related to each misclassified input. This adds information to the feature matrix about the misprediction behaviour of the model under test for each mispredicted input, which we believe builds better clusters to reflect common misprediction causes.

3.3.1.2 Dimensionality Reduction

By definition, the number of input features for a dataset corresponds to its dimensionality. Low density in high-dimensional spaces makes it difficult, in general, for typical clustering algorithms to find a continuous boundary that separates the different clusters [62]. Therefore, employing dimensionality reduction techniques can help clustering algorithms make the inputs and their related clusters more distinguishable. Because we are working with high-dimensional inputs (512 features from the VGG model and two features from the DNN model), we rely on the Uniform Manifold Approximation and Projection (UMAP) [63] dimensionality reduction technique. We selected UMAP because several studies [64], [65] have shown its effectiveness as a pre-processing step to boost the performance of clustering algorithms when compared to other state-of-the-art dimensionality reduction techniques, such as PCA [66] and t-SNE [67]. PCA is a linear dimensionality reduction technique that performs poorly on features with nonlinear relationships. To work with high-dimensionality data to obtain low-dimensionality and nonlinear manifolds, some nonlinear dimensionality reduction algorithms, such as UMAP and t-SNE, should be used [65]. However, t-SNE is more computationally expensive than UMAP and PCA. It is used in practice for data visualization and data reduction to two or three dimensions. Furthermore, it involves hyperparameters that are not always easy to tune in order to get the best results. Therefore, we relied on UMAP for dimensionality reduction as an effective pre-processing step to boost the performance of density-based clustering. This will be used in the next step.

3.3.1.3 Clustering

After performing dimensionality reduction, we apply the HDBSCAN [68] clustering algorithm to group mispredicted inputs that are similar and believed to result from the same causes (faults) in the DNN model. HDBSCAN is a density-based clustering algorithm where each dense region is considered a cluster and low-density regions are considered noise. In other words, it views clusters as areas of high density separated by areas of low density. Clusters found by HDBSCAN can be of any shape, as opposed to other types of clustering algorithms, such as k-means or hierarchical clustering, which assume that clusters are convex-shaped. Each cluster is supposed to correspond to a fault (common problems) in the DNN model because their inputs are similar in terms of extracted features and actual and mispredicted classes.
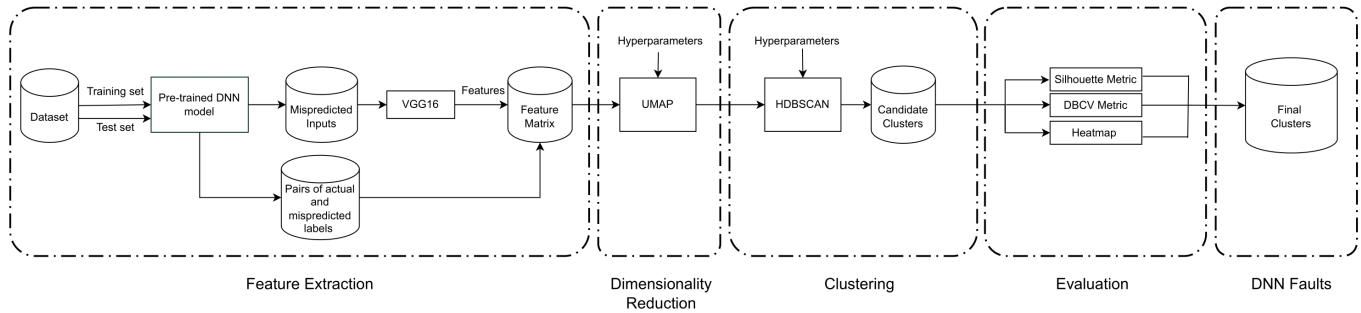
Figure 3: Estimating faults in DNNs

Dataset	Model	#Misp. in training set	#Misp. in test set	Silh.	DBCV	#Noisy test inputs	#Clusters
MNIST	LeNet-5	8055	1215	0.64	0.68	58	85
MNIST	LeNet-1	9754	1542	0.71	0.74	72	137
Fashion-MNIST	LeNet-4	5636	1157	0.65	0.53	101	141
Cifar-10	12-layer ConvNet	1173	1707	0.71	0.62	56	187
Cifar-10	ResNet20	2191	1384	0.75	0.63	78	177
SVHN	LeNet-5	151	3009	0.69	0.59	213	147

Table 2: Fault estimates across datasets and models

3.3.1.4 Evaluation

As with any clustering algorithm, there are several hyperparameters to fine-tune to obtain the best clustering results. Such hyperparameters include for example, the minimum distance that controls how tightly UMAP is allowed to pack points together, the number of neighbours to consider as locally connected in UMAP and the minimum size of clusters in HDBSCAN. We tried several hyperparameter configurations and selected the best configurations based on both manual and metric-based evaluations. For the latter, we relied on two standard metrics to evaluate the clusters, which are the Silhouette score [69] and the Density-Based Clustering Validation (DBCV) [70] metric.

The Silhouette score is one of the state-of-the-art clustering evaluation metrics that compare inter- and intra-cluster distances. It varies between minus one and one. The closer to one, the better the clustering. A score near zero represents clusters with inputs very close to the decision boundary of the neighboring clusters. A negative score generally indicates that the inputs are assigned to the wrong clusters.

We also relied on the DBCV metric to evaluate the generated clusters. This metric is dedicated to density-based clustering algorithms and assesses clustering quality based on the relative density connection between pairs of inputs. It evaluates the within- and between-cluster density connectedness [71]. Similar to Silhouette, DBCV generates scores between -1 and 1 [70]. High-density within clusters and low-density between clusters lead to high DBCV scores, indicating better clustering results.

We selected the configuration with the best Silhouette and DBCV scores. We further evaluated the generated clusters by performing a manual evaluation. First, we tried to check the content of the clusters to see whether their inputs were similar to or shared some features that might have led to mispredictions by the DNN model. Because of the large number of mispredicted inputs, an exhaustive manual inspection of the clusters was impractical. Therefore, we relied on generating the features' heatmaps related to each cluster to better visualize and assess the quality of the clusters. Figures 4, 5, 6 and 7 illustrate four representative examples of heatmaps where rows correspond to the inputs' ids in one cluster, columns refer to their features and colours encode the features' values. As we observe in Figures 4, 6 and 7, within a cluster, well-clustered inputs share common patterns in terms of the features' distribution while ill-clustered inputs (such as noisy inputs) do not, as is visible in Figure 5. Based on our manual analysis of the final selected clusters, we observed that most of them look like the first three figures and share common patterns. We therefore conclude that the mispredicted inputs inside each cluster are similar and share common characteristics (features), potentially causing mispredictions.

Table 2 describes the final clusters that we generated for the different datasets and models that we used in our experiments.
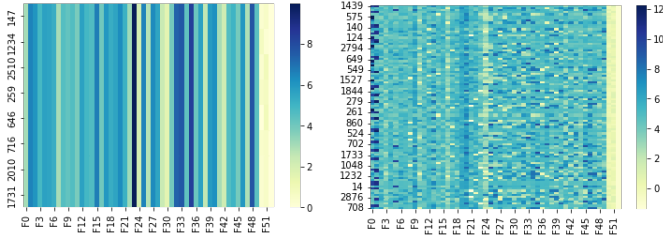
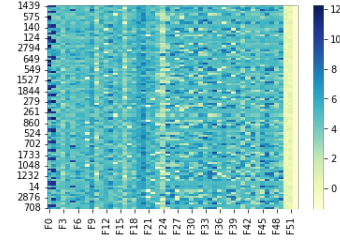Figure 4: Heatmap example 1 related to a final cluster

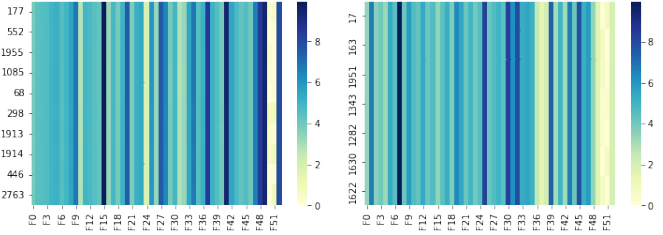Figure 5: Heatmap example 2 related to noisy cluster

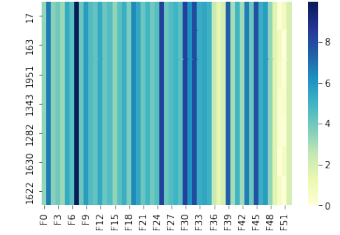Figure 6: Heatmap example 3 related to a final cluster

Figure 7: Heatmap example 4 related to a final cluster

We observe that the number of noisy inputs (inputs that do not belong to any cluster) is not large compared to the total number of mispredicted inputs. We decided to delete them from the sets of mispredicted inputs in all the following experiments because (1) they do not belong to any cluster and cannot therefore be associated with faults as we defined them, and (2) the minimum number of detected faults in the studied DNN models can thus be assumed to correspond to the number of clusters.

3.3.1.5 Validation

As mentioned earlier, we followed an approach similar to existing work [25], [61] to estimate faults in DNNs. The authors conducted an empirical study on six DNNs to validate the clustering-based fault estimation approach. They retrained the DNNs using the original training set and subsets selected from each identified cluster (i.e. fault), which led to a significant improvement in the models' accuracy, thus demonstrating the usefulness of clustering. To further validate the identified faults (clusters) in our work, we followed a finer-grained validation method which aimed to prove that (1) inputs in the same cluster tend to be mispredicted due to the same fault, and (2) inputs belonging to different clusters are mispredicted because of distinct faults.

Inputs that belong to the same cluster are mispredicted due to the same fault in the DNN model. If inputs within one cluster are mispredicted due to the same fault, retraining the model with a subset of the cluster should help fix the model with respect to that fault. We verified this hypothesis for each fault-related cluster C_i where $i \in [1, 2, ..., n]$ and n is the total number of the identified faults (clusters), by retraining the original DNN model under test with a retraining dataset consisting of the original training set and 85% of randomly selected mispredicted inputs inside C_i. The original training set was reused to prevent any reduction in model accuracy [25], [61], [72]. We then tested the retrained

model on the remaining 15% of inputs in C_i. We repeated this process five times (as we randomly selected inputs from cluster C_i) and measured the average accuracy of the retrained models when tested with 15% of the remaining inputs from cluster C_i. If clusters included mispredictions caused by the same fault, the retraining process was expected to alleviate the cause of input mispredictions in C_i and thus significantly improve the accuracy of the model for images in that particular cluster. We did not expect a perfect model with no mispredictions because we did not have a large number of mispredicted inputs in each cluster to retrain the model. Moreover, clustering was not expected to be perfect but only an approximation of faults. Because clusters did not have the same size and were not all large enough to enable this analysis, our analysis focused on the ten largest clusters across datasets and models. Due to the high computational expense of our experiments, we used two models, 12-layer ConvNet and ResNet20, to validate the fault-estimation method. Table 3 shows the accuracy of the retrained 12-layer ConvNet and ResNet20 models when tested on the ten largest clusters in each of their corresponding datasets. By definition, the accuracy of the original model on all cluster images was zero because they were all mispredicted. As shown in Table 3, there was significant improvement in the models' accuracy, with an average equal to 66.64% in 12-layer ConvNet and 69% in ResNet20.

The results therefore suggested that test inputs belonging to the same cluster are indeed mispredicted due to the same faults, thus supporting the hypothesis underlying our method of counting faults.

