Towards Reliable AI: Adequacy Metrics for Ensuring the Quality of System-level Testing of Autonomous Vehicles

Neelofar Neelofar
Monash University
Australia
neelofar.neelofar@monash.edu

Aldeida Aleti
Monash University
Australia
aldeida.aleti@monash.edu

ABSTRACT

AI-powered systems have gained widespread popularity in various domains, including Autonomous Vehicles (AVs). However, ensuring their reliability and safety is challenging due to their complex nature. Conventional test adequacy metrics, designed to evaluate the effectiveness of traditional software testing, are often insufficient or impractical for these systems. White-box metrics, which are specifically designed for these systems, leverage neuron coverage information. These coverage metrics necessitate access to the underlying AI model and training data, which may not always be available. Furthermore, the existing adequacy metrics exhibit weak correlations with the ability to detect faults in the generated test suite, creating a gap that we aim to bridge in this study.

In this paper, we introduce a set of black-box test adequacy metrics called "Test suite Instance Space Adequacy" (TISA) metrics, which can be used to gauge the effectiveness of a test suite. The TISA metrics offer a way to assess both the diversity and coverage of the test suite and the range of bugs detected during testing. Additionally, we introduce a framework that permits testers to visualise the diversity and coverage of the test suite in a two-dimensional space, facilitating the identification of areas that require improvement.

We evaluate the efficacy of the TISA metrics by examining their correlation with the number of bugs detected in system-level simulation testing of AVs. A strong correlation, coupled with the short computation time, indicates their effectiveness and efficiency in estimating the adequacy of testing AVs.

ACM Reference Format:
Neelofar Neelofar and Aldeida Aleti. 2024. Towards Reliable AI: Adequacy Metrics for Ensuring the Quality of System-level Testing of Autonomous Vehicles. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3597503.3623314

1 INTRODUCTION

Thanks to the availability of big data and computational power, AI-based systems, like AVs, have shown a remarkable increase in their capabilities. However, performing quality assurance of these systems requires a paradigm shift, as not all the logic is explicitly encoded in the source code. These systems are trained on enormous quantities of data and act based on the logic defined in terms of patterns observed in the data [54]. Therefore, traditional test adequacy measures can't be effectively applied to assess the quality and reliability of these systems, and novel contributions from researchers and practitioners are urgently required.

The challenge of testing AI-based systems is well accepted by the Software Engineering community, which is actively working on finding effective test adequacy metrics for such systems [55, 67, 77, 84, 86]. Since traditional white box code coverage metrics can not be effectively used to measure how well the program logic – that is partially determined by the underlying training data – has been adequately exercised, specialised white box adequacy metrics aiming at maximising neuron coverage [35, 40, 64, 81, 82] or surprised coverage [40] are introduced for model-level testing. A limitation of the coverage-based adequacy measures is that it requires full access to the underlying DNN and training data, both of which are not often available to testers when performing system-level testing. Furthermore, the performance of most coverage measures is assessed using adversarial inputs, thus focussing on the robustness of the model instead of correctness. Irrespective of the claimed sensitivity of these measures to adversarial inputs, studies have failed to find a significant correlation between these coverage measures and their fault detection capability [5, 18, 48].

Another widely used adequacy measure for traditional and AI-based systems is test suite diversity computed on test inputs or outputs [5, 13, 29, 50]. The diversity metrics are designed based on the intuition that similar test cases exercise similar parts of the source code or training examples, thus revealing the same faults. On the other hand, a system that is tested under a wide range of conditions is more likely to perform reliably in the real world. Therefore, it is believed that diversifying test scenarios improves the exploration of the fault space, leading to increased fault detection [5, 17, 87]. However, the current research on testing AI-based systems has identified a significant gap in the availability of good diversity metrics that show a strong correlation with fault detection [5].

Among many applications of AI in Software Engineering, AVs are one of the most sophisticated and highly complex systems that rely on a combination of sensors to collect data, AI-based systems to process the data and take decisions, and control systems to operate the AV safely in diverse real-world conditions. The diversity in the testing of AVs is crucial to maximising the probability of testing the system under boundary conditions or rare emergency situations that may occur in the real world. Furthermore, AVs interact with other road users, including pedestrians, cyclists and other vehicles driven by humans. Testing under diverse conditions is important

ICSE '24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0217-4/24/04…$15.00
https://doi.org/10.1145/3597503.3623314

to find how the system handles potentially unpredictable and irrational actions related to human behaviour. In addition, evaluating the coverage of testing is crucial to determine the extent to which the input/output behaviour of the AV system has been tested.

Given the significance of both diversity and coverage for black-box testing of AVs, we propose a novel set of metrics, called *Test suite Instance Space Adequacy (TISA)* metrics, that provides an objective measure of the quality of testing in terms of both diversity and coverage. TISA metrics are computed by identifying the features of the test scenarios having maximum impact on test outcome (safe vs unsafe), and then projecting these test scenarios – defined in terms of the impactful features – on $2D$ space, using a framework called Instance Space Analysis (ISA) [62]. The representation of the test scenarios in a 2D space facilitates a clear visualisation of the test suite, resulting in an easy evaluation of the diversity and coverage of the whole test suite across each feature, along with the diversity of the detected bugs. As a result, the proposed metrics serve as a valuable tool for testers to identify areas where the test suite may lack diversity or areas that require additional testing. This ultimately leads to an enhanced quality of testing by highlighting specific areas that need improvement or further exploration.

To investigate the effectiveness of the proposed test adequacy metrics, we analyze their correlation with the number of identified bugs. Our findings reveal a strong correlation between the TISA metrics and bug count. In contrast, we observe zero, weak, or inconsistent correlations with other test diversity metrics studied. This indicates that the TISA metrics provide a unique perspective on system diversity, complementing existing methods. Additionally, their low computational cost compared to alternative approaches further justifies their suitability for assessing large test suites' adequacy.

Overall, the main contributions of this study are as follows:

- We comprehensively investigate the state-of-the-art diversity metrics for testing AI-based systems and assess their effectiveness in terms of fault detection.
- We propose a set of novel metrics, called *TISA*, which provide visual insights into the diversity and coverage of test scenarios along with providing an objective measure of the quality of testing. We assess the effectiveness of the proposed metrics for their correlation with the fault detection capability of the test suites.
- We analyze how TISA metrics correlate with state-of-the-art diversity metrics.
- We assess the performance of TISA and other selected metrics in terms of computation time.
- A comprehensive replication package to reproduce the results obtained in this study is made available at https://doi.org/10.5281/zenodo.7784015.

2 BLACK-BOX TESTING ADEQUACY MEASURES

For this study, we select widely used adequacy metrics for black-box system testing of AI-based systems. The available coverage based metrics are either white-box [34, 40, 51, 64, 78], or task specific [9, 36, 45, 79], hence do not fit well for this study. Therefore, all the metrics we selected are diversity-based adequacy measures.

2.1 Shannon's Diversity Index

Shannon's Diversity Index is a widely accepted measure of diversity in Ecology, where diversity is defined as "the variety and abundance of species in a defined unit of study" [52]. It measures the diversity of a sample in terms of richness and evenness. *Richness* measures the number of species present in a population, while *evenness* measures the number of individuals per specie [52].

This concept of diversity has been adapted in software engineering, where a dataset is considered diverse, over a particular attribute, if it contains a rich variety of values of the attribute, and those values are evenly abundant [65]. It has been used as a measure of diversity in software testing and defect prediction studies [11, 15].

The Shannon's Diversity Index is mathematically calculated using equation 1:

$$div_{Shannon}(X) = -\sum_{i=1}^{s} p(x_i) \ln p(x_i) \tag{1}$$

where X is the attribute under consideration, $\{x_1, x_2 \cdots x_n\}$ is the set of possible values of X, $p(x_i)$ is the probability that X will take the value x_i. The probability $p(x_i)$ is the ratio of the number n_i of instances having x_i value for the attribute, to the total number N of individuals in the set, i.e. $p(x_i) = \frac{n_i}{N}$.

$div_{Shannon}(X)$ is directly proportional to the level of diversity within a single attribute; a higher value indicates a richer variety and even distribution. However, comparing $div_{Shannon}(X)$ values across attributes can be challenging as it is scaled to the number of possible attribute values, which can create discrepancies between attributes with varying levels of possible values. To address this, an *evenness measure* has been adapted to normalise the value of $div_{Shannon}(X)$ by its maximum possible value, ensuring that all attributes are considered equal [15]. We utilise this normalised Shannon's Index in our analysis.

