Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Beyond Coverage: An Empirical Study of Mutation-Score Proxies for Deep Neural Network Testing

Vivek V. Vekariya

vivek.vekariya@tum.de

Technical University of Munich

Mojdeh Golagha

Infineon Technologies (Germany)

Andrea Stocco

Technical University of Munich

Alexander Pretschner

Technical University of Munich

Research Article

Keywords: Deep Neural Network Testing, Mutation Testing, Test Adequacy Metrics, Coverage-Guided Fuzzing, Mutation Score Approximation

Posted Date: September 2nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10524158/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: Competing interest reported. Alexander Pretschner serves as an Editorial Board Member of the Software Quality Journal. In accordance with the journal's editorial policies, he will not be involved in the editorial handling, reviewer selection, or decision-making process for this manuscript. The remaining authors declare that they have no competing interests.

Beyond Coverage: An Empirical Study of Mutation-Score Proxies for Deep Neural Network Testing

Vivek V. Vekariya[1*], Mojdeh Golagha[4], Andrea Stocco[2,3], Alexander Pretschner[1,3]

[1*]Chair of Software and Systems Engineering, Technical University of Munich, Munich, Bayern, Germany.
[2]Chair of Software Engineering for Data-intensive Applications, Technical University of Munich, Munich, Bayern, Germany.
[3]fortiss GmbH, Munich, Bayern, Germany.
[4]Infineon Technologies AG, Munich, Bayern, Germany.

*Corresponding author(s). E-mail(s): vivek.vekariya@tum.de;
Contributing authors: mojdeh.golagha@infineon.com;
andrea.stocco@tum.de, stocco@fortiss.org;
alexander.pretschner@tum.de, pretschner@fortiss.org;

Abstract

High-quality test datasets are essential for evaluating the reliability of Deep Neural Networks (DNNs). Traditional test adequacy metrics lack empirical evidence that datasets evaluated using them can also expose diverse DNN faults. Mutation testing provides a fault-oriented perspective on dataset quality by assessing its ability to expose injected faults, as measured by the mutation score (MS). However, its high computational cost limits its practical adoption, motivating the search for computationally efficient proxies.

In this paper, we first systematically evaluate the correlation between MS and existing test adequacy criteria using 233 mutants spanning 15 DNN fault categories across four well-established computer vision benchmarks. Our results show that structural adequacy metrics exhibit no meaningful correlation with MS. Non-structural metrics, such as Likelihood and Distance-based Surprise Coverage, show only weak positive correlations, indicating their limited value as MS proxies.

We then compare covariance-aware latent-space deviation, represented by Mahalanobis Distance-based Surprise Coverage (MDSC), with direct latent-space

dispersion, operationalized through Latent Space Class Dispersion (LSCD). Although both metrics achieve comparable computational efficiency, they differ in how they characterize class distributions and in their relationship with MS. MDSC achieves a mean Pearson correlation of 0.505 and a mean Spearman rank correlation of 0.329 with MS. In comparison, LSCD achieves a mean Pearson correlation of 0.844 and a mean Spearman correlation of 0.507, indicating that direct latent-space dispersion exhibits a stronger relationship with MS under the studied settings. These empirical insights indicate that while both methods offer scalable approximations to mutation testing, LSCD shows a stronger correlation with the mutation score.

Keywords: Deep Neural Network Testing, Mutation Testing, Test Adequacy Metrics, Coverage-Guided Fuzzing, Mutation Score Approximation

1 Introduction

Deep Neural Networks (DNNs) are widely used for safety-relevant tasks, such as Automated Driving (AD) (Bojarski et al. 2016), medical diagnosis (Zhang et al. 2018), disease prediction (Zhao et al. 2019), or aircraft collision avoidance (Julian et al. 2018). DNNs are trained to process complex input data (e.g., images) from large datasets, and their deployment in the real world depends on their generalization capabilities to correctly process unseen data. To this end, engineers use evaluation metrics such as accuracy, precision, recall, or mean squared error on test datasets unknown at training time (Zhang et al. 2022; Riccio et al. 2020). Despite researchers' efforts to ensure the reliability and dependability of DNNs (Li et al. 2022; Kaminwar et al. 2021; Zhang et al. 2022; Riccio et al. 2020), *the quality of test datasets* remains a crucial factor in assessing their performance before deployment in real-world settings. This is because test datasets represent only a subset of the targeted operational domain, and they may fail to capture relevant properties of in-field inputs.

Ideally, high-quality test datasets contain samples similar to the training set distribution and samples that represent rare situations occurring in the real world (i.e., corner cases). Ensuring the adequacy and quality of test suites for DNNs is a crucial endeavor in software engineering (Li et al. 2022; Kaminwar et al. 2021; Zhang et al. 2022; Riccio et al. 2020). In the literature, this problem is mainly tackled in two ways: by defining test adequacy criteria for DNNs and by developing mutation testing techniques.

Regarding the adequacy criteria, various forms of neuron coverage have been introduced (Yuan et al. 2021; Pei et al. 2019), often referred to as structural adequacy criteria, as they are based on a DNN's architecture. On the other hand, non-structural adequacy criteria, such as surprise-based coverage, rely on latent-space representations to quantify how much test data deviates from the training distribution (Kim et al. 2019, 2023; Weiss et al. 2021; Yan et al. 2024). They provide a practical way to characterize the test data without requiring full access to the DNN architecture.

In contrast, the Mutation Score (MS) (Tufano et al. 2020; Hu et al. 2019; Humbatova et al. 2021; Zhang et al. 2025) quantifies the quality of test datasets by their

ability to reveal various faults in DNNs. To calculate MS, pre-training mutation operators (MOs) that replicate actual DNN faults (Humbatova et al. 2020) are used to generate mutant models (Tufano et al. 2020; Humbatova et al. 2021); post-training mutation operators generally do not reflect real DNN faults (Humbatova et al. 2021). A test dataset with a higher MS is considered more fault-revealing under the studied mutation setting and is a good representative of the targeted operational domain, effectively identifying various faults and assessing the generalizability of DNNs.

Both approaches, while popular, have shortcomings, particularly in terms of computational efficiency, which limits their scalability in practical applications. For instance, pre-training MOs requires re-training the original DNNs, making them computationally expensive and sensitive to the hyperparameters used (Humbatova et al. 2021; Zhang et al. 2025). Conversely, traditional structural test adequacy criteria, such as Neuron Coverage (NC), have proven inadequate (Harel-Canada et al. 2020) in many cases. The problem with structural coverage is that a small subset of the test data is often enough to achieve a high score, and that its computational complexity grows with the size of the DNN under test. Likelihood and Distance-based Surprise Coverage (LSC and DSC) are popular means to quantify test dataset quality as they measure surprising behaviors for a given input based on the latent space of a DNN (Kim et al. 2019). While promising, their effectiveness under diverse DNN fault conditions remains largely unexplored.

These observations motivate a systematic investigation of how different test adequacy metrics correlate to the mutation score. In our work, we evaluate seven well-studied test adequacy criteria (four structural and three non-structural) against MS. Our analysis is conducted using four established computer vision benchmark datasets-MNIST (Deng 2012), SVHN (Netzer et al. 2011), GTSRB (Stallkamp et al. 2011), and CIFAR-100 (Krizhevsky 2009)-and a total of 233 mutant models encompassing 15 unique DNN fault categories. Furthermore, we extend our analysis to include automatically generated corner cases from Coverage-Guided Fuzzing (CGF). Corner cases are rare in-field inputs that reveal unexpected behaviors in DNNs. In this study, corner cases refer to fuzzing-generated inputs that are misclassified by the DNN under test. This helps assess the effectiveness of each metric under rare, challenging conditions. Using these evaluation settings across diverse datasets, mutant types, and challenging inputs, our study evaluates their viability as practical and scalable surrogates for MS. The goal of this study is to assess whether existing test adequacy metrics approximate mutation score under the evaluated conditions, rather than to replace mutation-based evaluation.