Inputs belonging to different clusters are mispredicted due to distinct faults. If the clusters represent distinct faults in the DNN, retraining the model with a subset of a cluster C_i should have a significant effect on the other images in C_i when compared to images in other clusters.
Consequently, to validate that inputs belonging to different clusters are mispredicted due to distinct faults, we test each of the previously retrained DNN models for each cluster C_i on the other clusters C_j where $j \neq i$. We measure the average accuracy of each of the retrained DNN models over all remaining clusters and report the results in Table 3. The retrained ResNet20 and 12-layer ConvNet models are significantly more accurate on the clusters for which they were retrained than on other clusters. Indeed, in 12-layer ConvNet, for example, the average accuracy for the latter is only 27.74% compared to 66.64% for the former. We therefore conclude that inputs belonging to different clusters tend to be mispredicted due to distinct faults. Although they are quite limited, we nevertheless observed improvements in model accuracy on clusters for which the model was not retrained. This can be expected because fixing one fault in the DNN model through retraining may also, to a more limited extent, fix other related faults, potentially improving the accuracy on other clusters. As acknowledged previously, our clustering is not perfect and although the obtained clusters' Silhouette and DBCV scores are high, they are not equal to one as shown in Table 2.

Model	Faults C_i	Accuracy on the retraining cluster C_i	Average Accuracy on the other clusters $C_{j \neq i}$
12-layer ConvNet	Cluster 1	80%	28.23%
	Cluster 2	60%	27.42%
	Cluster 3	55%	27.45%
	Cluster 4	71.40%	27.06%
	Cluster 5	66.70%	27.40%
	Cluster 6	60%	27.95%
	Cluster 7	58.33%	27.76%
	Cluster 8	67.70%	28.60%
	Cluster 9	66.30%	27.74%
	Cluster 10	61%	27.80%
Average		**66.64%**	**27.74%**
ResNet20	Cluster 1	77%	30%
	Cluster 2	75%	29%
	Cluster 3	74%	28%
	Cluster 4	62.5%	31%
	Cluster 5	65%	29%
	Cluster 6	56.5%	28.70%
	Cluster 7	78.40%	30.3%
	Cluster 8	75%	29.9%
	Cluster 9	62.5%	31%
	Cluster 10	64%	31%
Average		**69%**	**29.80%**

Table 3: The results of faults validation experiment on 12-layer ConvNet and ResNet20. In each row, we retrained the model under test on 85% of each cluster C_i. We report in the third column the accuracy of the retrained model on the remaining 15% of cluster C_i. The last column refers to the average of accuracies of the retrained models over all other fault clusters C_j $(j \neq i)$

3.3.2 RQ1. To what extent do the selected diversity metrics measure actual diversity in input sets?

To directly evaluate the capability of the selected metrics to measure diversity in input sets, we studied how diversity scores changed while varying, in a controlled manner, the number of image classes covered by the input sets. Classes characterize the content of images. For example, a set of images, sampled from the Cifar-10 dataset and containing the two classes *Car* and *Deer* is considered more diverse than a set containing only *cars*. We assumed that diversity scores should increase with the number of classes that are present in an input set.

Algorithm 1 describes at a logical level our experiment's procedure to answer RQ1. This procedure aims to increase the actual diversity of the content of image sets in a controlled manner and observe whether diversity metrics are sensitive to such changes. Given a certain dataset, we started by randomly selecting the first class C_i from the dataset in our experiment (Line 1). Then, we randomly sampled, with replacement, 20 input sets of size 100. Each of these input sets was sampled from the same class C_i (Lines 2-4). We measured the diversity scores for each initial input set (Lines 5-7). For each such input set, we incrementally increased the number of classes it covered by replacing some of its inputs with new ones from a new class $C_{k \neq i}$ while maintaining a uniform distribution across classes inside the samples. To do so, for each initial input set $Fset$, we randomly selected another class C_k that we wanted to add (Lines 8 and 10)

and randomly selected new inputs from the class C_k as a $Newset$ (Line 10-11). We randomly kept about $100/k$ inputs (k is the number of selected classes) of each existing class in $Fset$ (Line 12) and merged their inputs in $Fset$ with the newly selected ones in $Newset$ (Line 13). Finally, we measured the diversity scores for each input set (Lines 15-17) and repeated the process until we included images from all classes in the selected dataset (Line 9). The distribution of the diversity scores that are related to each metric using boxplots is depicted in Figure 8. Each boxplot illustrates the distribution of the diversity scores of 20 input sets of size 100, each with the same number of classes.

For example, when we evaluated Cifar-10, we selected 20 input sets of size 100. All the selected images inside each input set corresponded to the class *Deer*. For each selected input set, we measured the GD, NCD and STD scores. For each metric, we reported the distribution of the diversity scores related to these samples using boxplots, as depicted in Figures 8.a, 8.b and 8.c. We then increased the number of classes inside each sample by randomly replacing 50 images in *Deer* with new images from the *Truck* class. Each input set contained equal distribution of *Deer* and *Truck* images. We reported again the distribution of the diversity scores using boxplots. We repeated the process until we reached a total number of 10 classes inside the selected samples, while maintaining a uniform distribution across classes inside the input sets at each sampling iteration. As shown in Figure 8, we observed that GD outperforms NCD and STD as it exhibits a monotonic increase when

(a) Evolution of GD on Cifar-10 (b) Evolution of STD on Cifar-10 (c) Evolution of NCD on Cifar-10

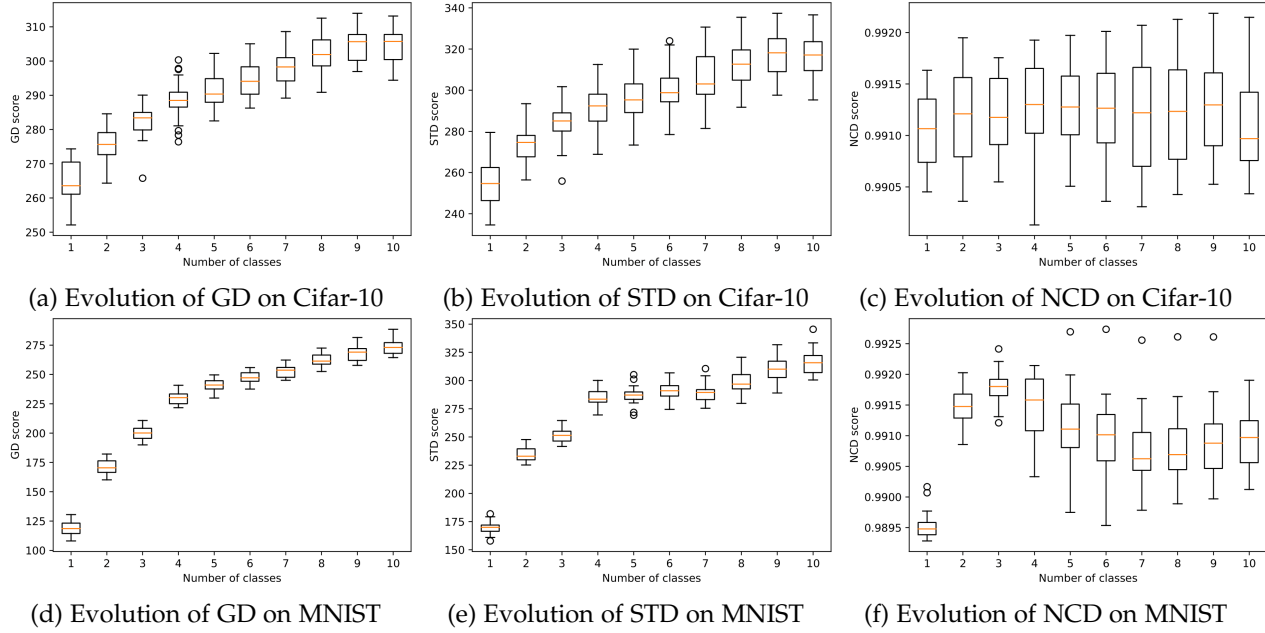(d) Evolution of GD on MNIST (e) Evolution of STD on MNIST (f) Evolution of NCD on MNIST

Figure 8: Evolution of the diversity scores for input sets from Cifar-10 and MNIST. Each boxplot shows the distribution of diversity scores of 20 input sets of size 100.

Algorithm 1: Experimental Procedure for RQ1

Input: C: set of n classes in the dataset
   $(C \leftarrow \{c_1, c_2, ..., c_n\})$
Output: $GDs, STDs, NCDs$
1 $c_i \leftarrow RandomClassSelect(1, C)$
2 **for** j in $\{1, 2, ..., 20\}$ **do**
3 $\quad$ $k \leftarrow 1$
4 $\quad$ $Fset \leftarrow RandomInputSelect(100, c_i)$
5 $\quad$ $GDs \leftarrow GD(Fset)$
6 $\quad$ $STDs \leftarrow STD(Fset)$
7 $\quad$ $NCDs \leftarrow NCD(Fset)$
8 $\quad$ $C \leftarrow C \setminus \{c_i\}$
9 $\quad$ **for** k in $\{2, ..., c_n\}$ **do**
10 $\quad\quad$ $c_k \leftarrow RandomClassSelect(1, C)$
11 $\quad\quad$ $NewSet \leftarrow RandomInputSelect(100/k, C_k)$
12 $\quad\quad$ $Fset \leftarrow Keep(100/k, Fset)$
13 $\quad\quad$ $Fset \leftarrow Merge(Fset, Newset)$
14 $\quad\quad$ $C \leftarrow C \setminus \{c_k\}$
15 $\quad\quad$ $GDs \leftarrow GD(Fset)$
16 $\quad\quad$ $STDs \leftarrow STD(Fset)$
17 $\quad\quad$ $NCDs \leftarrow NCD(Fset)$
18 **return** $GDs, STDs, NCDs$

---

increasing the number of classes inside the input sets. As shown in Figures 8.a and 8.d, the more diverse the input sets, the higher the GD in all datasets and models that we have evaluated. We observed a similar but noisier trend in STD. Using the example of STD scores for MNIST (Cf. Figure 8.e), we observe that these scores slightly decrease for samples embedding seven classes. A similar observation can be made in Cifar-10 when going from nine to ten classes (Cf. Figure 8.b).

Surprisingly, we found that NCD scores do not increase when input sets become more diverse. We also observed

that this diversity metric has low variability in the generated scores. As shown in Figures 8.c and 8.f, the range of the calculated mean NCD scores for the different input sets in the experiment is between 0.9895 and 0.9913. We should note that we have tried other types of features in our experiments with NCD to further assess the reliability of this metric in evaluating diversity. For this purpose, we followed one of the recommendations of Cohen *et al.* [31] and Cilibrasi *et al.* [39] and calculated the NCD scores of the input sets based on the raw images from MNIST. However, we obtained similarly poor results because the NCD score did not consistently increase when input sets became more diverse.

Besides its poor performance in measuring data diversity, we note that NCD is computationally expensive. It took approximately one hour to calculate the NCD score for one input set of size 100, suggesting another limitation regarding its applicability in testing DNN models. We conclude that, in our context, this metric is neither practical nor reliable in measuring data diversity and is therefore excluded from the rest of our study.

Note that the NCD metric's poor results may be due to the combination of feature inputs and compression tools that fail to generate accurate compression distances in our datasets. We therefore believe that NCD cannot be applied or function properly without careful selection of image formats and their associated feature representation and the compression tool, which is highly sensitive to these elements. Although we have tried several combinations of the aforementioned configurations based on existing prior works [31], [39], we aim in future work to investigate more combinations of feature images and dedicated compression tools to achieve better results for the NCD metric.

> **Answer to RQ1:** Geometric diversity and STD performed well in measuring actual data diversity in all the studied datasets. This is not the case with NCD, which we excluded from the following experiments.

3.3.3 RQ2. How does diversity relate to fault detection?

Dataset	Model	Test Set Size	Spearman	P-value
Cifar-10	12-layer ConvNet	100	8%	0.53
		200	17%	0.20
		300	-4%	0.78
		400	-9%	0.50
MNIST	LeNet-5	100	**-27%**	0.04
		200	4%	0.75
		300	-0.2%	0.98
		400	-10%	0.46

Table 4: Correlation between faults in subsets and clusters in the entire test set

We aimed to investigate whether higher diversity increases the fault detection capability of test sets. We randomly selected, with replacement, 60 samples of size $n \in \{100, 200, 300, 400, 1000\}$. For each sample, we calculated the corresponding diversity scores (GD and STD) and the number of faults (i.e. the number of covered clusters of mispredicted inputs). We calculated the Spearman correlation [73] between the diversity scores and the number of faults. The correlation results are reported in Tables 5, 6 and 7 for the different datasets and DNN models. The grey boxes in the table refer to statistically significant correlations (p-value $<= 0.05$). We chose to use the Spearman correlation because it measures the strength of a monotonic correlation between two variables, without making assumptions about the form of the relationship or data distributions [73].