2.2 Euclidean Diversity

Euclidean diversity (div_{Eu}) is the measure of the distance between two test instances in an Euclidean space. This measure of diversity has widely been used as an adequacy metric in the testing of AVs [13, 14, 39, 49, 50]. We calculate div_{Eu} following the approach adopted in [50], which is summarised below.

As different features of test scenarios are represented by a wide range of values, the feature values are first scaled between $[0, 1]$ using Min-Max normalisation [70]. Next the diversity of the k^{th} feature of two scenarios S_i, S_j is calculated as below:

$$div_{feat_k} = nor(|f_{ik} - f_{jk}|) \tag{2}$$

where f_{ik} and f_{jk} are the k_{th} feature values of S_i and S_j respectively. The diversity of S_i and S_j is then calculated by aggregating the diversities of all the features of these scenarios.

$$div_{i,j} = \frac{\sum_{k=0}^{m} div_{feat_k}}{m} \tag{3}$$

where m is the total number of features used to define the scenarios. The diversity of a scenario S_i is then calculated as:

$$div_S = \sum_{j=0}^{ns} div_{i,j}, j \neq i \tag{4}$$

where ns is the total number of scenarios.

2.3 Normalised Compression Distance

Normalised Compression Distance (NCD) is a feature-free, parameter-free, and alignment-free similarity metric based on compression [20], and has found many applications in clustering, classification, pattern recognition, software testing, and phylogeny [5, 7, 19, 21, 37, 41, 42, 44]. The metric is based on *Kolmogorov complexity* [43] and *Information distance* [10]. *Kolmogorov complexity* defines the information in a single object, while *Information distance* [10] measures the information required to transform one object into the other, among a *pair* of objects.

Cohen et al. extended NCD to support its application for calculating the similarity of multisets [20]. Due to the complexity of Kolmogorov complexity, they approximate it through real-world compressors [47]. However, the choice of the compressor is important as different compressors would impact the performance of NCD in terms of computation time, compression distance and used memory [29]. We followed Aghababaeyan et al. and used *Bzip2* that was found to be the best in their experiments for computing NCD for black-box testing of DNNs [5].

For a multiset S, NCD metric is calculated through an intermediate measure NCD_1 [5, 20]:

$$NCD_1(S) = \frac{C(S) - min_{s \in S}\{C(S)\}}{max_{s \in S}\{C(S) \setminus \{s\}\}} \quad (5)$$

$$NCD(S) = \max\{NCD_1(S), max_{Y \subset S}\{NCD(Y)\}\} \quad (6)$$

where C(S) is the length of the multiset (S) after compression. One limitation of NCD is the high computational cost that makes it impractical for big datasets.

2.4 Standard Deviation

Standard deviation, STD, measures the variability or dispersion of data around the mean value and is widely recognised as a robust statistical measure to quantify the variability or spread of a test suite [5, 16, 63]. For calculating STD for this work, the feature values are first scaled between $[0, 1]$ using Min-Max normalisation [70], and STD is then computed as the norm of the STD of each feature.

3 TEST SUITE INSTANCE SPACE ADEQUACY METRICS

Test suite Instance Space Adequacy (TISA) metrics measure the test suite adequacy from a coverage and diversity perspective. Along with providing an objective measure of the adequacy of a test suite, these metrics offer valuable insights into the relationship between the structural properties of the test instances and their impact on test outcomes. To estimate TISA metrics, we employ a framework called Instance Space Analysis (ISA) [62] that projects test instances – characterised in terms of features – from n-dimensional feature space to a 2-dimensional space, called *Instance Space* (IS). The projections are performed in such a way that there is a clear distinction between failing and passing test scenarios, and the impact of each feature on the test outcome (fail or pass) can easily be mapped. In the context of AV testing, a test scenario is deemed effective if it reveals a bug in the system. As we are using black-box system testing of AVs as a case study, an effective scenario is one that reveals a failure of the AV. Depending on the particular module of the AV under test, a failure could be a collision, driving dangerously close to other vehicles or obstacles, or being unable to drive within the lanes etc.

3.1 Generation of Instance Space

For the generation of instance space, three types of spaces are required (Figure 1):

- Test Scenario Space T: In the context of testing AVs, T encompasses all potential test scenarios that could be utilised. Within this space, a specific subset of scenarios, labelled as T' is chosen to comprise the test suite used in the present study.
- Feature Space F: It consists of a vector of meaningful features that define a test scenario. Features are domain-specific, and their extraction requires significant domain knowledge [60, 61].
- Performance Space P: It represents the performance of the test scenarios measured on the basis of the metric to assess the effectiveness of a scenario, e.g., number of collisions, violation of safety distance, out-of-bound episodes (OBEs) etc.

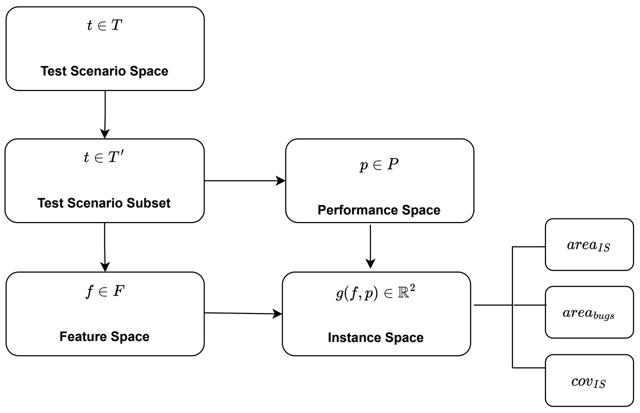


Figure 1: Instance Space Analysis for AV testing.

An instance space is the $2D$ representation of the test scenarios, defined in terms of features having maximum impact on scenario outcome. Therefore, *identification of the features having maximum impact on test outcome* is one of the most crucial steps in ISA. Feature identification and selection is an iterative process that uses machine learning techniques to find significant features that clearly differentiate effective, i.e., failure-revealing scenarios, from the passing ones. The initial stage of this process involves identifying a cluster of features that share similarities with each other. We use k-means clustering for this process as it is one of the simplest unsupervised machine learning algorithms and ISA is shown to perform well with this clustering technique in the previous studies [59–62]. The optimal number of clusters to be used by k-means is selected using silhouette analysis [8].

Once clusters are created, one feature from each cluster is taken to create a feature set. Assume that the number of clusters generated in the previous step is n, then each feature set will contain n features. This n-dimensional feature set is projected to a temporary $2D$ space using *Principal Component Analysis* [4]. The process is repeated for all the feature sets created by the possible combination of features from all the clusters. The coordinates of the temporary $2D$ spaces then become the input to a set of Random Forest (RF) models, which learn the feature combination giving the lowest predictive error in predicting the scenario outcome.

Now that the set of most effective features has been identified, we project the test instances defined in terms of nD feature space to a $2D$ coordinate system in such a way that the relationship between the features of the instances and the test outcomes can easily be identified. An ideal projection is one that creates a linear trend when each feature value and scenario outcome is inspected, i.e., low values of features/scenario outcomes at one end of a straight line and high values at the other. Furthermore, the instances that are neighbours in high dimensional feature space should remain as neighbours in the 2D instance space (topological preservation). These projection goals are achieved by using a projection method called *Projecting Instances with Linearly Observable Trends* (PILOT) [60], which seeks to fit a linear model for each feature and test outcome, based on the instance location in the $2D$ plane. Mathematically, this involves solving the following optimisation problem:

$$\begin{aligned} \min \quad & \left\|\tilde{\mathbf{F}} - \mathbf{B}_r\mathbf{Z}\right\|_F^2 + \|\mathbf{Y} - \mathbf{C}_r\mathbf{Z}\|_F^2 \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{A}_r\tilde{\mathbf{F}} \end{aligned} \tag{7}$$

where $\tilde{\mathbf{F}}$ is the matrix containing the n features of a test scenario, $\mathbf{Y}$ is the column vector containing scenario outcome, $\mathbf{Z} \in \mathbb{R}^{i\times 2}$ is the matrix containing z_1 and z_2 coordinate values of i scenarios in the $2D$ space, $\mathbf{A}_r \in \mathbb{R}^{2\times n}$ is a matrix that takes the feature values and projects them in $2D$ space, $\mathbf{B}_r \in \mathbb{R}^{n\times 2}$ is a matrix that takes the $2D$ coordinates and produces an estimation of the feature values, and $\mathbf{C}_r \in \mathbb{R}^{t\times 2}$ is a matrix that takes the $2D$ coordinates and makes an estimation of the technique's performance. In short, Equation 7 finds the difference between the actual values of the features and performances in a higher dimension and the estimation of these values in $2D$. The lower the difference, the higher the topological preservation. Mathematical proofs and additional technical details of PILOT are available in [60].