Our analysis shows that traditional structural adequacy metrics exhibit no meaningful correlation with MS and fail to distinguish between test data from the same distribution as the training data and corner case data. Non-structural metrics based on surprise coverage show only weak positive correlations with MS (mean Pearson correlations of 0.279 and 0.264 for LSC and DSC, respectively) across all the datasets we studied.

Motivated by these findings, we further investigate alternative latent space-based metrics to assess dataset quality. In particular, we compare covariance-aware latent-space distributions, represented by Mahalanobis Distance-based Surprise Coverage

(MDSC) (Kim et al. 2023), with direct latent-space dispersion, operationalized through Latent Space Class Dispersion (LSCD). MDSC characterizes how test inputs deviate from class distributions after covariance-based normalization (Kim et al. 2023), whereas LSCD measures how broadly test inputs are dispersed around class centroids derived using training data. Consequently, these two approaches characterize complementary geometric properties of the latent space: while covariance-based methods track the orientation and shape of class regions, LSCD reflects their overall internal dispersion. From a computational perspective, both approaches can be implemented with comparable runtime complexity. However, MDSC additionally requires estimating and storing class-wise covariance matrices, whereas LSCD relies only on class centroids, making it simpler to compute and more lightweight in practice. We treat MDSC and LSCD as complementary latent-space perspectives and empirically study which property is more closely associated with the mutation score under the studied settings.

Our empirical analysis shows that LSCD exhibits a consistently strong relationship with mutation score under the studied conditions, achieving a mean Pearson correlation of 0.844 across all datasets. To further assess the reliability of the relationship between adequacy metrics and MS, we also examine rank-based consistency using Spearman's correlation. While some metrics show moderate linear correlation, they fail to preserve consistent dataset ordering. For example, MDSC achieves a Pearson correlation of 0.505 but a lower Spearman coefficient of 0.329, indicating variability in ranking behavior. In contrast, LSCD maintains both higher Pearson (0.844) and Spearman (0.507) correlations, suggesting a more stable approximation of mutation score across datasets.

We also observe that both LSCD and MDSC retain favorable computational efficiency. While their runtime is comparable to other latent-space metrics such as MDSC, it avoids covariance-based modeling, making it easier to apply in practice. In comparison to other structural and density-based approaches, LSCD is on average 10.4 times faster than neuron coverage (the most efficient structural criterion) and up to 78.8 times faster than likelihood-based surprise coverage. These results indicate that LSCD provides a computationally efficient approximation of mutation-based evaluation, potentially reducing the need for expensive mutant generation when assessing test dataset quality.

In summary, this paper makes the following contributions:

- We conduct an empirical study of DNN test adequacy metrics as approximations of mutation score, utilizing 233 mutant models from 15 unique DNN fault categories across four diverse computer vision benchmarks.
- We show that structural test adequacy metrics and widely used surprise-based metrics exhibit weak or inconsistent relationships with mutation score under the studied settings.
- We identify direct latent-space dispersion as an additional useful empirical signal for mutation-score approximation and operationalize it through LSCD.
- We complement the correlation study with corner-case validity analysis and runtime evaluations. All results and artifacts are available in our replication package (2026).

2 Background

This section provides a brief overview of traditional structural and non-structural test adequacy metrics, followed by an introduction to mutation scores. We also discuss our motivation for why approximating mutation scores is necessary, and how they can provide a valuable and practical solution for assessing the quality of DNN test datasets.

2.1 Structural Test Adequacy Metrics

Pei et al. first proposed structural test adequacy metrics, particularly neuron coverage, in 2017 (Pei et al. 2019). Neuron coverage is inspired by line coverage in traditional software testing. The initial goal was to create a quantifiable, white-box method for assessing the quality of a test dataset for a DNN. Over time, various forms of NC were introduced by Ma et al., such as k-MultiSection Neuron Coverage (k-MNC), Neuron Boundary Coverage (NBC), etc (Ma et al. 2018).

While a neuron's activation can theoretically range from $-\infty$ to $+\infty$, during training its values typically stay within a specific *low* and *high* bound determined by the training data. The various coverage criteria leverage this idea differently: NC checks whether a neuron's activation surpasses a fixed threshold t. k-MNC divides the observed activation range $[low, high]$ into k sections, and NBC focuses on the ranges from $[-\infty, low)$ and $(high, +\infty]$. Consequently, a test dataset is considered high-quality if it achieves high activation across these criteria, as this indicates that it has successfully pushed the DNN to demonstrate a wider range of its potential behaviors.

The underlying motivation for all these structural adequacy criteria is to have a white-box methodology. This approach allows test engineers to gain a deeper understanding of the internal workings of a DNN and to quantify test dataset quality. Also, these metrics serve as a feedback mechanism to automatically generate corner case data that can reveal faults in the DNN under test (Xie et al. 2019; Tian et al. 2018). However, their reliance on complete access to the DNN's internal structure and activation values is a significant drawback when using them in practice. Additionally, their computational complexity increases with the size and depth of the network, making them less scalable for very large DNNs.

2.2 Non-Structural Test Adequacy Metrics

Unlike structural metrics, non-structural test adequacy metrics do not rely on explicit architectural elements of a DNN (e.g., individual neurons or layers). Instead, they utilize latent space vectors, treating the DNN under test as a feature extractor rather than analyzing its internal structure directly. This makes them more flexible and easier to apply in practice across different architectures. Furthermore, using the DNN's latent space is a crucial testing technique as it provides a systematic way to understand the network's decision-making process. By measuring how unexpected a new test input is relative to what the model has learned, these metrics can reveal important insights into DNN behavior. Before diving into non-structural test adequacy metrics, we will first explain the notion of latent space and its importance for testing.

2.2.1 Latent Space of a DNN

During training, the DNN learns complex feature representations from the input data and encodes them into a *latent space*, i.e., a low-dimensional representation embedded within the layers of a DNN (Goodfellow et al. 2016). In typical convolutional/feed-forward DNNs, each layer of the network transforms the input data into progressively more abstract and high-level representations; the final hidden layer before the output layer captures the most essential features of the input data, which are used for predictions (Goodfellow et al. 2016). The output layers for classification DNNs are usually softmax layers.

Previous works have shown the importance of the last hidden layers (also referred to as logit layers) in a DNN (Bengio et al. 2012; Kim et al. 2019, 2023). Therefore, consistent with these findings, in our work, we also consider the last hidden layers as the latent space where the learning of the DNN is summarized for decision-making. Given an input, the latent space embeds all the information that the DNN uses for inference. In other words, it encapsulates the maximum possible behaviors of the DNN. Therefore, a profound understanding of the latent space and its properties is essential for DNN testing and debugging.

2.2.2 Leveraging the Latent Space for DNN Test Adequacy

The DNN learns non-linear decision boundaries to separate input data distributions based on their characteristics. Ideally, a high-performing DNN has class regions that are well-separated from one another by non-linear classification/decision boundaries in the latent space. Consequently, we distinguish between latent space vectors corresponding to the input test data located:

1. *Near the training data distribution:* When a latent space vector corresponding to a given test input is located far away from the decision boundary, near the latent space vectors corresponding to training data, the DNN clearly identifies features from the input, and it correctly classifies the input with high confidence (Wong et al. 2020).
2. *In the proximity of the decision boundary:* These latent space vectors can be easily manipulated by adding some noise (e.g., adversarial attacks (Huang et al. 2017), or image corruptions (Hendrycks and Dietterich 2019; Lambertenghi et al.)) to cause incorrect classification.
3. *In sparse regions away from training distribution:* When a latent space vector is located in sparsely populated regions, away from training latent space vectors, the DNN fails to clearly identify the features from the input for reliable prediction. This could be due to an input with complex features not represented by similar examples during training, but it could still occur in the real world.