Nonetheless, there is a potential confounding factor in the correlation between faults and diversity. If we were to apply clustering in the test dataset, we would expect higher diversity to lead to more clusters. If there is a high similarity in distribution between the entire test dataset and the subset of mispredicted inputs, the correlation between diversity and faults in subsets could be due to the presence of such a confounding factor. To verify this, we analyzed the correlation between the number of clusters in the entire test dataset that were covered and the number of faults (i.e. fault-related clusters) in subsets. A low correlation would indicate that such a threat is unlikely.

We applied the same HDBSCAN clustering process (section 3.3.1) to the entire testing dataset to obtain data clusters. We then used the previously selected 60 samples of size $n \in \{100, 200, 300, 400\}$ and calculated the number of faults (i.e. the number of covered clusters of only mispredicted inputs) and data clusters (i.e. the number of covered clusters of both correctly predicted and mispredicted inputs) inside each sample. Finally, we calculated the Spearman correlation between the number of faults and the number of covered data clusters in subsets. We performed this experiment using 12-layer ConvNet (with Cifar-10) and LeNet-5 (with MNIST). The correlation results are reported in Table 4. As shown in the table, we did not find any positive and statistically significant correlation between faults and data clusters. We

Dataset	Model	Metric	Test Set Size	Spearman	P-value
Cifar-10	12-layer ConvNet	GD	100	**29%**	0.02
			200	**32%**	0.01
			300	**25%**	0.05
			400	**31%**	0.02
			1000	**29%**	0.02
		STD	100	**26%**	0.05
			200	**26%**	0.05
			300	19%	0.14
			400	21%	0.11
			1000	**33%**	0.01
		LSC	100	8%	0.53
			200	4%	0.74
			300	0.5%	0.97
			400	5%	0.70
			1000	-5%	0.70
		DSC	100	2%	0.85
			200	18%	0.18
			300	3%	0.80
			400	-8%	0.55
			1000	24%	0.07
		NC	100	-22%	0.10
			200	5.3%	0.69
			300	0.3%	0.98
			400	22%	0.10
			1000	14%	0.28
		KMNC	100	0.51%	0.97
			200	14%	0.29
			300	12%	0.35
			400	-4%	0.76
			1000	19%	0.15
		NBC	100	15%	0.27
			200	5.6%	0.67
			300	7%	0.58
			400	-0.6%	0.96
			1000	11%	0.40
		TKNC	100	15%	0.27
			200	**36%**	0.005
			300	**28%**	0.031
			400	**27%**	0.04
			1000	0.2%	0.98
		SNAC	100	16%	0.22
			200	5%	0.70
			300	8%	0.52
			400	-2%	0.89
			1000	15%	0.24
MNIST	LeNet-5	GD	100	**34%**	0.009
			200	**26%**	0.04
			300	**33%**	0.01
			400	**37%**	0.004
			1000	**35%**	0.005
		STD	100	6%	0.67
			200	**26%**	0.04
			300	**34%**	0.01
			400	23%	0.07
			1000	13%	0.34
		LSC	100	28%	0.83
			200	12%	0.36
			300	24%	0.07
			400	**32%**	0.01
			1000	19%	0.14
		DSC	100	3%	0.80
			200	-10%	0.42
			300	19%	0.16
			400	30%	0.33
			1000	8%	0.53
		NC	100	-7%	0.58
			200	2%	0.90
			300	2%	0.85
			400	**29%**	0.03
			1000	**27%**	0.03
		KMNC	100	-3%	0.83
			200	5%	0.69
			300	-13%	0.34
			400	11%	0.40
			1000	19%	0.14
		NBC	100	2%	0.88
			200	-22%	0.09
			300	9%	0.53
			400	-2%	0.87
			1000	**31%**	0.0006
		TKNC	100	27%	0.06
			200	22%	0.08
			300	17%	0.19
			400	13%	0.31
			1000	**30%**	0.02
		SNAC	100	5%	0.72
			200	**-29%**	0.02
			300	3%	0.80
			400	-10%	0.45
			1000	**32%**	0.0004

Table 5: Correlation results between test criteria and DNN faults. The grey boxes refer to statistically significant correlations (p-value $<= 0.05$)

Dataset	Model	Metric	Test Set Size	Spearman	P-value
MNIST	LeNet-1	GD	100	**33%**	0.01
			200	**39%**	0.002
			300	**28%**	0.04
			400	**33%**	0.01
			1000	**29%**	0.03
		STD	100	6%	0.67
			200	**26%**	0.04
			300	20%	0.15
			400	12%	0.36
			1000	16%	0.21
		LSC	100	-6%	0.62
			200	23%	0.08
			300	18%	0.19
			400	12%	0.35
			1000	16%	0.22
		DSC	100	13%	0.33
			200	-21%	0.10
			300	-25%	0.07
			400	17%	0.19
			1000	6%	0.67
		NC	100	6%	0.67
			200	22%	0.10
			300	24%	0.08
			400	-6%	0.67
			1000	3%	0.82
		KMNC	100	22%	0.10
			200	13%	0.32
			300	2%	0.86
			400	-3%	0.84
			1000	19%	0.15
		NBC	100	**-30%**	0.02
			200	**32%**	0.01
			300	20%	0.15
			400	8%	0.52
			1000	-13%	0.33
		TKNC	100	10%	0.46
			200	2%	0.89
			300	2%	0.88
			400	19%	0.15
			1000	13%	0.31
		SNAC	100	**-28%**	0.03
			200	**27%**	0.03
			300	25%	0.07
			400	4%	0.76
			1000	-13%	0.32
Fashion-MNIST	LeNet-4	GD	100	**31%**	0.02
			200	**32%**	0.01
			300	**30%**	0.02
			400	**26%**	0.05
			1000	**28%**	0.03
		STD	100	10%	0.48
			200	10%	0.43
			300	3%	0.82
			400	9%	0.48
			1000	18%	0.18
		LSC	100	14%	0.31
			200	10%	0.44
			300	11%	0.42
			400	9%	0.48
			1000	7%	0.61
		DSC	100	15%	0.28
			200	-19%	0.14
			300	-12%	0.37
			400	-14%	0.28
			1000	8%	0.55
		NC	100	-6%	0.66
			200	25%	0.06
			300	6%	0.63
			400	22%	0.10
			1000	10%	0.46
		KMNC	100	18%	0.18
			200	20%	0.13
			300	**36%**	0.01
			400	**27%**	0.03
			1000	24%	0.07
		NBC	100	-0.2%	0.99
			200	2%	0.88
			300	15%	0.24
			400	1%	0.93
			1000	25%	0.06
		TKNC	100	22%	0.09
			200	13%	0.31
			300	18%	0.18
			400	6%	0.66
			1000	13%	0.32
		SNAC	100	1%	0.96
			200	2%	0.86
			300	16%	0.22
			400	1%	0.95
			1000	25%	0.06

Table 6: Correlation results between test criteria and DNN faults. The grey boxes refer to statistically significant correlations (p-value <= 0.05)

Dataset	Model	Metric	Test Set Size	Spearman	P-value
Cifar-10	ResNet20	GD	100	**26%**	0.05
			200	**31%**	0.01
			300	**37%**	0.004
			400	**29%**	0.03
			1000	**28%**	0.03
		STD	100	16%	0.22
			200	**28%**	0.03
			300	**34%**	0.01
			400	20%	0.13
			1000	**28%**	0.03
		LSC	100	16%	0.24
			200	**33%**	0.01
			300	-10%	0.43
			400	9%	0.50
			1000	-6%	0.63
		DSC	100	-3%	0.82
			200	-14%	0.28
			300	1%	0.98
			400	9%	0.48
			1000	18%	0.16
		NC	100	12%	0.38
			200	16%	0.23
			300	-6%	0.64
			400	6%	0.66
			1000	7%	0.60
		KMNC	100	4%	0.79
			200	13%	0.32
			300	18%	0.17
			400	**26%**	0.05
			1000	**33%**	0.01
		NBC	100	13%	0.31
			200	13%	0.32
			300	15%	0.25
			400	12%	0.37
			1000	7%	0.61
		TKNC	100	-1%	0.96
			200	1%	0.95
			300	16%	0.23
			400	22%	0.10
			1000	1%	0.93
		SNAC	100	16%	0.24
			200	13%	0.34
			300	19%	0.15
			400	11%	0.40
			1000	11%	0.41
Fashion-MNIST	LeNet-4	GD	100	**27%**	0.04
			200	**27%**	0.03
			300	**27%**	0.04
			400	**33%**	0.01
			1000	**30%**	0.02
		STD	100	**27%**	0.04
			200	16%	0.21
			300	20%	0.13
			400	23%	0.08
			1000	6%	0.65
		LSC	100	-5%	0.72
			200	20%	0.12
			300	10%	0.45
			400	**27%**	0.04
			1000	3%	0.83
		DSC	100	-17%	0.20
			200	-9%	0.50
			300	-5%	0.69
			400	4%	0.75
			1000	8%	0.57
		NC	100	9%	0.49
			200	6%	0.64
			300	-2%	0.89
			400	-6%	0.67
			1000	-6%	0.63
		KMNC	100	15%	0.25
			200	22%	0.09
			300	-2%	0.89
			400	**26%**	0.05
			1000	7%	0.61
		NBC	100	-6%	0.64
			200	13%	0.31
			300	16%	0.22
			400	11%	0.41
			1000	-6%	0.64
		TKNC	100	14%	0.29
			200	4%	0.77
			300	10%	0.46
			400	9%	0.50
			1000	-16%	0.23
		SNAC	100	-6%	0.64
			200	13%	0.31
			300	-16%	0.22
			400	11%	0.41
			1000	-6%	0.64

Table 7: Correlation results between test criteria and DNN faults. The grey boxes refer to statistically significant correlations (p-value <= 0.05)

therefore conclude that there is no confounding factor in our correlation study between diversity and faults, thus giving us more confidence in the cause-effect relationship underlying the observed correlations. These results also suggest that correctly predicted inputs belong to separate clusters from fault-related clusters (i.e. clusters of mispredicted inputs) in general. They provide evidence that mispredicted inputs belong to the same cluster and share common characteristics that are different from the ones shared by correctly predicted inputs.

In our correlation experiment between diversity and faults, we evaluated a total of 60 configurations related to diversity metrics (6 models & datasets x 2 diversity metrics x 5 input sizes). As mentioned in Tables 5, 6 and 7, we found that GD outperforms STD in terms of fault-revealing capabilities as we observed that there was a positive, statistically significant correlation between GD and faults in all configurations (30/30). Furthermore, they were consistent across all the studied models, datasets and input set sizes. On the other hand, we found that STD had a positive significant correlation with faults in only ten configurations. These results were expected because, in RQ1, GD showed better performance in measuring actual data diversity than STD.

We expected to have a moderate correlation between diversity and faults because we relied on a clustering approach to approximate faults in DNNs (section 3.3.1). Such correlation is expected to be higher if we have a more straightforward method to identify faults in DNNs.

Nevertheless, the obtained results clearly indicate that GD can be used to effectively guide DNN testing by devising input sets with maximum diversity to increase their fault-revealing capabilities. Let us recall that GD also has the practical advantage of being black-box, as opposed to state-of-the-art DNN coverage metrics [12], [13], [10], [11], which require access to the internals of DNN models or their training sets.

> **Answer to RQ2:** There is a positive and statistically significant correlation between GD and faults in DNNs. GD is more frequently correlated to faults than STD. Consequently, GD should be used as a black-box approach to guide the testing of DNN models.