3.2 Computation of TISA Metrics

The adequacy metrics proposed by ISA can not be understood completely without visualising the instance space itself. For this, using the methodology explained in the above sections, we create an instance space and present its important components in Figure 2. The test suite for the generation of this instance space is obtained from a previous study on test-case prioritisation for AVs [13]. Detailed information about the features of these test scenarios is provided in Section 4.2, where we discuss the dataset and other relevant aspects of our experiments.

Figure 2a shows the distribution of test scenarios in the instance space based on their outcome. The blue points represent the effective test scenarios that push the AV to a safety critical situation (failed), while the red points represent safe scenarios (pass). It can be seen that the scenarios are distributed in such a way that there is a clear distinction between safe and unsafe scenarios, and it is easy to visualise the diversity of the bugs, as well as the diversity of the whole test suite. Figures 2b and 2c show the distribution of two of the features of test scenarios across the instance space. These are the features of the test instances that are identified to have the maximum impact on scenario outcome. The feature distributions are projected in the instance space in such a way that feature values are increasing/decreasing from one end of the space to the other, and are easy to map to the test outcome. It can be seen that for both features, the lower values result in safer scenarios, while medium to high values cause failure.

3.2.1 Area of the Instance Space ($area_{IS}$). This is the area occupied by all the test instances enclosed by the instance space and measures the diversity of the whole test suite. In Figure 2a, $area_{IS}$ represents the area of the region occupied by all the red and blue dots.

$area_{IS}$ provides a robust and defensible test adequacy metric, as increasing this area results in increasing the coverage of a broad range of test conditions and, therefore, improves the quality of testing. It is important to note here that the value of $area_{IS}$ will not increase by adding more instances to the instance space, until the feature values of new instances are different from the previous ones and add more diversity to the test suite. Test scenarios having similar features project on top of or very close to each other, therefore, do not increase the area of the instance space.

3.2.2 Area of Buggy region ($area_{bugs}$). This is the area of the instance space occupied by the majority of the test instances that reveal failure. In Figure 2a, this is the area of the instance space dominated by blue points (right half of the instance space).

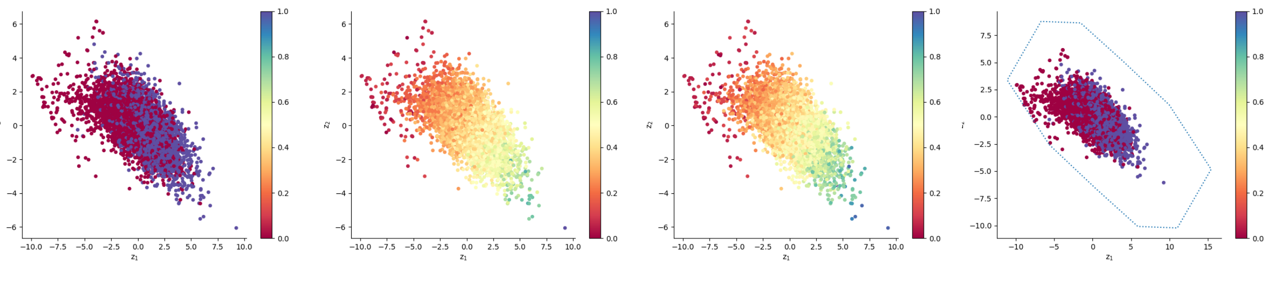
The number of bugs revealed by a test suite is a common measure of assessing the quality of testing for an AI-based system [3, 6, 46, 80]. However, for any AI-based system in general, and for AVs in particular, solely focusing on finding more bugs may overlook critical safety issues if the test suite does not cover a diverse range of scenarios, environmental conditions, and edge cases. Therefore, the above-proposed measure of assessing the quality of a test suite in terms of the diversity of the bugs detected appears to be both reasonable and robust.

Area calculation: For area calculation, we rely only on high-density regions of instance space to ensure that the values of $area_{IS}$ and $area_{bugs}$ won't get impacted by the outliers. For this, we use DBSCAN [71], a clustering algorithm that defines a cluster as a dense region of points. DBSCAN takes two parameters $\{k, \varepsilon\}$ as input, where k represents the minimum number of instances required to form a dense region, while ε represents the maximum radius of an instance required for it to be considered as a neighbour. The metric used to measure the distance is Euclidean. The values for k and ε are chosen automatically as recommended in [22], using Equations 8 and 9.

$$k \leftarrow \max\left(\min\left(\lceil r/20\rceil, 50\right), 3\right) \tag{8}$$

$$\varepsilon \leftarrow \frac{k\Gamma(2)}{\sqrt{r\pi}}\left(\text{range}(z_1) \times \text{range}(z_2)\right) \tag{9}$$

Where r is the number of failed scenarios in the calculation of $area_{bugs}$ and all the scenarios otherwise, $\Gamma(\cdot)$ is the Gamma function, and z_1, z_2 are the coordinates of the $2D$ instance space.



(a) Scenario outcome distribution (b) Num right turns (c) total angle (d) footprint

Figure 2: Instance Space Analysis for AV testing (a) shows the distribution of scenarios' outcome; (b and c) show the feature distributions that can be mapped to the scenario outcome; (c) shows the mathematical boundary created around the instance space for estimating the coverage of the test suite.

The footprint of a cluster is created using an α-shape, which is a generalisation of the convex hull concept from computational geometry [28]. It corresponds to a polygon that closely surrounds all the points within a cloud. An α-shape is created for each cluster, and all shapes are bounded together as a MATLAB polygon structure [2]. The area of the polygon is then computed using MATLAB Polyarea function [1].

In calculating the area of the bugs ($area_{bugs}$), there could be regions of the space where there is an overlap of buggy and bug-free test scenarios. The decision of whether to include such a region in the area calculation depends on the relative number of buggy instances in this area. If the number of bugs is higher than the bug-free instances, the region will be included, ignored otherwise.

3.2.3 Coverage of Instance Space (cov_{IS}). The instance space is the 2*D* representation of the test instances available in the test suite under study. These are represented as *Test Scenario Subset* (T') in Figure 1. However, these scenarios do not necessarily include all the test instances required to explore the feature space completely. Using ISA, we devise a way to compute a mathematical boundary that encloses all the test scenarios which are empirically possible to generate, though, may be missing from the current instance space/test suite. This complete set of scenarios is indicated as *Test Scenario Space* (T) in Figure 1.

Boundary Creation: Let $\mathbb{R}^{n\times n}$ be the correlation matrix of n features, defining a test scenario. We define two vectors $\mathbf{f}_U = \left[f_{U_1} \cdots f_{U_n}\right]^T$ and $\mathbf{f}_L = \left[f_{L_1} \cdots f_{L_n}\right]^T$ containing the upper and lower bounds of feature values. We define a vertex vector from these two vectors containing a combination of values from f_U and f_L such that only the upper or lower bound of a feature is included. For instance, $\mathbf{v}_1 = \left[f_{U_1} f_{L_2} \cdots f_{L_n}\right]^T$ represents a vertex vector containing the maximum value of feature 1 and minimum values of all the other features. We define a matrix $\mathbf{V} = \left[\mathbf{v}_1 \cdots \mathbf{v}_q\right] \in \mathbb{R}^{n\times q}$, $q = 2^n$ containing all possible vertices created by the feature combinations. The vertices in metric $\mathbf{V}$, connected by edges, define a hyper-cube that surrounds all the instances in the instance space.

Some of the vectors in $\mathbf{V}$ represent feature combinations that are unlikely to coexist. For example, if features 1 and 2 are strongly positively correlated, an instance having a high value of feature 1 and a low value of feature 2 is unexpected to be found. Therefore, a vertex vector $\mathbf{v} = \left[f_{U_1} f_{L_2} \cdots f_{L_n}\right]^T$ would be unlikely to be near any true instance. Similarly, a vertex vector cannot simultaneously contain $\{f_{U_1}, f_{U_2}\}$ or $\{f_{L_1}, f_{L_2}\}$ if feature 1 and feature 2 are strongly negatively correlated. All such unlikely vertex vectors are eliminated from $\mathbf{V}$. The edges connecting the remaining vertex vectors are then projected into 2*D* instance space using PILOT [60], whose convex hull now represents the mathematical boundary of the expanded instance space. Figure 2d shows the boundary created around the instance space shown in figure 2a.