Test datasets mapped to the sparse region and near the decision boundaries have a higher potential to assess the generalizability of a DNN. We refer to these regions outside the original training data distribution in the latent space, which is not tested with the currently available test dataset as the *untested regions of the latent space*. We consider it promising to define an evaluation metric for DNNs that is informed by

these untested regions of the latent space, and to seek to find more test data from this region. By doing so, we provide test engineers with a metric reflective of the DNN's intrinsic learning capability while quantifying the test dataset quality.

2.2.3 Likelihood-based Surprise Coverage

Kim et al. (Kim et al. 2019) introduced the Surprise Coverage (SC) criterion, which is designed to capture the *"surprise"* behavior of test data that deviates from its training data, as observed by a DNN. Test data with complex features may be surprising as a DNN struggles to identify features for classification, making them valuable for testing. Likelihood-based surprise coverage quantifies the quality of the test dataset by measuring its likelihood under a probabilistic model derived using Kernel Density Estimation (KDE) (Wand and Jones 1994), based on the training dataset. The basic idea is that a new input with a very low likelihood is considered highly surprising and is therefore more likely to reveal unique or problematic behaviors in the DNN under test.

2.2.4 Distance-based Surprise Coverage

Unlike LSC, distance-based surprise coverage utilizes latent space vectors to quantify the quality of the test dataset. DSC considers a test input surprising if its latent space vector is located away from the training distribution, specifically, near the decision boundary (as discussed in (Section 2.2.2)). High-quality test datasets with complex features will result in higher values of DSC for reliable assessment of DNN's performance for real-world applications. (While Kim et al. (Kim et al. 2019) used the term "Activation Trace (AT)", it is synonymous to "Latent Space Vector", the more widely used term in the literature. Thus, we will discuss ATs as latent space vectors in this paper.)

2.2.5 Mahalanobis Distance-based Surprise Coverage

Mahalanobis Distance-based Surprise Coverage (MDSC) extends distance-based surprise coverage by incorporating the statistical properties of class-wise latent representations. Unlike DSC, which relies on Euclidean distances to nearest neighbors, MDSC measures the distance of a test input from the distribution of training data by combining class centroids with covariance information. Intuitively, MDSC considers a test input to be surprising if its latent representation deviates from the distribution of training inputs, rather than from individual samples. It is based on Mahalanobis distance, which measures the distance between a point and a distribution using both the mean and covariance of the training data (Kim et al. 2023). As discussed in the original work, this formulation was introduced to provide a more direct and general measure of surprise compared to distance-based approaches, which rely on relative distances to specific training samples and are primarily suited for classification tasks (Kim et al. 2023).

While MDSC offers a flexible way to characterize deviations from the training distribution, it requires estimating and storing class-wise covariance matrices in addition

to class centroids. This introduces additional storage requirements and a precomputation step, although distance computation for new inputs remains efficient once these statistics are available (Kim et al. 2023). In (Section 3.3.4), we use an illustrative example to compare DSC, MDSC, and LSCD, highlighting how they capture different properties of latent-space representations.

2.3 Mutation Score

Unlike traditional test adequacy metrics, the mutation score characterizes the quality of test datasets based on their ability to capture changes in mutant DNNs. The hypothesis is that high-quality test data should maintain its effectiveness under various fault conditions. These arise from hidden defects like inadequately chosen hyperparameters, which can lead to unstable or suboptimal behavior of a DNN. To this end, various mutation operators broadly categorized as pre-training and post-training MOs, have been suggested to generate mutant DNNs. The mutation score measures discrepancies between the predictions of a mutant model and the original model on the test dataset. A higher MS indicates good quality test data as it reflects changes in the DL system under test. However, to compute an MS, a test engineer must first generate mutant models. While various post-training MOs have been proposed, their effectiveness in representing and reflecting real faults in DNNs has been questioned (Humbatova et al. 2021; Zhang et al. 2025; Panichella and Liem 2021). On the other hand, pre-training MOs are computationally expensive as they require re-training the DNN. Despite their computational cost, pre-training MOs are valuable in mutation testing due to a strong link to the real faults of DNNs (Humbatova et al. 2021; Zhang et al. 2025). Thus, we study pre-training MOs derived from real faults of DNNs in this paper and follow the mutation testing methodology from existing works (Tufano et al. 2020; Hu et al. 2019; Humbatova et al. 2021) for computing mutation scores.

2.3.1 Why Seek Scalable Approximations to Mutation Scores?

As discussed, the mutation score provides a fault-oriented perspective on test dataset quality. However, the computational time required to compute the mutation score will grow with the increased size and complexity of DNNs, making this approach impractical for emerging application domains (Zhang et al. 2022). This motivates the search for computationally efficient metrics that approximate the mutation score while retaining its usefulness for test dataset quality assessment.

We argue that non-structural test adequacy metrics are a more promising and scalable approximation to the mutation scores, as long as they are efficient to compute. This is because they eliminate the need to generate mutant models and do not require access to the DNN's internal architecture, unlike structural metrics. To validate this, we assess whether higher values of these metrics correlate with mutation score and compare their effectiveness against structural metrics. To the best of our knowledge, such correlations have not been systematically studied. In this paper, we address this gap by empirically analyzing these relationships and identifying efficient metrics that can serve as effective proxies for mutation score.

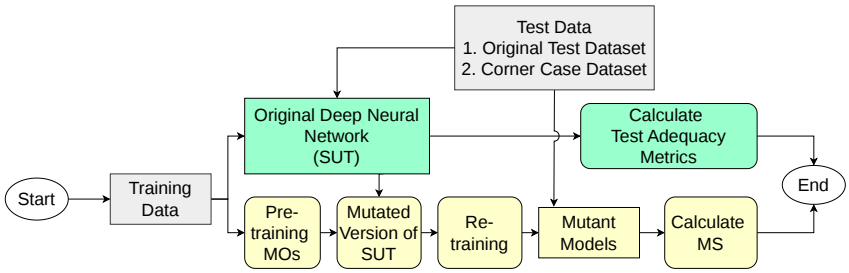


Fig. 1: Our framework to evaluate test dataset quality based on traditional test adequacy metrics (pipeline highlighted in green blocks). We compare our approach with the Mutation Score (highlighted in the pipeline with yellow colored blocks).

3 Methodology

In this section, we outline our methodology to perform test dataset quality assessment using traditional test adequacy metrics and mutation scores proposed for DNN-based systems (Tufano et al. 2020; Hu et al. 2019; Humbatova et al. 2021). We then describe the improvement in test dataset quality when using coverage-guided fuzzing, as measured by the same metrics. We selected fuzzing over adversarial samples to improve the test dataset quality because adversarial samples are typically engineered to target specific model vulnerabilities, whereas fuzzing provides a broader and more scalable method for revealing diverse and unforeseen behaviors in DNNs more aligned with real-world scenarios. We also compute the Pearson and Spearman rank correlation coefficients between these metrics and the mutation score. Lastly, we operationalize direct latent-space dispersion through LSCD and compare it with MDSC and DSC (its closest counterparts).

3.1 Correlation Analysis of Test Adequacy Metrics with Mutation Score

Following the insights presented so far, a high-quality test dataset, as indicated by its higher value of test adequacy metrics, has a higher potential to verify the generalization capabilities of DNNs. Similar to these metrics, a high MS also indicates a good quality test dataset that helps assess the generalization and fault detection capabilities of DNNs. Figure 1 overviews the individual steps, which include test dataset quality evaluation using test adequacy metrics and mutation scores. In our empirical study, we utilized four distinct structural metrics, NC, k-MNC, NBC, and Strong Neuron Activation Coverage (SNAC), which cover neurons' activation range from $[-\infty, +\infty]$ at various granularity levels. We also utilized three state-of-the-art non-structural metrics, LSC, DSC, and MDSC.