3.3.4 RQ3. How does coverage relate to fault detection?

Similar to the previous section on diversity, in this research question, we aim to study the correlation between state-of-the-art coverage criteria and faults in DNNs. Our goal is to understand how they compare to diversity in this respect. Based on three factors, we selected the following two coverage criteria: Likelihood-based Surprise Coverage (LSC) and Distance-based Surprise Coverage (DSC) [10]. First, we retained criteria that were recently published in the literature. We also chose those that (1) have been compared to other coverage criteria, and (2) showed better performance in guiding the testing of DNN models.

We selected coverage metrics that we could apply and replicate on our models and datasets. The first two factors

yielded four coverage metrics: Likelihood-based Surprise Coverage, Distance-based Surprise Coverage, Importance-Driven Coverage (IDC) [11] and Sign-Sign Coverage (SSC) [14]. However, we could not apply IDC and SSC on our datasets and models. We got several execution errors[2] when we tried to compute IDC on the 12-layer ConvNet and LeNet models. For SSC, we obtained different results from the original paper [14] when we applied this metric on LeNet-1, and encountered several execution errors in the remaining models. Therefore, we excluded these metrics from our research and only studied LSC and DSC in the correlation between coverage and fault detection in DNNs. In addition to this criteria, we included basic and widely used criteria such as Neuron Coverage (NC) [12] and Deep-Gauge [13] coverage metrics. We considered the following metrics related to DeepGauge: k-Multisection Neuron Coverage (KMNC), Neuron Boundary Coverage (NBC), Top-K Neuron Coverage (TKNC) and Strong Neuron Activation Coverage (SNAC). We describe these metrics and their limitations in Section 6.

To investigate the relationship between coverage and fault detection, we ran the same experiment as in RQ2 and evaluated the same selected subsets. We calculated the different coverage scores for all subsets. For LSC and DSC, we used the same recommended settings for hyperparameters (e.g. upper bound, lower bound, number of buckets) as in the original paper [10] and other existing papers in the literature [56]. We used the same hyperparameters that were recommended in the literature [13] for NC, KMNC and TKNC. We used the activation threshold of 10% for NC and fixed the number of buckets K to three for TKNC and 1,000 for KMNC; this is for the different models and datasets in our experiments. We calculated the Spearman correlation between each coverage criterion and the number of faults. The results are reported in Tables 5, 6, 7 and 8.

We considered 30 configurations per metric (6 models & datasets x 5 input sizes). Further, we accounted for a total of 210 configurations related to the coverage criteria (6 models & datasets x 7 criteria x 5 input sizes). As depicted in Table 8, out of 30 different configurations per metric, the distributions of positive, statistically significant correlations to faults are as follows: 5 for KMNC, 4 for TKNC, 3 for LSC, and 2 for NC, NBC and SNAC. DSC, however, did not show any statistically significant correlation with faults in any of the datasets and models.

In general, we conclude that significant correlations between coverage and faults are rare in the configurations of models and datasets that we used. None of the studied coverage metrics consistently showed statistically significant correlations across all the models, datasets and input set sizes. For example, we found that LSC is positively correlated to faults in only three out of 30 configurations related to LeNet-5 and ResNet20. However, we did not find any statistically significant correlation for this metric with LeNet-1, LeNet-4 and 12-layer ConvNet.

Our findings raise questions about the usefulness of the selected coverage criteria for enabling effective DNN testing in fault detection. These results confirm, from a different

---

2. Authors have been contacted but the execution errors have not been resolved.

Dataset	Model	GD	STD	LSC	DSC	NC	KMNC	NBC	TKNC	SNAC
MNIST	LeNet-1	5/5	1/5	0/5	0/5	0/5	0/5	1/5	0/5	1/5
MNIST	LeNet-5	5/5	2/5	1/5	0/5	2/5	0/5	1/5	1/5	1/5
Fashion-MNIST	LeNet-4	5/5	0/5	0/5	0/5	0/5	2/5	0/5	0/5	0/5
Cifar-10	12-layer ConvNet	5/5	3/5	0/5	0/5	0/5	0/5	0/5	3/5	0/5
Cifar-10	ResNet20	5/5	3/5	1/5	0/5	0/5	2/5	0/5	0/5	0/5
SVHN	LeNet-5	5/5	1/5	1/5	0/5	0/5	1/5	0/5	0/5	0/5
Total		**30/30**	**10/30**	**3/30**	**0/30**	**2/30**	**5/30**	**2/30**	**4/30**	**2/30**

Table 8: Number of positive statistically significant correlations between testing criteria and faults

angle, many recent studies [16], [18], [55] that questioned the reliability of such coverage criteria to guide the testing of DNN models. A central concern raised by these articles is whether such coverage metrics relate to the model's behaviour and its decision results. Our results suggest that this relationship is, at best, weak.

> **Answer to RQ3:** In general, significant positive correlations between coverage and faults are rarely based on the configurations and datasets used in our experiments. Coverage metrics are not a good indicator of fault detection.

3.3.5 RQ4. How do diversity and coverage perform in terms of computation time?

We aimed to compare the computation times of the selected diversity metrics and coverage criteria and assessed how they scaled with the sizes of test sets. For this purpose, we randomly selected, with replacement, 60 samples of size n $\in$ {100, 200, 300, 400}. We calculated, for each sample, diversity and coverage scores, and measured their respective computation times. Because we found in RQ2 that GD outperforms STD in correlation to faults, in the rest of this work we only used the GD metric to calculate diversity in input sets. For GD, we accounted for the sum of the following two computation times: (1) calculation of diversity based on the extracted features; and (2) the pre-processing time that is required to extract features with the VGG16 model. We report in Figures 9 and 10 the change in computation times for ResNet20 and LeNet-5 as we increased the sizes of the input sets. We observed that for both diversity and coverage metrics, computation time is linear with test set sizes up to 400. Both types of metrics are not computationally expensive. For example, the computation time related to diversity and coverage metrics in MNIST and LeNet-5 varies between 0.4 and 49 seconds for samples of size 400. We observed that KMNC, SNAC and NBC have the same computation time. Further, they are the most computationally expensive metrics. For example, KMNC, SNAC and NBC are approximately 25 times more computationally expensive than the other metrics in ResNet20. However, we found that GD generally outperforms most of the coverage metrics in computation time, except for NC and TKNC in LeNet-5. We used the Wilcoxon signed-rank test [74] to assess the statistical significance of the difference between GD's computation time and the other testing criteria. We found that GD statistically outperformed all coverage met-

rics in computation time. For example, it was three-to-five times faster to compute GD than LSC and DSC. Although absolute differences are a matter of seconds, such computations, in the context of test selection or minimization, may be performed thousands of times and thus become practically significant. We studied the distributions of the computation times for diversity and coverage metrics and analyzed their variations. Due to space limitations, we included the related boxplots in our replication package [75]. We found that the distributions of the computations times related to GD showed less variation than the studied coverage metrics for samples of the same size. For example, GD showed less variation than LSC and DSC for samples of the same size because the GD metric depends on the calculation of the determinant of a fixed-size feature matrix, while LSC and DSC depend on a search mechanism for the nearest inputs in the training set. Search time may vary from one sample to another, and therefore leads to increased variance in computation time.

Because GD is black-box, its computation time only depends on the used dataset and is not affected by the complexity of the DNN model (e.g. number of layers and neurons). In contrast, the computation time of white-box coverage metrics is highly sensitive to such complexity.

> **Answer to RQ4:** Both diversity and coverage metrics are not computationally expensive (seconds). However, GD significantly outperforms coverage criteria. In application contexts, such as test case selection and minimization, and based on searches in which we perform many test set evaluations, this difference can become practically significant.

3.3.6 RQ5. How does diversity relate to coverage?

We aimed to study the relationship between diversity and coverage to assess if diverse input sets increase the coverage of DNN models. Conversely, increasing coverage should, in theory, increase diversity. Although the results of previous research questions make it unlikely for such correlations to be strong, this needed to be investigated.

We ran the same experiment as in RQ2 and RQ3 and used the same selected subsets. For each subset, we calculated the geometric diversity and coverage scores and measured the Spearman correlation between each pair of diversity and coverage metrics. We evaluated a total of 175 configurations (1 diversity metric x 7 coverage criteria x 5 models & datasets x 5 test set sizes). We include all the
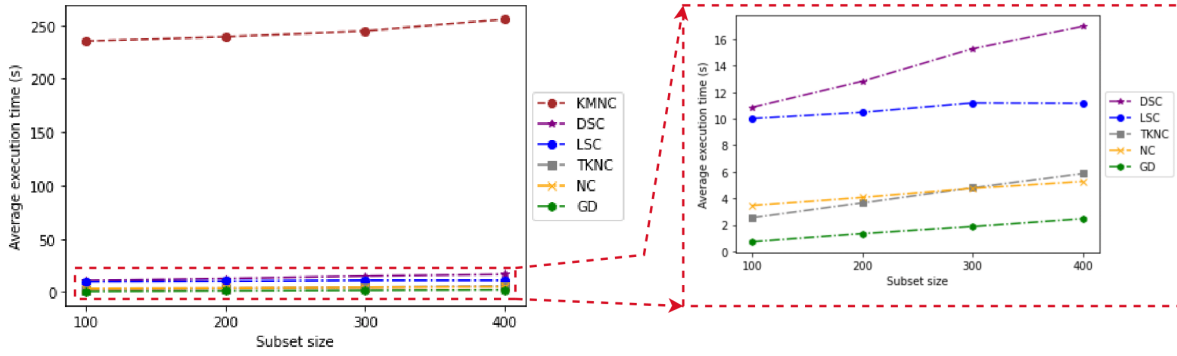
Figure 9: Computation time for diversity and coverage metrics with Cifar-10 and ResNet20
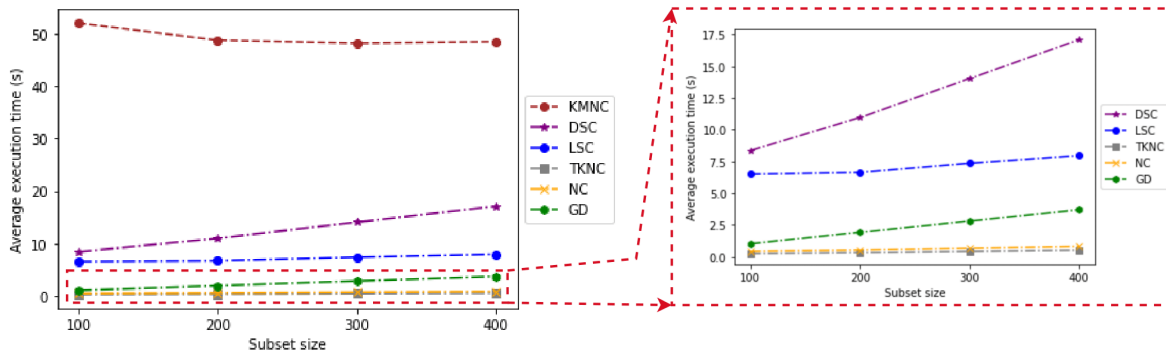

Figure 10: Computation time for diversity and coverage metrics with MNIST and LeNet-5

results in Appendix A, and they are available online [75]. Out of 175 configurations, only 13 correlations were positive and statistically significant. For example, the only positive correlation (46%) between GD and LSC was for input sets of size 400 from Cifar-10 using ResNet-20. Furthermore, the only statistically significant correlation (-35%) between GD and DSC was for input sets of size 300 from Fashion-MNIST using LeNet-4. For NC, we found only three statistically significant correlations in three different models (LeNet-4, LeNet-5 and ResNet-20). Although we found six statistically significant correlations between KMNC and GD, these results were not consistent across all models and input set sizes. Additionally, the only statistically significant correlation (48%) between GD and NBC was for input sets of size 1,000 from MNIST using LeNet-5. Finally, for TKNC and SNAC, we found only two statistically significant correlations for each metric with GD. To summarize, most configurations (159) show no significant correlations between diversity and coverage metrics, which suggests that, in general, diversity and coverage in DNN models are not correlated. In other words, diverse input sets do not necessarily increase the coverage of DNN models and higher coverage does not systematically lead to higher diversity. These results are also consistent with our previous observations in RQ3 and RQ4, where we found that while geometric diversity is correlated to fault detection, coverage is not.