The coverage of the instance space is defined as *the percentage of the bounded area ($area_{bound}$) covered by the instance space.*

$$\mathsf{Cov}_{IS}(\%) = \frac{\mathsf{area}_{IS}}{\mathsf{area}_{bound}}\,100 \tag{10}$$

As can be seen in figure 2d, there are vast empty regions around the instance space, depicting that the test suite lack sufficiency to test the system reliably. Increasing the coverage of the space, especially in the *high bug probability area*, would improve the effectiveness of the test suite by testing a wider range of conditions and scenarios.

4 EXPERIMENTAL DESIGN

4.1 Research Questions

Our empirical study is steered by the following research questions:

4.1.1 RQ1. How effective are the TISA metrics? Under this research question, we want to investigate the effectiveness of ISA metrics in terms of fault detection. This is done by performing the correlation analysis of the TISA metrics computed for the samples of test scenarios with the number of bugs detected [5, 18, 29]. In our experiments, the buggy scenarios are the ones that lead AV to collision, break safety distance or cause out-of-bound episodes (OBE). These are the widely used measure of failure in the system testing of AVs [6, 12, 32, 46, 50, 66]. We also investigate such correlation for the other diversity metrics selected for this study.

4.1.2 RQ2. How do the TISA metrics correlate with existing diversity adequacy metrics? There is a plethora of literature available on diversity measures for testing [29, 30]. However, different diversity metrics are designed to measure the diversity of the underlying dataset from a different perspective. For instance, Euclidean distance is a measure of the straight-line distance between two points in a multidimensional space. It is commonly used to measure the similarity or dissimilarity between two test scenarios based on the values of their features. On the other hand, the Shannon diversity index is a measure of the diversity of a set of objects based on their relative abundance. It is commonly used to measure the diversity of test scenarios in terms of their frequency or distribution across different categories or features. Both measures can be useful in assessing test diversity, however, their effectiveness is dependent on the specific context and goals of the testing process. Under this research question, we investigate how TISA metrics correlate with other state-of-the-art diversity metrics to investigate their uniqueness or redundancy.

4.1.3 RQ3. How efficient are the ISA adequacy metrics? We want to compare ISA metrics with other diversity metrics in terms of computation time. As testing of sophisticated AI systems, like AVs, requires executing an enormous number of test scenarios to cover wide driving conditions, computation time becomes one of the leading factors in deciding the acceptability of a metric.

4.2 Dataset

The current study employs test suites from two previous studies on search-based test selection and prioritisation for AVs [13, 49].

The first test suite is generated for testing Baidu Apollo Open Platform 5.0 using the LGSVL simulator [69]. It includes scenarios representing different types of roads in San Francisco. Failures in this dataset are defined as collisions or safety distance violations [73]. The suite consists of 28,964 unique test scenarios, with 12,932 (45%) being buggy scenarios.

The second test suite, generated by As-Fault [32, 33], combines search-based testing and procedural content generation for lane-keeping feature testing. Failures in this dataset are defined as Out Of Bound Episodes (OBEs). The scenarios focus on road features like turns and straight segments. The suite contains 5,639 test scenarios, with 2,541 (45%) being buggy scenarios.

4.3 Sampling for Experiments

To conduct a correlation analysis between adequacy metrics and the occurrence of bugs, it is crucial to have a diverse range of test suite samples with varying bug counts. To achieve this, we follow a systematic approach of generating samples by modifying the number of bugs within a range of 5% to 75%.

To begin, we divide the dataset into two subsets: one comprising failed scenarios and the other consisting of safe driving scenarios. From the failed test scenarios, we sample $n\%$ of the scenarios, while the remaining $100 - n\%$ is sampled from the safe subset. By combining these sampled scenarios from both subsets, we create an experimental sample.

In total, we generate 30 such samples for our experiments, each containing 2500 scenarios. We start with $n = 5$ and increment it by 5 for each subsequent sample until n reaches 75. If n reaches 75 and the number of samples is still less than 30, we reset the value of n to 5 and continue generating additional samples.

5 EXPERIMENTAL RESULTS

5.1 RQ1. Effectiveness of TISA metrics

We aim to investigate the effectiveness of TISA metrics in terms of how well they correlate with fault detection.

For correlation analysis, we use Spearman Rank Correlation, as it is a non-parametric method and does not require assumptions of linearity or normality of data [23], therefore, is widely used in similar studies for correlation analysis [5, 25, 26, 86].

Figures 3 and 4 show the correlation of TISA and other selected metrics with fault detection. The x-axis plots the diversity values computed by different diversity measures, while the count of bugs is shown at y-axis. The correlation value (ρ) is shown in red if the correlation is statistically significant (p-value <= 0.05), black otherwise.

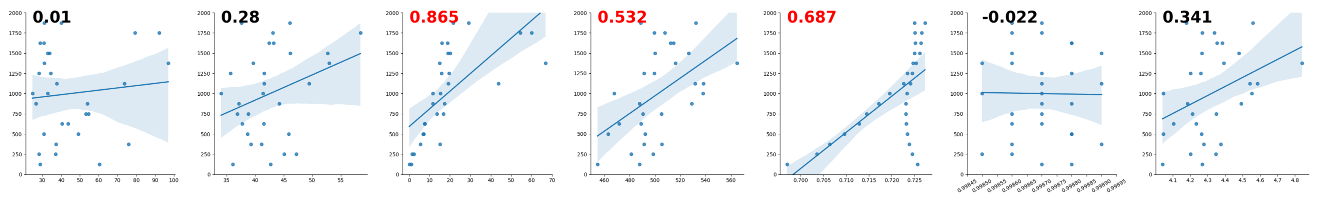
To consider a measure as a good estimator of fault detection, we anticipate a high and statistically significant correlation. This correlation should be consistent across different test suites. Our observations indicate that all TISA metrics exhibit a positive correlation with the number of bugs in both test suites. Among these metrics, the $area_{bugs}$ demonstrates a very high correlation that is statistically significant. On the other hand, the correlation for the cov_{IS} metric is moderate and statistically insignificant. As for the $area_{IS}$ metric, it shows a very low correlation with bugs in both test suites.

Euclidean Distance (div_{Eu}) shows a moderate to high positive correlation with bugs in our experiments. However, this correlation is not consistently significant across the datasets.

div_{Eu} and $area_{bugs}$ measure the spatial separation between data points in a multidimensional and $2D$ space, respectively. As the bug occurrences in the test suites under study exhibit distinct spatial patterns or clusters (as shown in Figure 2a), such metrics may better capture the proximity or dissimilarity between data points, thereby reflecting the presence of bugs more accurately as compared to other metrics. As the instances in $2D$ instance space are projected in a manner that aligns the distribution of features with the distribution of test outcomes, $area_{bugs}$, which is based on the instance space, exhibits a stronger correlation with the distribution of bugs compared to the div_{Eu}.

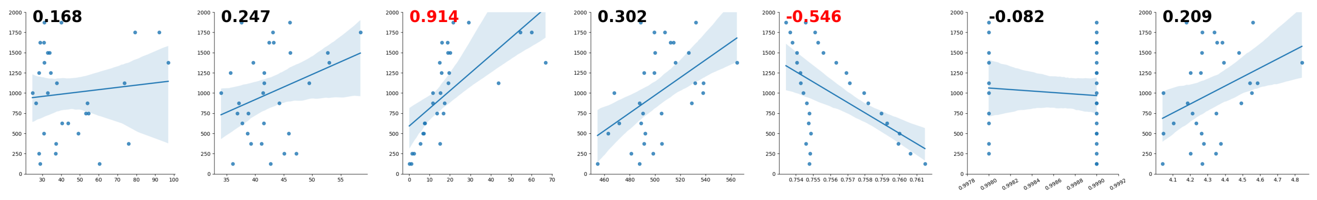
Among other metrics, $div_{Shannon}$ shows a strong positive correlation in test suite 1, however, the relationship becomes negative in test suite 2. Due to the correlation value of this metric changing signs across the test suite, its effectiveness is questionable, and we cannot place confidence in these findings. $div_{Shannon}$ focuses on the richness and evenness of test instances. It can help identify datasets with a high richness of different variables or dimensions, and can also highlight datasets where the abundance is distributed more evenly or unevenly across the variables. However, it may not capture other aspects of the data's diversity, such as patterns, interactions, or specific relationships between variables. Therefore, this measure doesn't show a consistent relationship with the number of bugs in our experiments.