We first calculate standard evaluation metrics such as accuracy and test dataset quality using the test adequacy metrics described above, and also measure the computation time required to compute each of the test adequacy metrics. Note that high accuracy alone might create a false sense of trustworthiness in a DNN's performance.

This is because when a test dataset yields low scores on structural metrics (like neuron coverage) and low surprise on latent space metrics, this indicates a lack of diversity in the present test dataset. Low structural scores suggest the test data fails to activate a wide range of the DNN's neurons, while low surprise scores suggest that the DNN effectively identifies features in the current test dataset similar to those in the training set. This gives rise to the possibility that the DNN under test may not generalize well to real-world data with more complex features.

To verify if this is the case, we evaluate the quality of corner cases identified using fuzzing. If these generated corner cases have both higher structural coverage (e.g., activating previously unactivated neurons/activation ranges) and higher latent space metric scores (e.g., LSC/DSC/MDSC) than the original test dataset, then this indicates that the DNN finds them "surprising" and where correct classification is more challenging. These corner case data can enhance the quality of the original test dataset as they can be mapped to the sparse and near decision boundary regions in the latent space. By expanding coverage beyond the training data distribution, they enable more effective evaluation of the model's generalization behavior in real-world scenarios.

To evaluate the effectiveness of test adequacy metrics under various fault conditions and select the most suitable criteria, we then compute their correlation with MS on the mutant DNNs obtained from pre-training MOs using both original test data and corner case data. A strong positive correlation would mean that test engineers can use a metric with the highest correlation as a computationally efficient approximation to the mutation score for evaluating test data quality, thereby eliminating the need to generate mutant DNNs. Post-training MOs tend to be unrealistic as they directly change the trained model, which is unlikely to happen in real-world settings (Humbatova et al. 2021; Abbasishahkoo et al. 2024). Also, they lack precise mapping to actual DNN faults (Humbatova et al. 2021, 2020; Abbasishahkoo et al. 2024). Thus, we focus on pre-training MOs that mimic real faults in DNNs.

3.2 Structural Test Adequacy Metrics: Definitions and Calculations

As briefly introduced in Section 2.1, structural test adequacy metrics are white-box criteria that require full access to a DNN's architecture and activation values. Neuron coverage represents the most simplistic form, which is calculated using Equation 1 (Pei et al. 2019). NC measures the total number of activated neurons n above threshold t using the test dataset T. In Equation 1, $a(n, x)$ is the activation value of neuron n using input x and N is the set of all neurons.

$$NC = \frac{|\{n|\exists x \in T : a(n, x) > t\}|}{|N|} \tag{1}$$

k-Multi-section neuron coverage is a fine-grained version of NC that divides the observed activation range of a neuron from $[low, high]$ into k sections. Using the training dataset, these bounds are derived based on the observed lowest and highest activation values. Then, k-MNC measures the percentage of these sections s covered

by the test suite T from the total S_n sections of neuron n, as shown in Equation 2 (Ma et al. 2018).

$$k - MNC = \frac{\sum_{n \in N} |\{s \in S_n | \exists x \in T : a(n,x) \in s\}|}{k \times |N|} \tag{2}$$

Neuron boundary coverage and strong neuron activation coverage specifically target a neuron's activation range outside $[low, high]$ (Ma et al. 2018). Instead of focusing on the activation values observed during training, they measure whether a test suite can push a neuron's activation beyond these known bounds. N_{upper_bound} and N_{lower_bound} in Equation 3, describes neuron's activation ranges from $(high, +\infty)$ and $(-\infty, low)$ respectively. NBC and SNAC are calculated using Equation 4. NBC considers the total number of both the lower and upper bounds activated out of the total bounds N as described in Equation 3. On the other hand, SNAC focuses only on the upper bound, which is often associated with "strong" activations. It can be observed that the computational complexity of these coverage criteria grows with the size of the DNN under test and the level of granularity required for the activation ranges.

$$\begin{aligned} N_{upper_bound} &= \{n \in N \mid \exists x \in T : a(n,x) \in (\text{high}, +\infty)\} \\ N_{lower_bound} &= \{n \in N \mid \exists x \in T : a(n,x) \in (-\infty, \text{low})\} \end{aligned} \tag{3}$$

$$\begin{aligned} NBC &= \frac{|N_{lower_bound} \cup N_{upper_bound}|}{2 * |N|} \\ SNAC &= \frac{|N_{upper_bound}|}{|N|} \end{aligned} \tag{4}$$

3.3 Non-Structural Test Adequacy Metrics: Definitions and Calculations

Unlike structural methods, non-structural test adequacy metrics can be applied in gray- or black-box settings, as they do not require explicit access to the architectural elements of a DNN (e.g., individual neurons). They instead rely on latent space vectors to assess test dataset quality. The following section describes widely studied non-structural coverage criteria.

3.3.1 Likelihood-based Surprise Coverage

Likelihood-based surprise coverage is calculated in multiple steps using Equation 5 and Equation 6. First, a density function $\hat{f}(.)$ is estimated using Kernel Density Estimation using the latent space vectors of training data T_{train} (Wand and Jones 1994; Kim et al. 2023). In Equation 5, K_H refers to the Gaussian kernel function. So, for a given test input x, first the density is estimated using Equation 5 for class i. Here, x_i denotes training samples belonging to class i. Later, LSC for given test input x is calculated as the negative logarithm of the estimated density function using Equation 6.

This converts the probability density into a surprise score, where inputs with a low probability density (i.e., those far from the training data) result in a high surprise score. Essentially, the lower the density of a data point in the latent space, the more "surprising" it is considered to be. For the entire test dataset X, LSC is calculated using Equation 13.

$$\hat{f}(x) = \frac{1}{|a(T_{train})|} \sum_{x_i \in T_{train}} K_H(a(n,x) - a(n,x_i)) \tag{5}$$

$$LSC(x) = -\log(\hat{f}(x)) \tag{6}$$

3.3.2 Distance-based Surprise Coverage

Distance-based surprise coverage is also calculated in multiple steps (Kim et al. 2019). Firstly, a new test input x is mapped to the nearest neighbor x_a in its predicted class c_x using Equation 7. Here, x_i denotes training samples and $D(x_i)$ represents the predicted class of x_i. Then, the Euclidean distance between the latent space vector of x and its nearest found neighbor x_a is calculated using Equation 8. Here, $\overrightarrow{\alpha_N}(.)$ refers to the latent space vector for a given input.

$$x_a = \arg \min_{D(x_i)=c_x} \|\overrightarrow{\alpha_N}(x) - \overrightarrow{\alpha_N}(x_i)\| \tag{7}$$

$$\text{dist}_a = \|\overrightarrow{\alpha_N}(x) - \overrightarrow{\alpha_N}(x_a)\| \tag{8}$$

Second, the nearest neighbour x_b for x is found from other output classes $C \setminus \{c_x\}$ using Equation 9. Using Equation 10, the Euclidean distance between the latent space vector of x_a and x_b is calculated. Here, C is the total number of output classes. Finally, DSC is computed as a ratio of distances calculated from the predicted class to the nearest input in another class using Equation 11. A higher DSC value indicates that a test input is more surprising, as its latent representation lies farther from its own class and closer to other classes, suggesting proximity to decision boundaries.

$$x_b = \arg \min_{D(x_i) \in C \setminus \{c_x\}} \|\overrightarrow{\alpha_N}(x_a) - \overrightarrow{\alpha_N}(x_i)\| \tag{9}$$

$$\text{dist}_b = \|\overrightarrow{\alpha_N}(x_a) - \overrightarrow{\alpha_N}(x_b)\| \tag{10}$$

$$\text{DSC}(x) = \frac{\text{dist}_a}{\text{dist}_b} \tag{11}$$

3.3.3 Mahalanobis Distance-based Surprise Coverage

Mahalanobis Distance-based Surprise Coverage extends distance-based formulations by measuring the deviation of a test input from the distribution of training data rather than from individual samples (Kim et al. 2023). Unlike DSC, which relies on distances between nearest neighbors, MDSC models the distribution of latent space vectors using both class-wise centroids and a covariance matrix.