> **Answer to RQ5:** In general, for most configurations there is no significant correlation between diversity and coverage in DNN models.

4 DISCUSSION AND RECOMMENDATIONS

We should note that our correlation results between testing criteria (diversity and coverage metrics) and faults are consistent across different datasets and DNN models. Based on our experiments, we show that studying the diversity of the features embedded in test input sets is more reliable as a basis to guide DNN testing than considering the coverage of their hidden neurons. We show that geometric diversity is potentially more effective than existing coverage metrics in guiding the testing of DNN models. This metric requires neither knowledge of the model under test nor access to the training set. Further, it does not require execution of the test input nor reliance on the output of the DNN model under test. It is therefore a practical, black-box approach that can be used to guide the testing of DNN models. Although the results are encouraging, we only investigated geometric diversity with DNN models using images as input. Further experiments should be conducted to evaluate the performance on other input data types, such as audio and text data. We will therefore explore appropriate feature extraction models to represent new data types with feature vectors used by our diversity metrics (mainly diversity and STD metrics, because NCD supports, by default, any data

type). Because diversity metrics are black-box and do not depend on the type of DNN model, we also aim in future work to consider other DNN models for regression and multi-classification tasks to further generalize our results.

In our experiments, we were surprised to see only a few significant correlations between coverage and faults across all the models and datasets we evaluated.

We selected both widely used coverage criteria in the literature and the best coverage metrics in published results and reproducibility (section 3.3.4). Nevertheless, coverage showed poor performance as an indicator of detected faults in DNNs. In traditional software, one of the potential reasons for the effectiveness of coverage criteria is that they rely on the logical structure of the system's source code. However, the decision logic in DNNs is not explicit, which makes the definition and usage of coverage criteria more challenging in DNNs. Also, in traditional software, relying on diverse test cases tends to increase code coverage and the fault-detection capabilities of test suites [19], [76]. In contrast, we show that in DNN testing diverse test input sets do not lead to increased DNN coverage but, at least for geometric diversity, lead to more fault detection in the DNN model.

Furthermore, traditional software systems and DNNs are fundamentally different with respect to the notion of fault and their detection. Given a test input, in general, we detect faults in software by comparing the actual test output to the expected output. If there is a mismatch, we consider this to be a failure, and we can debug the system using various fault localization techniques [77], [78], [79] to identify faulty statement(s). However, in DNN models, the notion of fault is elusive because of the black-box nature of DNN models. If the DNN model mispredicts an input, we consider this to be a failure, but debugging and localizing faults in the DNN causing such failure is challenging because there is no explicit and interpretable decision logic. This is also why DNNs are usually fixed through retraining [25]. Because it is common for many mispredicted inputs to be caused by the same problems in the DNN model [25], and because we cannot directly identify root causes, we relied on a clustering-based approach to group similar mispredicted inputs and therefore relied on the number of these clusters to approximate fault-counting in our experiments. Our clustering relied on a density-based clustering algorithm that grouped similar mispredicted inputs based on their (image) features and their misprediction behaviour (pairs of actual and mispredicted classes). Our fault estimation approach is therefore not "*complete*" because we only considered faults with a sufficient number of observed mispredictions to be grouped into a cluster. The others were considered noisy inputs by our clustering approach. In other words, we obtained a good approximation of commonly occurring faults, which underestimates the total number of faults because we do not account for noisy inputs. Because it is more important for testing approaches to detect faults, leading to more frequent mispredictions, this is practically acceptable. As described in Table 2 and to reinforce this point, the number of noisy inputs is very small compared to the total number of mispredicted inputs. We acknowledge that although our retraining-based validation and evaluation results show promise, they only indirectly validate our fault model since

there is no direct way to check its fault estimation accuracy. More research is needed to investigate alternative ways to enable fault detection comparisons in experiments involving DNN models.

Our study of the computation time of diversity and coverage metrics was generic and did not target a specific DNN testing scenario (e.g. selection, generation, minimization). However, this was intentional, as we wish to provide general insights into the computational complexity of coverage and diversity metrics. We showed that both types of metrics are not computationally intensive and that GD is generally three-to-five times faster to compute than the studied coverage metrics. However, whether such differences practically matter and to what extent depends on how frequently they are computed in a given application context. Some coverage-based test selection approaches entail computing the coverage score only once for each test input and selecting the test inputs with the highest coverage score. In contrast, in a typical diversity-based test selection approach, where the goal is to select a set of diverse test inputs, the GD score may be computed many times for selected subsets when, for example, the GD metric is used to drive a search algorithm. Finally, we aim in future work to further investigate the computation time of coverage and diversity metrics when used in specific DNN testing scenarios, such as test set selection, generation and minimization.

To summarize, before using any testing criteria to support a particular test scenario for DNNs (e.g. test selection, minimization and generation), one should investigate the correlation between the targeted testing criteria and faults. This is our main motivation in this work, where we investigate the relationship between testing criteria (coverage and diversity metrics) and faults in DNNs. The stronger the correlation between testing criteria and fault detection, the better. The practical implications of our results suggest that one should not rely on coverage, as currently defined, to guide DNN testing if the objective is to detect as many faults as possible. Alternatively, we show that geometric diversity has strong potential as an alternative. It outperforms existing coverage metrics in fault-revealing capability, applicability (as it is black-box) and computation time. We therefore recommend investigating its practical use in testing DNNs to guide the selection, minimization or generation of test input sets.

5 THREATS TO VALIDITY

In this section, we discuss the different threats to the validity of our study and describe how we mitigated them. **Internal threats to validity** concern the causal relationship between the treatment and the outcome. We reimplemented three diversity metrics because their source code was unavailable (GD and STD) or not applicable on our datasets (NCD). Consequently, an internal threat to validity might be related to our implementations. To mitigate this threat, we carefully checked our code and its conformance to the original papers in which they were published. We also verified the correctness of our implementation of the NCD metric by comparing our results with an existing implementation[3] that supported the calculation of the NCD score only for pairs of

---

3. https://github.com/simonpoulding/DataGenerators.jl

images or *txt* files. In **RQ1**, we tested, through a controlled experiment, the reliability of the selected diversity metrics in measuring actual data diversity and excluded the metrics that failed the test.

As we were targeting black-box diversity metrics, we needed to rely on a feature extraction model to build our feature matrix. Therefore, an internal threat to validity might be caused by a low-quality representation of inputs. To mitigate this threat, we relied on VGG16, which is one of the most-used, accurate, state-of-the-art feature extraction models. Furthermore, this DNN model has been pre-trained on the extremely large ImageNet dataset, which contains over 14 million labelled images belonging to 22,000 categories. Further, the configuration of the different hyperparameters in our study may induce additional internal threats to validity. We mitigate this threat in two ways: (1) for coverage metrics hyperparameters, we made use of the original papers' hyperparameter values [10] for each dataset and model that we used; and (2) for fault estimation hyperparameters (clustering), we tried more than 500 configurations related to HDBSCAN and UMAP for each of the datasets and models that we evaluated. To reduce potential bias, we evaluated the configurations' results by using two clustering evaluation metrics (section 3.3.1) and by visualizing heatmaps.

A final internal threat to validity is related to randomness when sampling test inputs. We addressed this issue by repeating such sampling multiple times while considering different input set sizes and different datasets and models. **Construct threats to validity** concern the relation between the theory and the observations. To study the effectiveness of a given test criterion in guiding DNN testing, we relied on a clustering-based approach to estimate detected faults in DNNs. It is a potential threat to construct validity because this estimate may not be sufficiently accurate. If that is the case, correlations with diversity and coverage might appear weaker than they actually are. Alternatively, relying on misprediction rates is, as previously discussed, misleading, because in practice, many similar mispredicted inputs typically result from the same problems in the DNN model. Accounting for numerous redundant test inputs would blur our correlation analysis, an effect we observed in our study. Further, we relied on a density-based clustering algorithm that is capable of grouping similar inputs in clusters of arbitrary shapes, as opposed to other types of clustering algorithms, such as k-means and hierarchical clustering, which assume that clusters are convex. Next, we clustered similar mispredicted inputs based on their (image) features and misprediction behaviour, thus relying on what semantically distinguishes images. Finally, we quantitatively and qualitatively assessed the obtained clusters to group test inputs with similar characteristics.

Reliability threats to validity concern the replicability of our study results. We relied on publicly available models and datasets and provided all the materials required to replicate our study results online [75]. This includes the set of all selected samples in the experiments and the different configurations we used for all the selected testing criteria. **Conclusion threats to validity** concern the relation between the treatment and the outcome. We relied on the Spearman correlation because it does not rely on assumptions about the data set distributions or on the shapes of the relationships, except for the latter being monotonic.

External threats to validity concern the generalizability of our study. We mitigated this threat by using four large datasets and five widely used and architecturally different DNN models. Further, in each of our experiments, we evaluated many samples and input set sizes. The selected coverage metrics may not be representative of all existing coverage criteria. However, we selected the best metrics based on their published results and our ability to reproduce their results.

6 RELATED WORK

The work presented in this paper relies on concepts related to test diversity, black-box testing and model coverage in DNNs. In this section, we provide an overview of existing coverage metrics for DNN models. We also describe existing work on black-box DNN testing and studies making use of diversity to guide testing of DNNs and traditional software systems.

6.1 Test Coverage Criteria for DNNs

Several coverage metrics have been proposed in the literature. The first attempt was carried out by Pei *et al*. [12] who proposed the Neuron Coverage (NC) metric for test inputs, which is defined as the proportion of activated neurons (neurons whose activation value is above the defined threshold) over all neurons when all available test inputs are supplied to a DNN. However, several studies [14], [15] have shown that 100% neuron coverage is easy to achieve with a small set of inputs, and consequently is going to limit the applicability of such metric when testing DNNs.

Ma *et al*. [13] proposed DeepGauge, a set of DNN coverage metrics. They introduced K-Multisection Neuron Coverage (KMNC), Neuron Boundary Coverage (NBC) and Strong Neuron Activation Coverage (SNAC). KMNC partitions the ranges of neuron activation values into K buckets based on training inputs and counts the number of total covered buckets by a given test input set. NBC measures the ratio of corner-case regions that have been covered. Corner-case regions correspond to the activation values that are beyond the activation ranges observed during training. SNAC measures how many upper corner-cases have been covered. Upper corner-cases correspond to neuron activation values that are greater than the activation ranges observed during training. The authors showed that input tests generated by adversarial methods increase coverage in terms of their metrics. However, they did not study how these metrics relate to DNN mispredictions using natural inputs.

Inspired by the MC/DC test coverage in traditional software testing, Sun *et al*. [14] proposed four coverage metrics that account for the causal relationship between neurons in neighbouring layers of a DNN model. These metrics were used to guide the generation of test inputs using adversarial methods to test the robustness of DNN models.

Kim *et al*. [10] proposed two coverage criteria called Likelihood-based Surprise Coverage (LSC) and Distance-based Surprise Coverage (DSC). These criteria are based on an analysis of how surprising test sets are given the training set. LSC uses Kernel Density Estimation (KDE) [80] to estimate the likelihood of seeing a test input during the

training phase. DSC relies on the calculation of Euclidean distances between vectors that correspond to (1) the neurons' activation values in inputs from the test set, and (2) the neurons' activation values in inputs from the training set. They argue that an input set that covers a wide and diverse range of surprise values is preferable to test and retrain a DNN model. They show that their metrics are correlated with existing coverage criteria [12], [13] when the diversity of inputs is increased. However, our study shows that there is no strong correlation between surprise adequacy coverage and diversity by using only natural inputs. We also show that there is no strong correlation between these coverage metrics and faults in DNNs. Another study conducted by Chen *et al.* [18] showed similar results with respect to DNN misprediction rates when using only natural inputs.