Both NCD and STD exhibit either no correlation or statistically insignificant correlations with bugs in both test suites. NCD is primarily designed to measure the similarity between two data sequences based on their compressed lengths. However, in our test suites, where the majority of input features are numeric (e.g.,



(a) $area_{IS}$ (b) cov_{IS} (c) $area_{bugs}$ (d) div_{Eu} (e) $div_{Shannon}$ (f) NCD (g) STD

Figure 3: Correlation between selected diversity metrics and number of bugs Using Spearman's Rank Correlation Coefficient: test suite 1



(a) $area_{IS}$ (b) cov_{IS} (c) $area_{bugs}$ (d) div_{Eu} (e) $div_{Shannon}$ (f) NCD (g) STD

Figure 4: Correlation between selected diversity metrics and number of bugs Using Spearman's Rank Correlation Coefficient: test suite 2

speed, distance, throttle, number of right turns, total angle), it is challenging to represent these numeric variables as data sequences that can be effectively compressed. As a result, NCD does not serve as a strong measure of diversity in our experiment.

On the other hand, STD is sensitive to outliers, which can disproportionately affect its calculation and potentially lead to misleading interpretations of data spread or diversity. Moreover, STD assumes a symmetric distribution around the mean, which may not accurately represent the dispersion of values in the presence of skewed distributions.

These limitations in the effectiveness of NCD and STD emphasise their limitations in accurately capturing diversity in our study.

Answer to RQ1: All TISA metrics exhibit a positive correlation with faults, but $area_{bug}$ stands out with a consistently strong and statistically significant correlation, highlighting its effectiveness in detecting faults and reliability in providing valuable insights into the effectiveness of the testing process.

5.2 RQ2. Relationship of TISA metrics with selected diversity metrics

We want to investigate how TISA metrics correlate with other adequacy metrics selected for this study. Understanding the correlation between test adequacy metrics can provide useful insights into redundancy or overlaps between them. For instance, if two metrics are highly correlated, they are likely to measure the same aspect of diversity, and thus it may be more efficient to use only one of them.

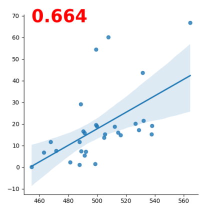
Section 5.1 shows that $area_{IS}$ has a weak correlation with the bugs in the system, while this correlation is not statistically significant for cov_{ISA}. As we are interested in determining the effectiveness of TISA measures in terms of fault detection, we will consider only $area_{bugs}$ for answering RQ2 and RQ3.

Figures 3 and 4 show the correlation of $area_{bugs}$ with div_{Eu}, $div_{Shannon}$, NCD and STD for test suites 1 and 2. The correlation values are coloured red if the correlation is statistically significant, and black otherwise. The metric that shows the strongest positive correlation with $area_{bug}$ is div_{Eu}, however, this correlation is not consistently significant across the test suites. STD shows a similar trend as div_{Eu}; positively strong and statistically significant correlation for one test suite, however, weak and insignificant correlation for the other. Just like the correlation with the number of bugs, $div_{Shannon}$ changes the sign of correlation across test suites, therefore making the results unreliable. Lastly, NCD consistently shows a negative and weak correlation with $area_{bugs}$.

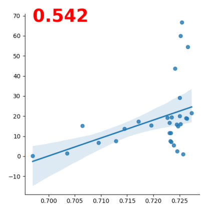
Answer to RQ2: In conclusion, the TISA metric, $area_{bug}$, exhibits a weak or inconsistent correlation with other diversity metrics, suggesting that it captures data diversity from a unique perspective compared to the other metrics analysed. However, due to its strong correlation with the number of bugs and its emphasis on high-impact features, $area_{bug}$ is suitable for use as a black-box adequacy measure for diversity.

5.3 RQ3. The efficiency of TISA metrics

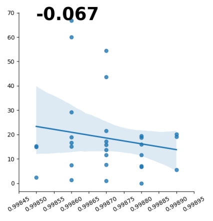
The testing of AI systems, in particular AVs, requires a huge number of test scenarios to cover wide driving situations on real roads. Therefore, the most appropriate adequacy metric is one that is



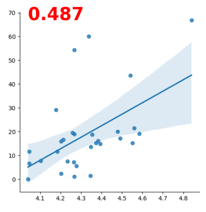
(a) div_{Eu}



(b) $div_{Shannon}$

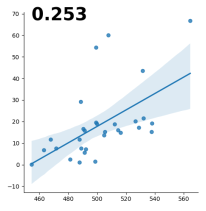


(c) NCD

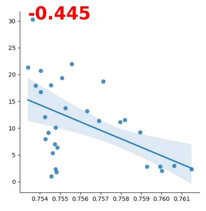


(d) STD

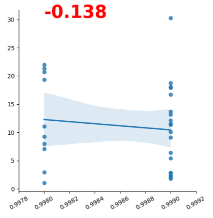
Figure 5: Spearman's Rank Correlation between $area_{bug}$ and other diversity metrics for test suite 1



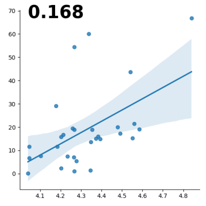
(a) div_{Eu}



(b) $div_{Shannon}$



(c) NCD



(d) STD

Figure 6: Spearman's Rank Correlation between $area_{bug}$ and other diversity metrics for test suite 2

practically efficient as well as effective. We want to assess the efficiency of the TISA and other selected adequacy metrics in terms of computation time. For this, we selected 8 samples comprising 500, 1000, 1500, 2000, 2500, 3000, 5000 and 10,000 test scenarios. We applied the selected diversity measure to all the samples and recorded the execution time in seconds. Similarly, we generated the instance spaces for all the samples, computed the TISA metrics, and recorded the time of the whole process. All the experiments are conducted on an Ubuntu server with 32 GB memory and an Intel Core 5.20GHz processor.

The graph presented in Figure 7a illustrates how the computation time of different metrics changes as the sample size increases. The computation time of NCD is exponentially high compared to all other metrics. The computational complexity of this metric is generally proportional to the length of the input data, and the performance of the compression algorithm. However, a previous study found that even when computing this metric using various compression algorithms to determine the optimal one, the computation time for NCD remains impractical for larger datasets, even with the best compression algorithm [5]. This makes this metric infeasible for testing AI-based systems, which usually have a huge number of test scenarios, with many features. The execution time of div_{Eu} is less than that of NCD, but it is still considerably higher than other metrics. Moreover, the execution time of div_{Eu} increases exponentially once the sample size surpasses a certain threshold.

To view the trends of STD, $TISA$ metrics and $div_{shannon}$ more clearly, which are hidden behind each other in Figure 7a, we have plotted their computation time separately in Figure 7b. The computation time of STD remains almost constant, while $div_{Shannon}$ has a slight increase in the time with the increase in sample size. As compared to these two metrics, $TISA$ takes more time to generate the instance space and compute the adequacy results. However, even for a large test suite having 10k test scenarios, this time is still under 50 seconds. It is important to note here that, unlike other metrics, the computation time for TISA includes the generation of instance space, adequacy metrics computation, and generation of feature and performance distribution graphs. Therefore, irrespective of the relative difference of computation time of $area_{bug}$ compared to div_{Eu}, $div_{Shannon}$ and STD, this metric is quite efficient considering its dual advantage of providing the visual insights into the diversity along with a numeric value.

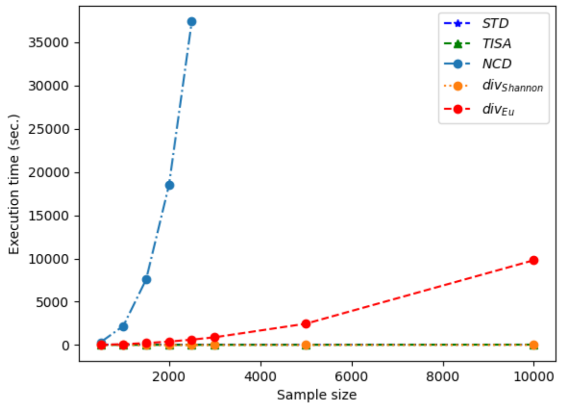
Answer to RQ3: In summary, the computational cost of TISA metrics is not high. Additionally, due to their ability to visualise diversity and effectively detect bugs, these metrics are well-suited for black-box testing of AVs.