First, a mean vector μ_c is estimated for each output class c from the training latent vectors. Next, all training latent vectors are centered with respect to their class means, and a pooled covariance matrix Σ is estimated from the resulting class-centered traces. Given a test input x with predicted class c_x, its MDSC score is computed using Equation 12 as the Mahalanobis distance between its latent representation and the class-wise mean μ_{c_x} using the shared covariance matrix.

$$\text{MDSC}(x) = (\overrightarrow{\alpha_N}(x) - \mu_{c_x})^T \Sigma^{-1} (\overrightarrow{\alpha_N}(x) - \mu_{c_x}) \tag{12}$$

MDSC increases when the latent space vector of a test input x lies farther from the learned training distribution. Unlike DSA, it does not require computing the distance to all individual training samples once the global statistics have been computed. Instead, it relies only on the corresponding mean vector and covariance matrix (Kim et al. 2023). This design makes MDSC computationally more efficient than LSC and DSC once precomputation is complete, because it uses neither KDE nor pairwise distances to individual training inputs. However, it requires a precomputation step and the estimation and storage of covariance matrices (and corresponding mean vectors), which must be reused across the evaluation of the dataset (Kim et al. 2023).

Finally, for a test dataset X, the coverage values for LSC, DSC, and MDSC are computed using Equation 13. The corresponding metric values for all inputs $x \in X$ are discretized into k segments defined by an upper bound U and a set of buckets $B = \{b_1, b_2, \ldots, b_k\}$ that partition the interval $(0, U]$. The number of segments k is a hyperparameter whose selection significantly affects the calculation coverage values. While the original definitions of LSC, DSC, and MDSC yield ratios between 0 and 1, we report these values on a 0–100% scale in our results to maintain consistency with structural coverage criteria.

$$\begin{aligned} \mathcal{M}(X) &= \frac{1}{k} \left| \left\{ b_i \mid \exists x \in X : \mathcal{M}(x) \in \mathcal{I}_i \right\} \right| \times 100 \\ &\text{where } \mathcal{M} = \text{LSC/DSC/MDSC} \\ &\text{and } \mathcal{I}_i = \left(U \cdot \frac{i-1}{k}, U \cdot \frac{i}{k} \right] \end{aligned} \tag{13}$$

3.3.4 Latent-Space Properties Captured by DSC and MDSC

Figure 2 illustrates a potential limitation while using DSC using the example of two new test inputs X_1 and X_2 belonging to the same class. On these inputs, the DNN under test classifies X_1 correctly and X_2 incorrectly. First, DSC identifies the nearest latent space vectors from the training dataset and calculates the Euclidean distance a_1 and a_2 for X_1 and X_2, respectively. In the second step, DSC calculates inter-class distances b_1 and b_2 between these test inputs and the nearest vectors from other classes. As a result of this, DSC calculated using Equation 11 results in $\frac{a_1}{b_1} \approx \frac{a_2}{b_2}$. Furthermore, the presence of outliers in the training dataset can potentially skew the results, as the first step in DSC calculation relies on nearest-neighbor distances. These factors lead to a key limitation: DSC may fail to distinguish between correctly

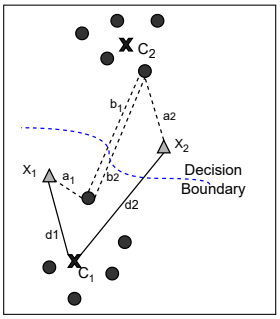


Fig. 2: Illustration of DSC and LSCD calculation. Black dots represent latent space vectors from the training set for classes 1 and 2 with their centers C_1 and C_2. New test inputs x_1 (correctly classified) and x_2 (falsely classified), both belonging to class 1, are shown to demonstrate the metric calculations.

classified samples and misclassified samples. Another limitation of using DSC is its computational complexity, which increases with the size of the training data.

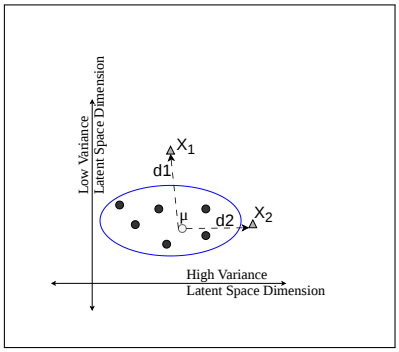


Fig. 3: Geometric interpretation of MDSC and LSCD in latent space. Black dots denote training samples from the same class cluster with centroid μ. Two test inputs, X_1 and X_2, are shown. MDSC and LSCD assign different scores because MDSC normalizes distances by covariance structure, while LSCD preserves direct Euclidean dispersion around the centroid. The figure illustrates different inductive biases rather than an inherent deficiency of either metric.

As illustrated in Figure 3, MDSC partially addresses these issues by modeling the training data distribution using class means μ together with a covariance matrix. It measures deviation from a distribution rather than from individual samples. Compared to DSC, this formulation reduces sensitivity to local nearest-neighbor effects and can more effectively capture deviations that align with the overall structure of the latent space. However, as shown in Figure 3, two test inputs X_1 and X_2 may exhibit similar Euclidean distances from the class centroid μ (denoted by d_1 and d_2), yet differ in how they align with the underlying data distribution. MDSC normalizes these distances

using covariance, which leads to deviations along high-variance directions being treated as less significant than those along low-variance directions. This results in X_2 being less surprising than X_1, even though both inputs are similarly distant from the centroid.

These observations suggest that existing latent-space metrics capture different latent-space properties. DSC captures local nearest-neighbor relationships, and MDSC captures covariance-normalized deviation from class distributions. Neither DSC nor MDSC explicitly captures the overall dispersion of test inputs around class centroids. Since our goal is to identify computationally efficient approximations of the mutation score, we additionally investigate whether direct latent-space dispersion provides a useful signal. To study this hypothesis, we operationalize direct latent-space dispersion through Latent Space Class Dispersion (LSCD). Since mutation score measures behavioral differences between the original and mutant models, it is not obvious a priori which of these signals should align most closely with MS. We therefore evaluate this relationship empirically in Section 5.

3.4 Latent Space Class Dispersion: Definition and Interpretation

To operationalize direct latent-space dispersion, we define Latent Space Class Dispersion (LSCD), a metric that quantifies the dispersion of latent representations around class centroids. LSCD characterizes the overall geometric spread of test inputs within each class and captures a latent-space property that complements existing metrics. The $LSCD_i$ is a per-class evaluation metric defined in Equation 14.

$$LSCD_i = \frac{1}{|X_i|} \sum_{x^{(j)} \in X_i} \left\| \overrightarrow{\alpha_N}(x^{(j)}) - \mu_i \right\| \tag{14}$$

First, the mean vector μ_i of each output class i is computed using the training dataset. Let X_i denote the set of input samples belonging to class i, where each sample $x^{(j)}$ is mapped to a latent representation $\overrightarrow{\alpha_N}(x^{(j)})$. We define $LSCD_i$ as the mean Euclidean distance of these latent representations from the corresponding class mean μ_i. Note that μ_i corresponds to the class centroid and is consistent with the mean used in MDSC formulations. The overall $LSCD$ score is calculated using Equation 15, which aggregates the $LSCD_i$ values for all C output classes.

$$LSCD = \frac{\sum_{i=1}^{C} LSCD_i}{C} \tag{15}$$

LSCD is not intended to replace distribution-aware metrics such as MDSC. Rather, it captures a complementary latent-space property: the overall dispersion of test inputs around class centroids. In this work, we evaluate whether this property provides a useful approximation of mutation-score-based test dataset quality.