Gerasimou *et al.* [11] proposed the Importance-Driven Coverage (IDC) criterion to focus on the coverage of the most influential neurons in DNN predictions. They argue that IDC is sensitive to adversarial inputs and achieves higher values when applied to input sets that comprise diverse inputs. They also evaluated DeepGauge [10] and surprise adequacy coverage criteria [13] in their experiments and observed that IDC shows a similar increase in these coverage criteria when evaluated with test sets augmented with adversarial inputs.

Byun et al. [81] have recently proposed Manifold Combination Coverage (MCC), a black-box coverage metric for testing DNN models based on projecting test inputs onto a manifold space using a Variational AutoEncoder (VAE). This metric relies on manifold learning [82] that compresses a high-dimensional input space into a lower-dimensional manifold space. It then measures the coverage of test inputs within a subset in the manifold space to assess test thoroughness [81]. The authors compared the misprediction-revealing effectiveness and retraining efficacy of MCC and state-of-the-art white-box coverage metrics. They found that the misprediction-revealing effectiveness of MCC is similar to that of the selected white-box coverage metrics. Further, MCC failed to outperform the white-box coverage metrics in retraining effectiveness. Given the reported performance of MCC compared to white-box coverage metrics, we did not consider this recent work in our empirical analysis.

Despite active research on DNN coverage, several recent articles have questioned the usefulness of coverage criteria to guide the testing of DNN models [16], [17], [18]. For example, Li *et al.* [16] studied a number of structural coverage criteria and discussed their limitations in fault detection capabilities in DNN models. Their experiments found no strong correlation between coverage and the number of misclassified inputs in a natural test set. Furthermore, Dong *et al.* [17] found that retraining DNN models with new input sets that improve coverage does not increase the robustness of the model to adversarial attacks.

Our work on diversity metrics is orthogonal to existing research regarding DNN test coverage. Most of the state-of-the-art coverage metrics require full access to the internals of DNN state or training data, both of which are often not available to testers in practical contexts. Thus, in our approach we focused on black-box diversity metrics, and aimed to provide guidance to assess test suites or select test cases for DNNs.

State-of-the-art coverage criteria have been largely validated with artificial inputs that have been generated based on adversarial methods [12], [13], [14], [10], [11]. However, their relationship to (often unrealistic) adversarial inputs does not imply they relate to the fault detection capability of natural test input sets. Li *et al.* [16] argue that adversarial inputs are distributed pervasively over the divided space defined by existing coverage criteria. On the other hand, misclassified natural inputs are distributed sparsely, making their detection difficult when using such coverage criteria [16]. Existing studies [16], [18] have failed to find a significant correlation between coverage and the number of misclassified inputs in a test set. Consequently, coverage criteria may be ineffective at guiding DNN testing to increase the fault-detection capability of natural input sets. Furthermore, existing studies [12], [13], [14], [10], [11] have used the number of mispredicted inputs to study the effectiveness of coverage criteria to support DNN testing. However, as previously discussed, simply comparing mispredictions is highly misleading because many test inputs may (and usually do) fail due to the same causes in the DNN model. To address this problem, we approximated faults (i.e. common misprediction causes) by relying on a clustering strategy and by studying the correlation between test criteria (i.e. coverage and diversity) and faults instead of misprediction rates.

6.2 Black-box DNN Testing

In this section we describe black-box testing approaches for DNN models because the focus of this paper is on black-box metrics and diversity metrics, and on their association with detected faults in DNN models. Feng *et al.* [83] proposed DeepGini, a black-box test selection approach that prioritizes test inputs along with higher uncertainty scores. Intuitively, a test input is likely to be misclassified by a DNN if the model is uncertain about the classification and outputs similar probabilities for each class [83]. They found that DeepGini outperforms random and coverage-based test selection approaches to reveal misclassifications. However, this approach is only applicable to classification problems and cannot be used for regression tasks. A recent study by Gao *et al.* [84], developed concurrently to our work, proposed Adaptive Test Selection (ATS), a method based on uncertainty scores and distribution of the output probability vectors of test inputs in DNN models. The selection is guided by a fault pattern coverage score that is computed using the output probability vectors of the DNN under test. They introduce a mapping from the output domain of the DNN model to a set of intervals in local domains to describe the fault pattern of a given test input or test set. They select test inputs that both cover different fault patterns [84] and have higher uncertainty scores. Similar to DeepGini, this approach can only be used for classification problems. Empirical results show that ATS outperforms coverage and uncertainty-based test selection methods (including DeepGini [83]) in misprediction-revealing capability. They also studied the effectiveness of the proposed approach in finding diverse mispredicted inputs. Consequently, they introduced the concept of fault types by looking at pairs of actual test input labels and labels predicted by the DNN model under test. Inputs that have different pairs of actual

and predicted labels are assumed to correspond to different types of faults. However, as opposed to our work, this proposed method for counting distinct faults was not validated. We relied on a more fine-grained fault estimation approach that is based on clustering mispredicted inputs and accounts for their labels (actual and mispredicted) as well as their features.

Li *et al.* [22] proposed two black-box metrics called Cross Entropy-based Sampling (CES) and Confidence-based Stratified Sampling (CSS) for DNN test set minimization. These metrics are used to guide the selection of a small set of test inputs that can accurately estimate the entire testing dataset. The authors show that their approach outperforms random sampling and requires only about half the labeled test inputs to achieve the same level of accuracy as the whole testing set. The authors also report that CSS does not perform well on poorly trained DNNs because the confidence values it produces cannot be trusted [22].

These black-box testing approaches and their underlying metrics are conceptually different from our black-box diversity metrics. They are model-dependent because they rely on DNN outputs (output diversity) and uncertainty assessments. From a practical standpoint, this implies that all inputs must first be executed to be selected based on their diversity, which is a strong practical impediment in many application contexts, for example when working with large models and large databases of unlabeled inputs. Further, as mentioned above, these diversity metrics cannot be trusted when the model is not accurate. In contrast, our diversity metrics are model-independent and based on an analysis of the diversity of input features, thus requiring no model execution. Finally, these black-box testing approaches focus on specific testing scenarios, while our study is generic and focuses on investigating fundamental and pervasive assumptions related to the relationship between testing criteria and faults in DNNs.

6.3   Diversity in Testing

In this section, we describe existing work that relied on diversity to test DNNs and regular software.

Diversity in DNN Testing. A recent study by Langford and Cheng [85] proposed Enki, a DNN input-generation approach based on evolutionary search [86]. Their objective is to diversify image transformation types and to generate new inputs from existing ones to test and retrain DNN models. They started by evolving an archive of image transformation types that have a diverse impact on the DNN model. Given a subset of synthetic inputs generated with a certain image transformation type, the diversity of the impact was evaluated against three elements: (1) the F1-score of the DNN model when applied on the subset; (2) the neuron coverage score [12]; and (3) the neuron's activation pattern [85]. After building the final Enki archive containing the most diverse image transformation types, they (1) tested the DNN models using synthetic inputs generated with the identified image transformation types, and (2) studied the accuracy of the DNNs by retraining them with such synthetic training data. They also compared their results with random input generation and DeepTest [87]. They concluded that Enki outperformed these two input generation approaches, and

reported that testing DNNs with their generated data led to the lowest DNN model accuracy. They also reported that retraining DNNs with their generated data increased the accuracy of DNN models.

What differentiates our work from Enki is that Enki provides a search-based approach to diversify image transformation types. Its goal is to minimize the model accuracy and then use it to guide the generation of synthetic inputs to test and retrain DNNs. In contrast, our approach investigated ways to measure diversity in natural test input sets and compared the best diversity metric with state-of-the-art coverage criteria to guide DNN testing to maximize fault detection. Such diversity metrics can then be used for multiple purposes such as test suite assessment and guidance for selection, minimization and generation. Our focus on faults, as opposed to accuracy, aimed to find test inputs whose mispredictions resulted from distinct root causes. For practical reasons, as already discussed and as opposed to Enki, we intentionally devised an approach that is black-box and did not rely on internal information about the model or its training set.

Diversity in Software Testing. Input and output diversity has been investigated to support different aspects related to traditional software testing. Since executing similar test cases tends to exercise similar parts of the source code, this is likely to lead to revealing the same faults in the system under test. Therefore, relying on diverse test cases should increase the exploration of the fault space and thus increase fault detection rates [88], [20], [89].

Feldt *et al.* [21] proposed Test Set Diameter (TDSm), a diversity-based test case selection strategy. The approach uses the NCD metric to measure the diversity of test inputs. They applied their approach on four systems and concluded that diverse test input sets increase code coverage. Finally, they show that test sets with larger NCD scores exhibit better fault-detection capabilities.

Hemmati *et al.* [90] conducted an empirical study on similarity-based test selection techniques for test cases generated from state machine models. They studied and compared over 320 variants that relied on different similarity metrics and selection algorithms. Based on their experiments, they found that the best test-selection technique used the Gower-Legendre similarity function [91] and applied a (1+1) Evolutionary Algorithm [92] to select tests with minimum pairwise similarity and thus maximized the diversity of the selected test cases. They further showed that such similarity-based test selection configuration outperformed random selection and coverage-based techniques in fault detection rates and computational cost.

Biagiola *et al.* [19] introduced a web test generation algorithm that produces and selects candidate test cases that are executed in the browser based on their diversity. They showed that their test generation technique achieved higher code coverage and fault detection rates when compared to state of-the-art, search-based web test generators [76], [93].

Our objectives in this paper are similar to these studies, but in the context of DNN testing. As several studies have shown the effectiveness of diversity metrics in guiding the testing of software systems, we investigated its usefulness in testing DNN models. We compared the performance of existing diversity metrics with state-of-the-art DNN cover-

age criteria in terms of their fault detection capabilities and computational cost.

7 CONCLUSION

In this paper, we studied the effectiveness of input diversity metrics in guiding the testing of DNN models. We focused on DNN models using images as inputs, as they are common in many systems. Our motivation is to provide a black-box mechanism that does not rely on DNN internal information or training data to assess test sets. This requirement aims to make our approach more applicable in the many practical contexts where such information is not (easily) accessible. We also do not rely on output diversity because this approach would require executing the model with all inputs and would be affected by poor model accuracy. Further, we compare the results achieved by white-box coverage criteria defined for DNNs with those achieved by black-box input diversity.

To this end, we selected and adapted three input diversity metrics and, by means of a controlled experiment, evaluated their capability to measure actual input diversity. We selected the best metrics and analyzed their association with fault detection in DNNs using four datasets and five DNN models. Because simply comparing mispredictions is highly misleading, as many test inputs fail for the same reasons, we relied on a clustering-based approach to group similar mispredicted inputs and thus estimated faults based on the number of such clusters. We further selected the best state-of-the-art coverage criteria based on published results and our ability to reproduce such results. We studied the associations of the selected coverage criteria to both diversity and fault detection.

Based on our experiments, we found that the best diversity metric is geometric diversity and that, though there is still room for improvement (e.g. fault estimation in DNNs, investigating other diversity metrics and feature extraction models), it is a far better surrogate metric than the investigated coverage criteria in terms of their relationship to fault detection. This metric outperforms these coverage criteria in correlation to detected faults and computational time. We therefore recommend investigating the use of geometric diversity as a black-box metric to guide the testing of DNN models using images as inputs. We aim to extend our work by studying the application of input diversity in supporting different testing applications such as test set selection, minimization and generation. We also intend to investigate alternatives for DNN fault estimation.

ACKNOWLEDGEMENTS

REFERENCES

[1] X. Yang, F. Li, and H. Liu, "A survey of dnn methods for blind image quality assessment," *IEEE Access*, vol. 7, pp. 123 788–123 806, 2019.

[2] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 4034–4038.

[3] P. K. Mallick, S. H. Ryu, S. K. Satapathy, S. Mishra, G. N. Nguyen, and P. Tiwari, "Brain mri image classification for cancer detection using deep wavelet autoencoder-based deep neural network," *IEEE Access*, vol. 7, pp. 46 278–46 287, 2019.