6 RELATED STUDIES

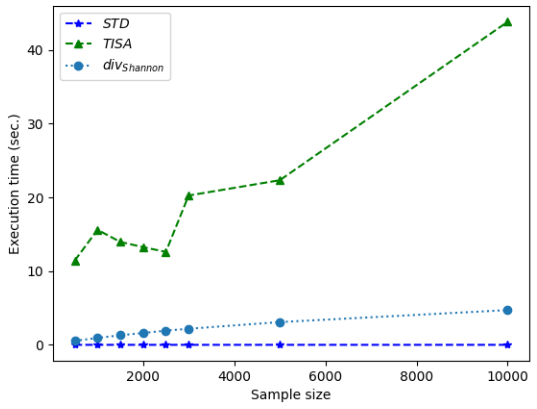
6.1 Model-level Testing

The majority of testing methods employed for AVs and other AI-based systems primarily revolve around model-level testing. [24, 31, 34, 40, 51, 64, 75, 76, 78, 81, 85].

DeepRoad [85] suggests employing Generative Adversarial Networks (GANs) to generate driving inputs that closely resemble real-world data streams, thereby enhancing the realism of the generated inputs. DeepTest [81] leverages affine transformations commonly used in computer vision to generate novel inputs that induce misbehaviour in the deep neural network (DNN). DeepXplore [64] employs white-box testing techniques to generate inputs that aim to achieve maximum coverage of neurons and diverse behavioural outcomes. In their study, Kim et al. [40] introduce a set of white-box test adequacy criteria based on the notion of surprise. Surprise is defined as the variation in a deep neural network's behaviour when presented with a new test input compared to its training data. The researchers propose generating test inputs that cover a wide range of surprise levels. Deng et al. [24] examine the robustness of DNN



(a) Execution time of adequacy metrics for different sample sizes



(b) Magnified view of the execution time of STD, $TISA$ and $div_{Shannon}$ from Figure 7a

Figure 7: Computation time taken by adequacy metrics for different sample sizes

models against adversarial attacks. ThirdEye [75] utilises attention maps, derived from the field of explainable AI, to anticipate the occurrence of misbehaviour in AVs. Feng et al. propose *DeepGini* [31], a test prioritisation technique based on the principle that a test case for which the model classifies all the labels/classes with similar probabilities indicates that the model has lower confidence in its classification and is more likely to make mistake as compared to a test case with a significantly higher probability for one label compared to others.

In contrast to the aforementioned studies, our research focuses specifically on conducting "black-box system-level testing" for autonomous vehicles (AVs). In system-level testing, the DNN is integrated into the operational environment where it is intended to operate, and any identified test failures are characterised in terms of the overall system's malfunction or misbehaviour. Furthermore, unlike the majority of model-based approaches mentioned earlier, our study employs test cases that replicate real driving scenarios through driving simulations. These simulations encompass various elements such as road conditions, weather conditions, AV behaviour, and other vehicles present on the road. In contrast, the datasets used in many of the mentioned model-based approaches typically consist of images as test cases, with the AI model being evaluated primarily for image classification or other tasks at the model level.

6.2 System-level Testing

This section reviews recent work done for system-level black-box testing of AVs. Specifically, we'll be discussing the studies using diversity and coverage metrics within this context.

6.2.1 Diversity. Diversity is an apparent indicator of quality because it directly indicates how thoroughly the system is tested across various conditions and scenarios. It is a widely used adequacy measure in traditional software [5, 9, 36, 45, 53, 79] and also deemed important for testing of AI-based systems [13, 27, 29, 50, 56]. We will summarise work related to the use of diversity in black-box testing of AI-based systems in this section.

Birchler et al. prioritise test cases based on diversity and execution cost using static road features [13, 14]. Lu et al. prioritise test cases based on diversity, demand, collision information and collision probability [50]. Ebadi et al. generate test scenarios for pedestrian detection, measuring effectiveness based on fault detection and diversity [27]. Moghadam et al. use bio-inspired algorithms to test lane keeping functionality of AVs, measuring effectiveness in terms of diversity and the number of bugs detected [56]. Vincenzo et al. use roads' dissimilarity measured in terms of weighted Levenshtein distance to evaluate how two test cases are different from each other [68].

What makes our work different from these studies is that we propose a novel set of adequacy metrics that assess the quality of a test suite, both in terms of diversity and coverage. The proposed framework also facilitated visual insights by projecting the test scenarios defined in terms of n features, from n-dimensional feature space to $2D$ instances space.

Aghababaeyan et al. performed a correlation analysis of black-box diversity measures (Geometric Diversity [44], Normalised Compression Distance [20], and Standard Deviation) with fault detection [5]. They found that Geometric Diversity had the highest and statistically significant correlation. Our work is similar in analysing diversity metrics' correlation with bugs but introduces new metrics with medium to high fault detection correlation, outperforming other metrics in fault detection and computation time. Furthermore, Geometric Diversity, assessed on image datasets using VGG-16 [58], requires a large number of features, which are not feasible for system testing. Therefore, this diversity measure cannot be used in the context of black-box system testing.

6.2.2 Coverage. The majority of the approaches proposing black-box coverage metrics for testing AVs are in the white-box domain and at model level [34, 40, 51, 64, 78]. A limited work introducing black-box coverage criteria for testing AI-based systems is discussed below:

Hauer et al. [36] propose a statistical approach to ensure all possible scenario types, like overtaking, lane change, etc., are covered in the test suite. Arcaini et al. consider low-level driving characteristics as scenario types for coverage measurement [9]. Tang et al. classify scenarios based on map topological structures and evaluate coverage accordingly [79]. Laurent et al. [45] use weight coverage to cover different configurations of a path planner. Unlike these studies, our work introduces a coverage metric that provides a quantitative measure of quality. Furthermore, our proposed framework is generic and can be applied to the black-box testing of any AI-based system.

A closely related work is presented by Zohdinasab et al. [87], who propose DeepHyperion, a framework that analyses the impact of features on Deep Learning (DL) system quality using illumination search [57]. DeepHyperion evaluates test instances based on a fitness function, visualising the results in a $2D$ map. However, its manual open coding procedure for feature labelling limits its scalability and makes it impractical for testing instances with many features. Additionally, DeepHyperion presents feature maps in an n-dimensional coordinate system, which restricts the visualisation of the combined impact of more than three features. In contrast, our framework projects test instances from an n-dimensional feature space to a $2D$ instance space, providing intuitive and easy-to-analyze visualisations.

7 THREATS TO VALIDITY

Internal validity threats arise when the results are influenced by internal factors. One such factor is the lack of source code for the selected diversity metrics, which could affect our implementation. To mitigate this, we extensively tested our code against the original paper. Another threat is the limited variability in the number of bugs in the samples due to random sampling. We addressed this by using a custom sampling method to ensure diverse bug counts. Lastly, the configuration of hyperparameters for machine learning methods could pose a threat. To mitigate this, we employed established techniques such as silhouette analysis [8] for k-means and an automated method for parameter calculation for DBSCAN [22]. These measures strengthen the validity of our study.

Conclusion threats to validity pertain to issues that can affect the accuracy and reliability of the study's conclusions. One such threat in this study is the choice of correlation analysis method, which may be influenced by data distributions or the shape of the relationship. To mitigate this, we use Spearman correlation, which does not rely on assumptions about data distributions but requires a monotonic relationship.

Another such threat concerns the validity of feature combinations used for the estimation of the boundary for coverage calculation. These combinations are learned from test scenarios based on feature correlation analysis. If the test suite contains invalid or unrealistic scenarios, incorrect correlations can be learned, rendering the boundary invalid. To address this threat, we utilise test suites from two previous studies that have reported the generation of valid and realistic driving scenarios [13, 49].

External validity threats impact the generalizability of the results. To minimise these impacts, we selected two different test suites having a different number of features and test instances. Furthermore, we have selected the widely used adequacy metrics published in the studies of black-box testing of AI-based systems.

8 DISCUSSION

The TISA coverage metric ($area_{IS}$), measures the coverage of a test suite in relation to the entire feature space or extended instance space. It represents the percentage of the area covered by the test suite out of the total possible area. While it may not strongly correlate with bug detection, it serves as a valuable objective measure for evaluating the extent to which the test suite explores the feature space effectively. By visualising the coverage achieved, $area_{IS}$ can offer a graphical representation of the coverage landscape. This visualisation allows the testers to observe the areas that are well covered and those that remain unexplored. It can help identify gaps in testing and guide efforts to improve coverage by targeting specific regions of the feature space.