LSCD Interpretation. We discuss how test engineers can interpret LSCD scores in conjunction with traditional evaluation metrics, e.g., accuracy. High accuracy (e.g., $> 90\%$) combined with low LSCD values (i.e., comparable to those of the training dataset) indicates that most test inputs are located in close proximity to the learned class regions in the latent space. In this case, the test data remain close to the learned

class regions of the DNN and primarily reflect the features and distributions already observed during training. While such results are consistent with a well-trained DNN in terms of accuracy, they also indicate that other regions of the latent space remain untested by the current test suite. Thus, the test data exhibit poor diversity, which threatens the reliability of DNN's real-world assessments because in-field data are unlikely to always resemble the training data.

In contrast, higher LSCD values indicate that test inputs are more dispersed from their corresponding class centroids, suggesting that they explore sparse or previously untested regions of the latent space. However, a higher LSCD value should be interpreted as indicating broader dispersion in latent space rather than as by itself guaranteeing the semantic validity of the corresponding test inputs. As illustrated in Figure 2, distances d_1 and d_2 correspond to the distances of X_1 and X_2 from their class centroid. In a well-trained DNN, d_1 (for the correctly classified input X_1) is smaller than d_2 (for the misclassified input X_2), allowing LSCD to differentiate between correctly and incorrectly classified samples. Furthermore, as shown in Figure 3, LSCD captures dispersion directly without covariance-based normalization. Even when two inputs have comparable distances from the centroid, MDSC may treat them differently due to variance normalization, whereas LSCD reflects their direct Euclidean dispersion around the class centroid. This enables LSCD to characterize the overall geometric dispersion of data around class centroids.

By directly measuring dispersion, LSCD captures a complementary latent-space property to DSC and MDSC. Unlike DSC, it does not depend on local nearest-neighbor relationships and is therefore less sensitive to outliers and dataset size. Unlike MDSC, it does not rely on covariance estimation and therefore reflects dispersion directly around class centroids without covariance-based reweighting. This makes LSCD computationally simple while providing a direct measure of latent-space dispersion around class centroids.

In this paper, we also scrutinize corner case data generated using the CGF framework (Section 3.5) that shows improved test dataset quality through increased LSCD. In our study, we consider corner case data in image classification as inputs that are misclassified by the DNN under test. A high LSCD value for corner case data, compared with the original train and test sets, indicates that many inputs lie farther from the trained class regions and therefore explore *previously untested regions of the latent space*. The combination of low accuracy scores (e.g., $< 5\%$) and higher LSCD values compared with the train and test sets suggests that these corner cases constitute a more challenging and diverse test set, as the DNN fails to generalize reliably to such rare or highly varied inputs.

Finally, LSCD offers practical advantages. Unlike surprise coverage metrics, it does not require discretization using a predefined number of buckets, making it a continuous and more robust measure across datasets and architectures. Moreover, class-wise $LSCD_i$ scores provide fine-grained insights into model behavior, enabling practitioners to identify underrepresented classes and guide targeted retraining or robustness improvements.

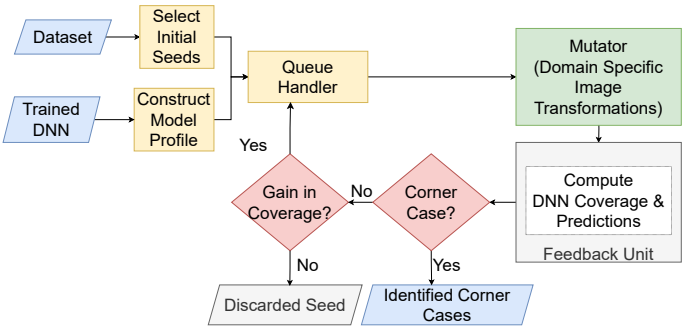


Fig. 4: Coverage-Guided Fuzzing used in our evaluation.

3.5 Corner Case Generation using Coverage-Guided Fuzzing

We use coverage-guided fuzzing to sample corner case data to improve the original test dataset quality, which was evaluated using test adequacy metrics and mutation scores. As discussed before, a test dataset that covers the untested regions of latent space has higher potential to verify a DNN's generalizability. A test dataset that maximizes the latent space coverage can be achieved through two primary approaches: (1) upsampling latent space vectors from the untested latent space region and generating an input using a decoder or (2) using some automated tool for generating inputs with desired properties that can increase the latent space coverage.

Previous work by Mani et al. (Mani et al. 2019) focuses on the first case, generating corner case images by upsampling and traversing through the latent space vectors rather than randomly selecting. However, the validity of the generated images strictly depends on the quality of the decoder used to reconstruct the image from the latent space vector, which may fail to reliably reconstruct the image from the latent space vector. Moreover, the use of a decoder often results in the generation of many invalid images that contain only noise and are semantically distinct from the original test data (Mani et al. 2019). Indeed, latent space exploration for test generation is still an open research topic (Riccio and Tonella 2023; Maryam et al.; Weißl et al. 2025).

Therefore, in our work, we focus on the second case and remove the dependency on a decoder-based DNN to generate corner case images. To this end, we use CGF, a promising technique for the automated generation of corner cases of DNNs using various coverage criteria like NC, k-MNC, NBC, frequently used in previous works (Xie et al. 2019; Pei et al. 2019; Ma et al. 2018; Tian et al. 2018). As highlighted in several industry surveys (Zhang et al. 2022; Li et al. 2022), CGF is also adopted in practice as an effective method for automatically uncovering corner cases in DNNs leveraging available test datasets. In contrast to CGF, adversarial attack methods typically generate imperceptible perturbations that do not correspond to realistic or semantically meaningful corner cases. As a result, they often fail to capture the kinds of failures that actually show up in real-world systems. Additionally, we also incorporate LSC and DSC as coverage criteria during fuzzing for corner case generation, which, to the best of our knowledge, has not been examined in previous work.

Table 1: Summary of hyper-parameters for coverage-guided fuzzing used to sample corner case data.

Fuzzing step	Requirement	Hyper-parameters
Selection of Initial Seeds	To maximize fuzzing efficiency	Correctly classified images in classification
Coverage Criteria	To maximize DNN's behavior	NC, k-MNC, NBC with hyper-parameters 0.75, 1500, and 10 respectively LSC and DSC: positive gain in surprise score relative to the original input
Mutator	To add realistic transformation	Widely used domain-specific transformation (Hendrycks and Dietterich 2019; Stocco et al. 2022)
Corner Case Identification	Criteria to select corner cases	Falsely classified images
Number of iterations	Stopping criteria for fuzzing	15,000 iterations for all classification DNNs

The use of coverage-guided fuzzing helps generate diverse test data by leveraging annotated test data and identifying hidden defects and weaknesses of the DNN under test. The coverage criteria used in CGF aim to verify the maximum possible behaviors of DNN against various input conditions, which can be useful in increasing the diversity of the original test data. Our experimental study empirically validates this hypothesis (Section 4).

The CGF framework is depicted in Figure 4. The fuzzing process starts with the selection of initial seeds on which the DNN has high performance as reflected by evaluation metrics. Then, the mutator applies a transformation to the selected seed image. If a transformed image helps increase the currently used coverage criterion, it is added to the queue as it helps explore the maximum possible behavior of the DNN. Corner cases are identified if the DNN makes false predictions on these transformed images. A detailed summary of the hyper-parameters associated with fuzzing in our case study can be found in Table 1. We run fuzzing until the coverage criterion used reaches a saturation level because continuing with the same criteria might lead to diminishing returns in fuzzing efficiency. This is because the same criteria are likely to identify fewer corner cases over time. We calculate and compare the standard evaluation metrics, test adequacy metrics, and mutation scores on fuzzing data to demonstrate the enhancement in test dataset quality as indicated by these metrics.