[4] V. Rajinikanth, A. N. Joseph Raj, K. P. Thanaraj, and G. R. Naik, "A customized vgg19 network with concatenation of deep and handcrafted features for brain tumor detection," *Applied Sciences*, vol. 10, no. 10, p. 3429, 2020.

[5] T. G. Debelee, S. R. Kebede, F. Schwenker, and Z. M. Shewarega, "Deep learning in selected cancers' image analysis—a survey," *Journal of Imaging*, vol. 6, no. 11, p. 121, 2020.

[6] J. Pan, C. Liu, Z. Wang, Y. Hu, and H. Jiang, "Investigation of deep neural networks (dnn) for large vocabulary continuous speech recognition: Why dnn surpasses gmms in acoustic modeling," in *2012 8th International Symposium on Chinese Spoken Language Processing*. IEEE, 2012, pp. 301–305.

[7] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[8] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 359–371.

[9] X. Cai and M. R. Lyu, "The effect of code coverage on fault detection under different testing profiles," in *Proceedings of the 1st International Workshop on Advances in Model-based Testing*, 2005, pp. 1–7.

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

[11] S. Gerasimou, H. F. Eniser, A. Sen, and A. Cakan, "Importance-driven deep learning system testing," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 702–713.

[12] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[13] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepgauge: Multi-granularity testing criteria for deep learning systems," *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 120–131, 2018.

[14] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Structural test coverage criteria for deep neural networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–23, 2019.

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

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

[17] Y. Dong, P. Zhang, J. Wang, S. Liu, J. Sun, J. Hao, X. Wang, L. Wang, J. Dong, and T. Dai, "An empirical study on correlation between coverage and robustness for deep neural networks," in *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 2020, pp. 73–82.

[18] J. Chen, M. Yan, Z. Wang, Y. Kang, and Z. Wu, "Deep neural network test coverage: How far are we?" *arXiv preprint arXiv:2010.04946*, 2020.

[19] M. Biagiola, A. Stocco, F. Ricca, and P. Tonella, "Diversity-based web test generation," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 142–153.

[20] H. Hemmati, Z. Fang, and M. V. Mantyla, "Prioritizing manual test cases in traditional and rapid release environments," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2015, pp. 1–10.

[21] R. Feldt, S. Poulding, D. Clark, and S. Yoo, "Test set diameter: Quantifying the diversity of sets of test cases," in *2016 IEEE In-*

ternational Conference on Software Testing, Verification and Validation (ICST). IEEE, 2016, pp. 223–233.

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

[23] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. Tse, "Adaptive random testing: The art of test case diversity," Journal of Systems and Software, vol. 83, no. 1, pp. 60–66, 2010.

[24] T. Y. Chen, R. Merkel, P. Wong, and G. Eddy, "Adaptive random testing through dynamic partitioning," in Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings. IEEE, 2004, pp. 79–86.

[25] H. Fahmy, F. Pastore, M. Bagherzadeh, and L. Briand, "Supporting deep neural network safety analysis and retraining through heatmap-based unsupervised learning," IEEE Transactions on Reliability, 2021.

[26] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," arXiv preprint arXiv:1207.6083, 2012.

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

[28] W. Mousser and S. Ouadfel, "Deep feature extraction for papsmear image classification: A comparative study," in Proceedings of the 2019 5th International Conference on Computer and Technology Applications, 2019, pp. 6–10.

[29] T. Kaur and T. K. Gandhi, "Automated brain image classification based on vgg-16 and transfer learning," in 2019 International Conference on Information Technology (ICIT). IEEE, 2019, pp. 94–98.

[30] Z. Gong, P. Zhong, and W. Hu, "Diversity in machine learning," IEEE Access, vol. 7, pp. 64 323–64 350, 2019.

[31] A. R. Cohen and P. M. Vitányi, "Normalized compression distance of multisets with applications," IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 8, pp. 1602–1614, 2014.

[32] J. R. Hershey and P. A. Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07, vol. 4. IEEE, 2007, pp. IV–317.

[33] B. Chen, X. He, B. Pan, X. Zou, and N. You, "Comparison of beta diversity measures in clustering the high-dimensional microbial data," PloS one, vol. 16, no. 2, p. e0246893, 2021.

[34] H. Lin and J. A. Bilmes, "Learning mixtures of submodular shells with application to document summarization," arXiv preprint arXiv:1210.4871, 2012.

[35] B. Kang, "Fast determinantal point process sampling with application to clustering," Advances in Neural Information Processing Systems, vol. 26, 2013.

[36] H. Xu and Z. Ou, "Scalable discovery of audio fingerprint motifs in broadcast streams with determinantal point process based motif clustering," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 24, no. 5, pp. 978–989, 2016.

[37] H. Z. Nafchi, A. Shahkolaei, R. Hedjam, and M. Cheriet, "Mean deviation similarity index: Efficient and reliable full-reference image quality evaluator," Ieee Access, vol. 4, pp. 5579–5590, 2016.

[38] R. S. Borbely, "On normalized compression distance and large malware," Journal of Computer Virology and Hacking Techniques, vol. 12, no. 4, pp. 235–242, 2016.

[39] R. Cilibrasi and P. M. Vitányi, "Clustering by compression," IEEE Transactions on Information theory, vol. 51, no. 4, pp. 1523–1545, 2005.

[40] M. Elfeki, C. Couprie, M. Riviere, and M. Elhoseiny, "Gdpp: Learning diverse generations using determinantal point processes," in International Conference on Machine Learning. PMLR, 2019, pp. 1774–1783.

[41] B. Gong, W.-L. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," Advances in neural information processing systems, vol. 27, pp. 2069–2077, 2014.

[42] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," Proceedings of the National Academy of Sciences, vol. 107, no. 10, pp. 4511–4515, 2010.

[43] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies." Journal of Machine Learning Research, vol. 9, no. 2, 2008.

[44] E. Celis, V. Keswani, D. Straszak, A. Deshpande, T. Kathuria, and N. Vishnoi, "Fair and diverse dpp-based data summarization," in International Conference on Machine Learning. PMLR, 2018, pp. 716–725.

[45] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai, "Better mixing via deep representations," in International conference on machine learning. PMLR, 2013, pp. 552–560.

[46] A. N. Kolmogorov, "Three approaches to the quantitative definition ofinformation'," Problems of information transmission, vol. 1, no. 1, pp. 1–7, 1965.

[47] C. H. Bennett, P. Gács, M. Li, P. M. Vitányi, and W. H. Zurek, "Information distance," IEEE Transactions on information theory, vol. 44, no. 4, pp. 1407–1423, 1998.

[48] M. Li, X. Chen, X. Li, B. Ma, and P. M. Vitányi, "The similarity metric," IEEE transactions on Information Theory, vol. 50, no. 12, pp. 3250–3264, 2004.

[49] R. Feldt, R. Torkar, T. Gorschek, and W. Afzal, "Searching for cognitively diverse tests: Towards universal test diversity metrics," in 2008 IEEE International Conference on Software Testing Verification and Validation Workshop. IEEE, 2008, pp. 178–186.

[50] D. Coltuc, M. Datcu, and D. Coltuc, "On the use of normalized compression distances for image similarity detection," Entropy, vol. 20, no. 2, p. 99, 2018.

[51] A. Kocsor, A. Kertész-Farkas, L. Kaján, and S. Pongor, "Application of compression-based distance measures to protein sequence classification: a methodological study," Bioinformatics, vol. 22, no. 4, pp. 407–412, 2006.

[52] C. Henard, M. Papadakis, M. Harman, Y. Jia, and Y. Le Traon, "Comparing white-box and black-box test prioritization," in 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). IEEE, 2016, pp. 523–534.

[53] P. M. Bueno, W. E. Wong, and M. Jino, "Improving random test sets using the diversity oriented test data generation," in Proceedings of the 2nd international workshop on Random testing: co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), 2007, pp. 10–17.

[54] D. Leon and A. Podgurski, "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases," in 14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003. IEEE, 2003, pp. 442–453.

[55] F. Harel-Canada, L. Wang, M. A. Gulzar, Q. Gu, and M. Kim, "Is neuron coverage a meaningful measure for testing deep neural networks?" in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020, pp. 851–862.

[56] S. Yan, G. Tao, X. Liu, J. Zhai, S. Ma, L. Xu, and X. Zhang, "Correlations between deep neural network model coverage criteria and model quality," in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020, pp. 775–787.

[57] K. Alex, N. Vinod, and H. Geoffrey. The cifar-10 dataset. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html

[58] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, 2012.

[59] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[60] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, "Deephyperion: exploring the feature space of deep learning-based systems through illumination search," in Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2021, pp. 79–90.

[61] M. O. Attaoui, H. Fahmy, F. Pastore, and L. Briand, "Black-box safety analysis and retraining of dnns based on feature extraction and clustering," ACM Trans. Softw. Eng. Methodol., jul 2022, just Accepted. [Online]. Available: https://doi.org/10.1145/3550271

[62] M. Joswiak, Y. Peng, I. Castillo, and L. H. Chiang, "Dimensionality reduction for visualizing industrial chemical process data," Control Engineering Practice, vol. 93, p. 104189, 2019.

[63] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," arXiv preprint arXiv:1802.03426, 2018.

[64] A. Diaz-Papkovich, L. Anderson-Trocmé, and S. Gravel, "A review

of umap in population genetics," *Journal of Human Genetics*, vol. 66, no. 1, pp. 85–91, 2021.

[65] Y. Hozumi, R. Wang, C. Yin, and G.-W. Wei, "Umap-assisted k-means clustering of large-scale sars-cov-2 mutation datasets," *Computers in biology and medicine*, vol. 131, p. 104264, 2021.

[66] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[67] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[68] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[69] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[70] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 2014, pp. 839–847.

[71] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "Density-connected sets and their application for trend detection in spatial databases." in *KDD*, vol. 97, 1997, pp. 10–15.

[72] 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 Transactions on Software Engineering and Methodology*, 2022.

[73] D. G. Bonett and T. A. Wright, "Sample size requirements for estimating pearson, kendall and spearman correlations," *Psychometrika*, vol. 65, no. 1, pp. 23–28, 2000.

[74] R. F. Woolson, "Wilcoxon signed-rank test," *Wiley encyclopedia of clinical trials*, pp. 1–3, 2007.

[75] Z. Aghababaeyan, M. Abdellatif, L. Briand, R, S, and M. Bagherzadeh. Dnn testing replication package. [Online]. Available: https://github.com/zohreh-aaa/DNN-Testing

[76] M. Biagiola, F. Ricca, and P. Tonella, "Search based path and input data generation for web application testing," in *International Symposium on Search Based Software Engineering*. Springer, 2017, pp. 18–32.

[77] D. Zou, J. Liang, Y. Xiong, M. D. Ernst, and L. Zhang, "An empirical study of fault localization families and their combinations," *IEEE Transactions on Software Engineering*, 2019.

[78] S. Pearson, J. Campos, R. Just, G. Fraser, R. Abreu, M. D. Ernst, D. Pang, and B. Keller, "Evaluating and improving fault localization," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 609–620.

[79] W. E. Wong, V. Debroy, R. Gao, and Y. Li, "The dstar method for effective software fault localization," *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 290–308, 2013.

[80] M. P. Wand and M. C. Jones, *Kernel smoothing*. CRC press, 1994.

[81] T. Byun, S. Rayadurgam, and M. P. Heimdahl, "Black-box testing of deep neural networks," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2021, pp. 309–320.

[82] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[83] 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 *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 177–188.

[84] X. Gao, Y. Feng, Y. Yin, Z. Liu, Z. Chen, and B. Xu, "Adaptive test selection for deep neural networks," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*. IEEE, 2022, pp. 73–85.

[85] M. A. Langford and B. H. Cheng, "Enki: A diversity-driven approach to test and train robust learning-enabled systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 15, no. 2, pp. 1–32, 2021.

[86] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2003.