The accuracy of $area_{IS}$ depends on the feasibility of boundary estimation, which involves determining the upper and lower limits of the features associated with the driving scenarios. In the present study, the boundary calculation utilises the maximum and minimum values of the features that are currently available within the test suite. Consequently, the unoccupied regions in Figure 2d indicate the scenarios that are absent relative to the test suite itself. However, the estimation of absolute feature boundaries by defining Operational Design Domains (ODDs)[83] for the feasible driving conditions in which autonomous vehicles (AVs) can theoretically and practically operate, is an area of active research[72, 74]. Some initial efforts have been made to establish these boundaries [38]. Coverage based on absolute boundaries would provide a measure of how well the entire feature space, as defined by the potential driving conditions, is tested, regardless of the specific test suite employed for generating the instance space.

9 CONCLUSION AND FUTURE WORK

In this study, we introduce a set of adequacy metrics, called *TISA* metrics, to assess the quality of black-box testing of AI-based systems. The proposed metrics evaluate the test suite both from a coverage and diversity perspective. They also facilitate insights into the diversity of the bugs identified by the test suite by providing the measure of the diversity of the failed test scenarios. Along with giving the measure of diversity in numeric form, the diversity of the test suite is presented in $2D$ *instance space*. The instance space also provides a clear and precise representation of the test coverage, making it easy to identify if tests for a particular feature combination are missing or if there is a need to remove the redundant ones; therefore, it can be a powerful tool to improve the quality of the test suite.

We assess the effectiveness of the proposed metrics in terms of fault detection. For this, we perform the correlation analysis of these metrics with the number of bugs identified. Furthermore, to test if these metrics are redundant to other diversity metrics common in testing, we performed a correlation analysis of *TISA* metrics with other metrics. Experimental results show that *TISA* metrics show a high correlation with bugs, measure a different aspect of diversity as compared to other diversity metrics, and are computationally efficient.

We expect to extend this work by incorporating TISA metrics in test generation, selection and prioritisation strategies. Furthermore, we plan to generate a test generation strategy that identifies the feature values of the missing scenarios from the instance space and estimates new scenarios using these features.

ACKNOWLEDGEMENTS

The funding for this research is provided by the Australian Research Council under grant DP210100041.

REFERENCES

[1] Mathworks polyarea. https://www.mathworks.com/help/matlab/ref/polyarea.html.

[2] Mathworks polygons. https://au.mathworks.com/help/map/create-and-display-polygons.html.

[3] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 143–154. IEEE, 2018.

[4] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[5] Zohreh Aghababaeyan, Manel Abdellatif, Lionel Briand, and Mojtaba Bagherzadeh. Black-box testing of deep neural networks through test case diversity. *arXiv preprint arXiv:2112.12591*, 2021.

[6] Sumaya Almanee, Xiafa Wu, Yuqi Huai, Qi Alfred Chen, and Joshua Garcia. scenorita: Generating less-redundant, safety-critical and motion sickness-inducing scenarios for autonomous vehicles. *arXiv preprint arXiv:2112.09725*, 2021.

[7] Cecile ANe and Michael J Sanderson. Missing the forest for the trees: Phylogenetic compression and its implications for inferring complex evolutionary histories. *Systematic Biology*, 54(1):146–157, 2005.

[8] S Aranganayagi and Kuttiyannan Thangavel. Clustering categorical data using silhouette coefficient as a relocating measure. In *International conference on computational intelligence and multimedia applications (ICCIMA 2007)*, volume 2, pages 13–17. IEEE, 2007.

[9] Paolo Arcaini, Xiao-Yi Zhang, and Fuyuki Ishikawa. Targeting patterns of driving characteristics in testing autonomous driving systems. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 295–305. IEEE, 2021.

[10] Charles H Bennett, Péter Gács, Ming Li, Paul MB Vitányi, and Wojciech H Zurek. Information distance. *IEEE Transactions on information theory*, 44(4):1407–1423, 1998.

[11] Kwabena Ebo Bennin, Jacky Keung, Passakorn Phannachitta, Akito Monden, and Solomon Mensah. Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. *IEEE Transactions on Software Engineering*, 44(6):534–550, 2017.

[12] Christian Birchler, Nicolas Ganz, Sajad Khatiri, Alessio Gambi, and Sebastiano Panichella. Cost-effective simulation-based test selection in self-driving cars software with sdc-scissor. In *29th IEEE International Conference on Software Analysis, Evolution, and Reengineering, Honolulu, USA (online), 15-18 March 2022.* ZHAW Zürcher Hochschule für Angewandte Wissenschaften, 2022.

[13] Christian Birchler, Sajad Khatiri, Pouria Derakhshanfar, Sebastiano Panichella, and Annibale Panichella. Automated test cases prioritization for self-driving cars in virtual environments. *arXiv preprint arXiv:2107.09614*, 2021.

[14] Christian Birchler, Sajad Khatiri, Pouria Derakhshanfar, Sebastiano Panichella, and Annibale Panichella. Single and multi-objective test cases prioritization for self-driving cars in virtual environments. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2022.

[15] Marcel Böhme. Stads: Software testing as species discovery. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 27(2):1–52, 2018.

[16] Xinye Cai, Haoran Sun, and Zhun Fan. A diversity indicator based on reference vectors for many-objective optimization. *Information Sciences*, 430:467–486, 2018.

[17] Emanuela G Cartaxo, Patrícia DL Machado, and Francisco G Oliveira Neto. On the use of a similarity function for test case selection in the context of model-based testing. *Software Testing, Verification and Reliability*, 21(2):75–100, 2011.

[18] Junjie Chen, Ming Yan, Zan Wang, Yuning Kang, and Zhuo Wu. Deep neural network test coverage: How far are we? *arXiv preprint arXiv:2010.04946*, 2020.

[19] Rudi Cilibrasi and Paul MB Vitányi. Clustering by compression. *IEEE Transactions on Information theory*, 51(4):1523–1545, 2005.

[20] Andrew R Cohen and Paul MB Vitányi. Normalized compression distance of multisets with applications. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1602–1614, 2014.

[21] Dinu Coltuc, Mihai Datcu, and Daniela Coltuc. On the use of normalized compression distances for image similarity detection. *Entropy*, 20(2):99, 2018.

[22] Michal Daszykowski, Beata Walczak, and DL Massart. Looking for natural patterns in data: Part 1. density-based approach. *Chemometrics and Intelligent laboratory systems*, 56(2):83–92, 2001.

[23] Joost CF De Winter, Samuel D Gosling, and Jeff Potter. Comparing the pearson and spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data. *Psychological methods*, 21(3):273, 2016.

[24] Yao Deng, Xi Zheng, Tianyi Zhang, Chen Chen, Guannan Lou, and Miryung Kim. An analysis of adversarial attacks and defenses on autonomous driving models. In *2020 IEEE international conference on pervasive computing and communications (PerCom)*, pages 1–10. IEEE, 2020.

[25] Yizhen Dong, Peixin Zhang, Jingyi Wang, Shuang Liu, Jun Sun, Jianye Hao, Xinyu Wang, Li Wang, Jin Song Dong, and Dai Ting. There is limited correlation between coverage and robustness for deep neural networks. *arXiv preprint arXiv:1911.05904*, 2019.

[26] Yizhen Dong, Peixin Zhang, Jingyi Wang, Shuang Liu, Jun Sun, Jianye Hao, Xinyu Wang, Li Wang, Jinsong Dong, and Ting 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)*, pages 73–82. IEEE, 2020.

[27] Hamid Ebadi, Mahshid Helali Moghadam, Markus Borg, Gregory Gay, Afonso Fontes, and Kasper Socha. Efficient and effective generation of test cases for pedestrian detection-search-based software testing of baidu apollo in svl. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, pages 103–110. IEEE, 2021.

[28] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4):551–559, 1983.

[29] Robert Feldt, Simon Poulding, David Clark, and Shin Yoo. Test set diameter: Quantifying the diversity of sets of test cases. In *2016 IEEE international conference on software testing, verification and validation (ICST)*, pages 223–233. IEEE, 2016.