4 Empirical Study

4.1 Research Questions

RQ_1 (Correlation Study): *What is the correlation between mutation score and various test adequacy metrics?*

We performed an empirical study to evaluate which metric among traditional test adequacy metrics, along with LSCD, can be used as a computationally efficient approximation for mutation scores by computing their correlation with MS using the Pearson and Spearman correlation coefficients (Wikipedia 2024; Spearman 1904). To do so, we first quantify the dataset quality of both the original test data and corner case data using four structural, three non-structural test adequacy metrics, LSCD, and MS obtained from a set of mutant models. We then calculate the Pearson and Spearman correlation coefficients, along with the corresponding statistical significance tests.

$RQ_{1.1}$ (Corner Case Validity): *What is the validity of the corner case data as assessed by automated input validators?*

Table 2: Overview of Datasets

Dataset	No. of Output Classes	Number of Images							
		Training	Validation	Test	Corner Case Images using				
					NC	k-MNC	NBC	LSC	DSC
MNIST	10	48,000	12,000	10,000	14,129	12,868	14,006	15,610	15,598
SVHN	10	62,268	10,989	26,032	14,390	14,216	14,339	14,306	14,300
GTSRB	43	35,610	3,870	12,630	13,327	14,476	14,249	12,116	12,327
CIFAR-100	100	40,000	10,000	10,000	90,933	95,790	85,063	97,323	98,249

As higher scores of test adequacy metrics and MS can be achieved with noisy and "unrealistic" data, it is essential to assess whether increases in these values are associated with the generation of inputs that are valid in the sense of being "realistic", that is, recognizable by humans in the input domain (Riccio and Tonella 2023). To retain the realism and reliability of our correlation study in RQ_1, we hence assess whether the corner case data are valid according to an automated input validator from the literature.

RQ_2 (Computational Time Required): *What is the computational efficiency of different test adequacy metrics?*

In this research question, we systematically compare the computation time of different test adequacy metrics across both structural and non-structural approaches. Prior work (Weiss et al. 2021; Kim et al. 2023) highlights the high computational overhead of the original DSC calculation, particularly when implemented using a single-thread (ST) execution model (Kim et al. 2019). To address this limitation, Weiss et al. (Weiss et al. 2021) proposed a multi-threaded (MT) refinement of DSC, significantly improving its efficiency. Accordingly, we consider both ST and MT implementations of the DSC metric in our analysis.

4.2 Datasets and Deep Neural Networks

Our empirical study encompasses datasets of increasing diversity and complexity, along with a range of DNN architectures, to enhance the reliability and generality of our findings. Particularly, we consider the popular MNIST (foundational digit recognition) (Deng 2012), SVHN (real-world house number identification) (Netzer et al. 2011), GTSRB (safety-critical traffic sign recognition) (Stallkamp et al. 2011), and CIFAR-100 (high-cardinality object recognition) (Krizhevsky 2009) datasets for image classification with DNNs. Table 2 reports key dataset statistics, such as the number of output classes and their size (divided into training, validation, test set, and identified corner case images using CGF).

MNIST (Deng 2012) is a dataset of hand-written digits depicted as gray-scale images, often used in DNN testing works. SVHN (Netzer et al. 2011) is a large-scale dataset from Google Street View images that comprises real-world images of house numbers. SVHN is helpful in studying advanced classification DNNs due to its complexity and diversity. GTSRB (Stallkamp et al. 2011) is a benchmark dataset for German traffic sign recognition. It is a feature-rich dataset consisting of images with various illumination changes, rotations, and occlusions likely to be encountered in

Table 3: Details about DNN architectures used for the empirical study.

Network	Layer	Parameters/Description
LeNet-5	Convolutional 1	6 filters, 5x5 kernel, stride 1, ReLU, Average Pooling
	Convolutional 2	16 filters, 5x5 kernel, stride 1, ReLU, Average Pooling
	Fully Connected 1	Flatten and 120 neurons, ReLU
	Fully Connected 2	84 neurons, ReLU
	Output (Logit Layer + Softmax)	10 neurons (MNIST), Softmax
Spatial Transformer Network	Localization Network	Convolutional layers (learn affine transformation parameters)
	Grid Generator	Creates sampling grid
	Sampler	Samples input feature map
	Convolutional 1	100 filters, 5x5 kernel, LeakyReLU, BatchNormalization
	Convolutional 2	150 filters, 3x3 kernel, LeakyReLU, BatchNormalization
	Convolutional 3	250 filters, 3x3 kernel, LeakyReLU, BatchNormalization
	Fully Connected 1	Flatten and 350 neurons, ReLU
	Output (Logit Layer + Softmax)	10 and 43 neurons (SVHN and GTSRB respectively), Softmax
VGG-16	Convolutional Blocks 1–5	13 layers total; 3x3 kernels, stride 1, padding 1, ReLU
	Batch Normalization Blocks 1–5	13 layers total; Output Size Same as Convolutional layers
	Max Pooling 1–5	2x2 kernel, stride 2 (after each block)
	Fully Connected 1	512 neurons, ReLU
	Fully Connected 2	512 neurons, ReLU
	Output (Logit Layer + Softmax)	100 neurons (CIFAR-100), Softmax

actual driving. Thus, it serves as a benchmark to test safety-critical traffic sign classification. CIFAR-100 (Krizhevsky 2009) is a high-complexity benchmark consisting of 100 diverse object categories and is widely used to benchmark modern deep architectures. Its complexity allows us to assess adequacy metrics on a more challenging and demanding dataset.

We use well-established benchmark architectures based on LeNet-5 (Lecun et al. 1998), Spatial Transformer Networks (STNs) (Hira 2026), and VGG-16 (Simonyan and Zisserman 2014), in our study to cover models of varying complexity and latent-space structure. The choice of LeNet-5 for MNIST helps understand the latent space and its properties under less complex settings. On the other hand, using the same STN architecture for SVHN and GTSRB allows us to understand the latent space for datasets of varying complexity and output dimensions. For CIFAR-100, we adopt the widely used VGG-16 architecture, which provides a deeper and more complex model suitable for high-cardinality object recognition. These DNNs are trained using the Adam (Kingma and Ba 2017) optimizer with a learning rate scheduler and early stopping criteria for 50 epochs (MNIST), 100 epochs (SVHN and GTSRB), and 200 epochs (CIFAR-100). The architectures of the DNNs and their performance using test data are presented in Table 3 and Table 5, respectively.

4.3 Latent Space Vector Selection

To calculate non-structural metrics LSC, DSC, MDSC, and LSCD, first, we need to select latent space vectors. A fully connected layer is a common type of output layer used in DNNs for the image classification task. The size of the fully connected layer

is equal to the number of output classes. Thus, the latent space vector is a vector of a constant size corresponding to the given input image. As highlighted in previous studies (Bengio et al. 2012; Kim et al. 2019, 2023), we also used the latent space vector derived from the last fully connected layer (often referred to as the logit layer as shown in Table 3) for the calculations. Although we use the last fully connected layer in this study, latent-space-based metrics, including LSCD, are not inherently tied to this specific representation choice. Applying them to other architectures or domains primarily requires selecting an appropriate latent-space representation layer.