[87] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: Association

for Computing Machinery, 2018, p. 303–314. [Online]. Available: https://doi.org/10.1145/3180155.3180220

[88] E. G. Cartaxo, P. D. Machado, and F. G. O. Neto, "On the use of a similarity function for test case selection in the context of model-based testing," *Software Testing, Verification and Reliability*, vol. 21, no. 2, pp. 75–100, 2011.

[89] F. G. de Oliveira Neto, A. Ahmad, O. Leifler, K. Sandahl, and E. Enoiu, "Improving continuous integration with similarity-based test case selection," in *Proceedings of the 13th International Workshop on Automation of Software Test*, 2018, pp. 39–45.

[90] H. Hemmati, A. Arcuri, and L. Briand, "Achieving scalable model-based testing through test case diversity," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 22, no. 1, pp. 1–42, 2013.

[91] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[92] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+ 1) evolutionary algorithm," *Theoretical Computer Science*, vol. 276, no. 1-2, pp. 51–81, 2002.

[93] A. Mesbah, A. Van Deursen, and D. Roest, "Invariant-based automatic testing of modern web applications," *IEEE Transactions on Software Engineering*, vol. 38, no. 1, pp. 35–53, 2011.

APPENDIX A
CORRELATION RESULTS BETWEEN GEOMETRIC DIVERSITY AND COVERAGE METRICS

Note that the grey boxes in all the following tables refer to statistically significant correlations (p-value <= 0.05)

Dataset	Model	Metric	Test Set Size	Spearman	P-value
Cifar-10	12-layer ConvNet	GD & LSC	100	13%	0.45
			200	-2%	0.92
			300	1%	0.97
			400	-10%	0.58
			1000	-12%	0.53
		GD & DSC	100	-20%	0.27
			200	-11%	0.55
			300	11%	0.49
			400	-17%	0.36
			1000	-7%	0.73
		GD & NC	100	-30%	0.09
			200	12%	0.49
			300	10%	0.56
			400	2%	0.90
			1000	2%	0.91
		GD & KMNC	100	44%	0.01
			200	15%	0.40
			300	15%	0.37
			400	9%	0.63
			1000	12%	0.54
		GD & NBC	100	31%	0.08
			200	6%	0.75
			300	17%	0.31
			400	-10%	0.62
			1000	-28%	0.14
		GD & TKNC	100	-19%	0.30
			200	17%	0.33
			300	-4%	0.83
			400	7%	0.69
			1000	7%	0.71
		GD & SNAC	100	34%	0.05
			200	6%	0.75
			300	13%	0.44
			400	-13%	0.47
			1000	-28%	0.14

Table 9: Correlation results between GD and coverage metrics using 12-layer ConvNet and Cifar-10.

Dataset	Model	Metric	Test Set Size	Spearman	P-value
MNIST	LeNet-1	GD & LSC	100	-6%	0.78
			200	-12%	0.57
			300	-6%	0.78
			400	15%	0.30
			1000	-4%	0.80
		GD & DSC	100	-27 %	0.17
			200	-10%	0.64
			300	-27%	0.17
			400	25%	0.09
			1000	-14%	0.41
		GD & NC	100	18%	0.38
			200	-15%	0.47
			300	3%	0.89
			400	-3%	0.85
			1000	-30%	0.08
		GD & KMNC	100	-15%	0.47
			200	-16%	0.45
			300	8%	0.68
			400	28%	0.06
			1000	6%	0.71
		GD & NBC	100	6%	0.78
			200	6%	0.79
			300	-9%	0.65
			400	15%	0.30
			1000	-15%	0.39
		GD & TKNC	100	7%	0.73
			200	11%	0.61
			300	29%	0.14
			400	21%	0.16
			1000	21%	0.23
		GD & SNAC	100	-6%	0.78
			200	7%	0.75
			300	-14%	0.47
			400	17%	0.26
			1000	-13%	0.45

Table 10: Correlation results between GD and coverage metrics using LeNet-1 and MNIST.

Dataset	Model	Metric	Test Set Size	Spearman	P-value
MNIST	LeNet-5	GD & LSC	100	3%	0.88
			200	25%	0.17
			300	26%	0.26
			400	-22%	0.23
			1000	-7%	0.74
		GD & DSC	100	13%	0.52
			200	-10%	0.59
			300	-22%	0.34
			400	5%	0.81
			1000	-1%	0.95
		GD & NC	100	-22%	0.28
			200	-11%	0.57
			300	-4%	0.86
			400	-34%	0.06
			1000	39%	0.05
		GD & KMNC	100	-1%	0.94
			200	20%	0.29
			300	4%	0.88
			400	5%	0.79
			1000	-31%	0.11
		GD & NBC	100	-27%	0.17
			200	-1%	0.95
			300	-4%	0.86
			400	-14%	0.46
			1000	48%	0.01
		GD & TKNC	100	4 %	0.84
			200	24%	0.18
			300	-17%	0.46
			400	-2%	0.90
			1000	3%	0.90
		GD & SNAC	100	-18%	0.38
			200	-1%	0.95
			300	-5%	0.81
			400	-21%	0.25
			1000	41%	0.03

Table 11: Correlation results between GD and coverage metrics using LeNet-5 and MNIST.

Dataset	Model	Metric	Test Set Size	Spearman	P-value
Fashion-MNIST	LeNet-4	GD & LSC	100	0.2%	0.99
			200	28%	0.17
			300	19%	0.29
			400	21%	0.20
			1000	2%	0.94
		GD & DSC	100	17%	0.38
			200	-7%	0.73
			300	-35%	0.05
			400	-2%	0.90
			1000	2%	0.93
		GD & NC	100	-37%	0.04
			200	0.3%	0.99
			300	-5%	0.77
			400	23%	0.16
			1000	-14%	0.49
		GD & KMNC	100	23 %	0.22
			200	30%	0.14
			300	33%	0.06
			400	64%	0.0001
			1000	6%	0.76
		GD & NBC	100	-25 %	0.19
			200	-31%	0.13
			300	12%	0.51
			400	-22%	0.19
			1000	-21%	0.31
		GD & TKNC	100	-1 %	0.95
			200	41%	0.04
			300	-3%	0.85
			400	24%	0.15
			1000	11%	0.61
		GD & SNAC	100	-25%	0.19
			200	-31%	0.13
			300	12%	0.52
			400	-21%	0.20
			1000	-19%	0.35

Table 12: Correlation results between GD and coverage metrics using LeNet-4 and Fashion-MNIST.

Dataset	Model	Metric	Test Set Size	Spearman	P-value
Cifar-10	ResNet-20	GD & LSC	100	19 %	0.31
			200	14%	0.55
			300	39%	0.11
			400	**46%**	0.02
			1000	-26%	0.18
		GD & DSC	100	4 %	0.85
			200	-8%	0.74
			300	5%	0.84
			400	-10%	0.65
			1000	-3%	0.86
		GD & NC	100	5 %	0.81
			200	26%	0.25
			300	-2%	0.94
			400	12%	0.58
			1000	**-38%**	0.04
		GD & KMNC	100	19 %	0.3
			200	39%	0.08
			300	17%	0.51
			400	**41%**	0.05
			1000	28%	0.15
		GD & NBC	100	24 %	0.20
			200	12%	0.60
			300	-4%	0.87
			400	-9%	0.67
			1000	-18%	0.35
		GD & TKNC	100	2 %	0.9
			200	18%	0.43
			300	10%	0.70
			400	**58%**	0.003
			1000	-7%	0.73
		GD & SNAC	100	17 %	0.35
			200	11 %	0.62
			300	4 %	0.89
			400	-3 %	0.90
			1000	-14 %	0.47

Table 13: Correlation results between GD and coverage metrics using ResNet-20 and Cifar-10.

Dataset	Model	Metric	Test Set Size	Spearman	P-value
SVHN	LeNet-5	GD & LSC	100	15%	0.37
			200	27%	0.09
			300	2%	0.91
			400	15%	0.46
			1000	34%	0.06
		GD & DSC	100	-16 %	0.33
			200	9%	0.58
			300	19%	0.28
			400	14%	0.47
			1000	-14%	0.45
		GD & NC	100	-8%	0.61
			200	4%	0.79
			300	30%	0.08
			400	-28%	0.14
			1000	-12%	0.52
		GD & KMNC	100	**44%**	0.001
			200	**66%**	0.002
			300	23%	0.19
			400	-2%	0.9
			1000	**46%**	0.01
		GD & NBC	100	9%	0.58
			200	5%	0.77
			300	-19%	0.27
			400	26%	0.18
			1000	28%	0.12
		GD & TKNC	100	-3%	0.83
			200	1%	0.95
			300	-22%	0.20
			400	-15%	0.43
			1000	18%	0.34
		GD & SNAC	100	9%	0.58
			200	5%	0.77
			300	-19%	0.27
			400	26%	0.18
			1000	28%	0.12

Table 14: Correlation results between GD and coverage metrics using LeNet-5 and SVHN.

APPENDIX B
AUTHORS' BIOGRAPHIES

Zohreh Aghababaeyan is a PhD student at the School of EECS at the University of Ottawa and a member of the Nanda Lab. She gained practical experience during her internship at the research and development lab in General Motors, USA. She received several academic awards, including a PhD admission scholarship, an international doctoral scholarship from University of Ottawa, and an honourable award of admission to the master's program in computer science at Amirkabir University of Technology in Iran. She was also ranked the third-best student among all computer science students at Amirkabir University in 2017. Her research interests include testing and verification of machine learning-based systems and empirical software engineering.

Manel Abdellatif received her PhD in computer science from Polytechnique Montréal, Canada (2021). She is a faculty member at École de Technologie Supérieure, Canada. She was a postdoctoral fellow at the School of EECS, University of Ottawa (2022). She received her Master's degree in Information Technology from École de Technologie Supérieure (2016) and earned her Bachelor's degree from École Nationale d'Ingénieurs de Tunis (2013). She served as a program committee member and a reviewer in several journals and conferences. Her research interests include testing machine learning-based systems, service computing, and empirical software engineering.
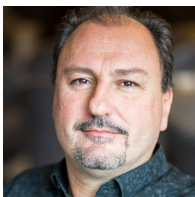
Lionel Briand is professor of software engineering and has shared appointments between (1) School of Electrical Engineering and Computer Science, University of Ottawa, Canada and (2) The SnT centre for Security, Reliability, and Trust, University of Luxembourg. He is the head of the SVV department at the SnT Centre and a Canada Research Chair in Intelligent Software Dependability and Compliance (Tier 1). He has conducted applied research in collaboration with industry for more than 25 years, including projects in the automotive, aerospace, manufacturing, financial, and energy domains. He is a fellow of the IEEE and ACM. He was also granted the IEEE Computer Society Harlan Mills award (2012), the IEEE Reliability Society Engineer-of-the-year award (2013), and the ACM SIGSOFT Outstanding Research Award (2022) for his work on software testing and verification. More details can be found on: http://www.lbriand.info.

Ramesh S, Senior Technical Fellow, has been with General Motors Research and Development (R&D) for more than 15 years conducting and leading advanced research projects in the areas of model-based development of embedded systems and software, rigorous verification and validation and more recently AI/ML based systems. Prior to joining GM R&D, he was a full professor in the department of Computer Science and Engineering at Indian Institute of Technology Bombay, India where he co-founded a Centre for Formal Design and Verification of Software. He has published more than 125 research papers in International Journals and Conferences and author many patents in the areas of modeling, analysis and verification of embedded systems and software. He has been on the program committees of several international research conferences and on the editorial boards of journals. He is leading an USCAR committee and serving as an expert in ISO and SAE committees for developing guidelines for AI/ML based systems.

Mojtaba Bagherzadeh is a highly experienced Software Engineer with a proven track record of success in both industry and academia. He currently works as a software engineer at Cisco Systems and has previously worked as a software developer at IBM and a startup company. He has contributed to this research during his tenure as a postdoctoral researcher at the University of Ottawa. He obtained his PhD in Computer Science from Queen's University, Canada, in 2019. His research interests include testing and debugging of machine learning-based systems, model-driven engineering, software testing, and empirical software engineering.