[30] Robert Feldt, Richard Torkar, Tony Gorschek, and Wasif Afzal. Searching for cognitively diverse tests: Towards universal test diversity metrics. In *2008 IEEE International Conference on Software Testing Verification and Validation Workshop*, pages 178–186. IEEE, 2008.

[31] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu 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*, pages 177–188, 2020.

[32] Alessio Gambi, Marc Mueller, and Gordon Fraser. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 318–328, 2019.

[33] Alessio Gambi, Marc Müller, and Gordon Fraser. Asfault: Testing self-driving car software using search-based procedural content generation. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 27–30. IEEE, 2019.

[34] Simos Gerasimou, Hasan Ferit Eniser, Alper Sen, and Alper Cakan. Importance-driven deep learning system testing. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 702–713, 2020.

[35] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. Dlfuzz: Differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 739–743, 2018.

[36] Florian Hauer, Tabea Schmidt, Bernd Holzmüller, and Alexander Pretschner. Did we test all scenarios for automated and autonomous driving systems? In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2950–2955. IEEE, 2019.

[37] Christopher Henard, Mike Papadakis, Mark Harman, Yue Jia, and Yves Le Traon. Comparing white-box and black-box test prioritization. In *Proceedings of the 38th International Conference on Software Engineering*, pages 523–534, 2016.

[38] Bernhard Kaiser. Application story of odd as part of safety assurance. https://www.asam.net/index.php?eID=dumpFile&t=f&f=4303&token=3135965e578e5bb92a01725cd37823c3979da158, 2021.

[39] Baekgyu Kim, Takato Masuda, and Shinichi Shiraishi. Test specification and generation for connected and autonomous vehicle in virtual environments. *ACM Transactions on Cyber-Physical Systems*, 4(1):1–26, 2019.

[40] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019.

[41] Steven R Kirk and Samantha Jenkins. Information theory-based software metrics and obfuscation. *Journal of Systems and Software*, 72(2):179–186, 2004.

[42] András Kocsor, Attila Kertész-Farkas, László Kaján, and Sándor Pongor. Application of compression-based distance measures to protein sequence classification: a methodological study. *Bioinformatics*, 22(4):407–412, 2006.

[43] Andrei N Kolmogorov. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1(1):1–7, 1965.

[44] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[45] Thomas Laurent, Stefan Klikovits, Paolo Arcaini, Fuyuki Ishikawa, and Anthony Ventresque. Parameter coverage for testing of autonomous driving systems under uncertainty. *ACM Transactions on Software Engineering and Methodology*, 2022.

[46] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 25–36. IEEE, 2020.

[47] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. The similarity metric. *IEEE transactions on Information Theory*, 50(12):3250–3264, 2004.

[48] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun 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)*, pages 89–92. IEEE, 2019.

[49] Chengjie Lu, Yize Shi, Huihui Zhang, Man Zhang, Tiexin Wang, Tao Yue, and Shaukat Ali. Learning configurations of operating environment of autonomous vehicles to maximize their collisions. *IEEE Transactions on Software Engineering*, 2022.

[50] Chengjie Lu, Huihui Zhang, Tao Yue, and Shaukat Ali. Search-based selection and prioritization of test scenarios for autonomous driving systems. In *International Symposium on Search Based Software Engineering*, pages 41–55. Springer, 2021.

[51] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, and Yang Liu. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*, pages 120–131, 2018.

[52] Anne E Magurran. Measuring biological diversity. *Current Biology*, 31(19):R1174–R1177, 2021.

[53] István Majzik, Oszkár Semeráth, Csaba Hajdu, Kristóf Marussy, Zoltán Szatmári, Zoltán Micskei, András Vörös, Aren A Babikian, and Dániel Varró. Towards system-level testing with coverage guarantees for autonomous vehicles. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 89–94. IEEE, 2019.

[54] Christopher D Manning. *Introduction to information retrieval*. Syngress Publishing,, 2008.

[55] Silverio Martínez-Fernández, Justus Bogner, Xavier Franch, Marc Oriol, Julien Siebert, Adam Trendowicz, Anna Maria Vollmer, and Stefan Wagner. Software engineering for ai-based systems: a survey. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(2):1–59, 2022.

[56] Mahshid Helali Moghadam, Markus Borg, Mehrdad Saadatmand, Seyed Jalaleddin Mousavirad, Markus Bohlin, and Björn Lisper. Machine learning testing in an adas case study using simulation-integrated bio-inspired search-based testing. *arXiv preprint arXiv:2203.12026*, 2022.

[57] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[58] Wafa Mousser and Salima Ouadfel. Deep feature extraction for pap-smear image classification: A comparative study. In *Proceedings of the 2019 5th International Conference on Computer and Technology Applications*, pages 6–10, 2019.

[59] Mario A Muñoz and Kate A Smith-Miles. Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evolutionary computation*, 25(4):529–554, 2017.

[60] Mario A Muñoz, Laura Villanova, Davaatseren Baatar, and Kate Smith-Miles. Instance spaces for machine learning classification. *Machine Learning*, 107(1):109–147, 2018.

[61] Mario A Muñoz, Tao Yan, Matheus R Leal, Kate A Smith-Miles, Ana C Lorena, Gisele L Pappa, and Rodrigues Romulo M. An instance space analysis of regression problems. *ACM Transactions on Knowledge Discovery from Data*.

[62] Neelofar, Kate Smith-Miles, Mario Andrés Muñoz, and Aldeida Aleti. Instance space analysis of search-based software testing. *IEEE Transactions on Software Engineering*, 2022.

[63] Borislav Nikolik. Test diversity. *Information and Software Technology*, 48(11):1083–1094, 2006.

[64] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[65] Tuan Pham, Rob Hess, Crystal Ju, Eugene Zhang, and Ronald Metoyer. Visualization of diversity in large multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1053–1062, 2010.

[66] Andrea Piazzoni, Jim Cherian, Mohamed Azhar, Jing Yew Yap, James Lee Wei Shung, and Roshan Vijay. Vista: a framework for virtual scenario-based testing of autonomous vehicles. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, pages 143–150. IEEE, 2021.

[67] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering*, 25:5193–5254, 2020.

[68] Vincenzo Riccio and Paolo Tonella. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 876–888, 2020.

[69] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, and Shalin Mehta. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, pages 1–6. IEEE, 2020.

[70] C Saranya and G Manikandan. A study on normalization techniques for privacy preserving data mining. *International Journal of Engineering and Technology (IJET)*, 5(3):2701–2704, 2013.

[71] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

[72] Edward Schwalb, Patrick Irvine, Xizhe Zhang, Siddartha Khastgir, and Paul Jennings. A two-level abstraction odd definition language: Part ii. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1669–1676. IEEE, 2021.

[73] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[74] Andrew Smart, Chess Stetson, and Kiran Jesudesan. Autonomous vehicle safety assessment with fully quantified odds. *SAE International Journal of Advances and Current Practices in Mobility*, 4(2021-01-1011):270–277, 2021.

[75] Andrea Stocco, Paulo J Nunes, Marcelo D'Amorim, and Paolo Tonella. Thirdeye: Attention maps for safe autonomous driving systems. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022.

[76] Andrea Stocco, Brian Pulfer, and Paolo Tonella. Model vs system level testing of autonomous driving systems: a replication and extension study. *Empirical Software Engineering*, 28(3):73, 2023.

[77] Weidi Sun, Yuteng Lu, and Meng Sun. Are coverage criteria meaningful metrics for dnns? In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[78] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Structural test coverage criteria for deep neural networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–23, 2019.

[79] Yun Tang, Yuan Zhou, Yang Liu, Jun Sun, and Gang Wang. Collision avoidance testing for autonomous driving systems on complete maps. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 179–185. IEEE, 2021.

[80] Haoxiang Tian, Yan Jiang, Guoquan Wu, Jiren Yan, Jun Wei, Wei Chen, Shuo Li, and Dan Ye. Mosat: finding safety violations of autonomous driving systems using multi-objective genetic algorithm. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 94–106, 2022.

[81] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[82] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 146–157, 2019.

[83] Xinchen Ye and Xuesong Wang. Operational design domain of automated vehicles at freeway entrance terminals. *Accident Analysis & Prevention*, 174:106776, 2022.

[84] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 48(1):1–36, 2020.

[85] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 132–142, 2018.

[86] Peixin Zhang, Jingyi Wang, Jun Sun, and Xinyu Wang. Fairness testing of deep image classification with adequacy metrics. *arXiv preprint arXiv:2111.08856*, 2021.

[87] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo 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*, pages 79–90, 2021.