4.4 Procedure and Metrics

4.4.1 RQ_1: Correlation Study

To study the correlation between various test adequacy metrics and mutation score, we begin with a baseline evaluation of the DNN under test. This involves calculating accuracy and a set of structural and non-structural test adequacy metrics using the original train and test datasets. Then, we generate the corner case dataset using NC, k-MNC, NBC, LSC, and DSC as coverage criteria during fuzzing. These metrics are selected because they collectively capture diverse behavioral aspects of the DNN, including structural activation patterns and latent-space deviations, thus providing broad coverage across different testing perspectives. In contrast, metrics such as SNAC, MDSC, and LSCD are not used to guide the test data generation. Using the same metric for both test generation and evaluation may introduce bias, as the generated data would be optimized toward that specific metric. Therefore, these metrics are used solely for evaluation purposes to ensure a fair and unbiased comparison.

After generating the corner-case dataset, we recompute all metrics on the generated test set. The NC, k-MNC, NBC, and SNAC values were computed using the formulas in Section 3.2. We chose an NC threshold of 0.75 and used 1500 sections for the other three metrics, which aligns with common settings from previous work (Xie et al. 2019). The LSC, DSC, and MDSC results are calculated using Equation 13 using bucket size k=1000, which is commonly used in the literature (Kim et al. 2023; Weiss et al. 2021). The LSCD is calculated using Equation 15. Unless otherwise stated, the metric definitions described in Section 3.3 use their standard formulations. In the primary evaluation setting of this study, LSCD and MDSC are computed using ground-truth classes of the test dataset to align their comparison under the labeled-dataset setting. To assess whether the observed relationship with mutation score depends on this choice, we also evaluate LSCD and MDSC using the predicted class of test inputs separately. This additional analysis is limited to LSCD and MDSC because they are the closest latent-space comparators in the study and, therefore, most relevant for assessing whether the main comparison depends on class conditioning.

To investigate which metric from a diverse set of available test adequacy metrics is better suited for evaluating dataset quality and approximating mutation scores, we study their correlation with mutation scores, which assess test dataset quality based on their ability to capture faults in DNNs. We consider the fault categories proposed by Humbatova et al (Humbatova et al. 2020). We identified 15 unique DNN faults that belong to 3 (Models, Training and API) out of 5 top-level categories of the final

Table 4: Fault names, mutation operators, and their associated hyper-parameters employed to generate mutant models (Humbatova et al. 2020).

No.	Fault Group	Fault Name	Mutation Operator	Hyper-Parameters Used
1	Training - Hyper parameters	Suboptimal batchsize	Increase Batch Size	64
2			Decrease Batch Size	2048, 4096
3		Suboptimal learning rate	Decrease Learning Rate	0.0001
4			Increase Learning Rate	0.01, 1
5		Wrong optimizer function	Change Optimizer	Adagrad
6	API	Missing API Call	Remove Zero Grad	API call
7			Remove Call	API call
8	Training - Training Data	Low quality training data	Add Training Noise	0.25, 0.75, 0.9
9			Remove Samples	0.25, 0.75, 0.9
10		Overlapping output classes	Make Classes Overlap	0.25, 0.75, 0.9
11		Wrong labels for training data	Change Most Labels (in %)	25, 60
12	Model - Activation Function	Wrong activation / Missing Relu activation function	Change Activation	Tanh, LogSigmoid, Sigmoid
13	Model - Layer Properties	Wrong filter size for convolutional and FC layers	Layer Size	Output channels changed
14		Bias needed in a layer	Remove Bias	according to architecture
15	Model - Missing/Redundant /Wrong Layer	Missing dropout layer	Dropout	0.25, 0.8

taxonomy. The two omitted categories (GPU Usage and Tensors) pertain to the use of incorrect hardware and to the use of incorrect tensors for training DNNs, which do not apply in our implementation. Moreover, we used selected faults to define MOs corresponding to them. We trained 233 mutant models for the classification DNNs using these MOs. The total number of mutants varies across datasets and mutation operators. For instance, certain MOs, such as overlapping output class mutations, produce more mutants when applied to datasets with a larger number of classes (e.g., CIFAR-100), leading to an uneven distribution of mutants across configurations. A detailed description of fault categories, MOs, and corresponding hyperparameters is provided in Table 4.

We calculate MS using Equation 16, a modified version of the formula proposed in DeepMutation++ (Hu et al. 2019).

$$MS_{m'} = \frac{|\{t \mid t \in T \wedge m(t) \neq m'(t)\}|}{|T|} \tag{16}$$

Here, T denotes the evaluated dataset used in our analysis, i.e., using the original test dataset or the validated corner-case dataset. The mutation score $MS_{m'}$ measures the discrepancy between the predictions of a mutant model m' and the original model m on the test dataset $t \in T$, as defined in Equation 16. The only modification compared to DeepMutation++ is not aggregating the MS for all mutant models. In this way, we calculate individual test adequacy metrics and MS for all mutant models.

After calculating these metrics, we employ both Pearson and Spearman correlation coefficients (Virtanen et al. 2020; Wikipedia 2024; Spearman 1904) to analyze the relationship between test adequacy metrics and mutation score. Pearson correlation

captures linear relationships, whereas Spearman correlation measures monotonic relationships based on ranks and is less sensitive to outliers. Since the relationship between test adequacy metrics and mutation score may not be strictly linear, Spearman's rank correlation provides a more robust indication of whether the relative ordering of datasets is preserved across metrics.

A strong positive correlation between MS and any of the test adequacy metrics would support the proposition that these metrics offer a computationally inexpensive way to quantify test dataset quality, as they do not rely on mutant models. The metric with the highest correlation coefficient can therefore be considered a more suitable alternative for estimating mutation score. To ensure that the observed correlations are not due to random variation, we perform statistical significance testing using the p-value, with a conventional threshold of 0.05. A p-value below this threshold indicates that the observed correlation is statistically significant and unlikely to have occurred by chance.

4.4.2 $RQ_{1.1}$: Corner Case Validity

The results of this correlation study can be adversely affected by the inclusion of invalid and inherently poor-quality corner case images. A higher MS and test adequacy scores can also be achieved when using noisy images that are barely or not recognizable by humans as well. Thus, we also assess and report the validity and quality of the identified corner cases using an automated input validator, SelfOracle (Stocco et al. 2020). It is a distribution-aware input validator for image data, which has shown high agreement with the human assessment of validity in a large comparative study about test input generators for deep learning (Riccio and Tonella 2023).

SelfOracle is based on a Variational Autoencoder (VAE) (An and Cho 2015) trained to minimize the reconstruction error between the original and reconstructed inputs. After training, the VAE will have a higher reconstruction error on invalid data (highly corrupted data that fall out-of-distribution), whereas it will exhibit lower reconstruction errors for valid, in-distribution data (Riccio and Tonella 2023). SelfOracle leverages probability distribution fitting to find the threshold that discriminates valid inputs from invalid ones (Riccio and Tonella 2023). It fits a Gamma distribution to the training data through maximum likelihood estimation (Choi and Wette 1969). Probability distribution fitting provides a statistical model for the reconstruction errors of nominal data. This statistical model can be used by the tester to configure the threshold with the desired rate of false alarms (i.e., inputs that belong to the nominal dataset but are classified as invalid).

Using the same hyperparameters of the original study (Riccio and Tonella 2023), we trained SelfOracle using the training set of each dataset for 100 epochs, with a learning rate of 0.001. The network uses the Adam optimizer to minimize the mean squared error (MSE) loss. The latent space dimension was 1024. After training, we applied SelfOracle to reconstruct all corner case images for all datasets using the false alarms rate of the original study (0.01%). Inputs with a reconstruction error above this threshold are considered invalid, while inputs below it are considered valid (Riccio and Tonella 2023).

4.4.3 RQ_2: Computational Time Required

To compute the computational time required to calculate the structural test adequacy metrics, LSC and MDSC, we use their available single-threaded versions. The original DSC calculation uses single-thread execution, which makes it computationally expensive (Kim et al. 2019; Weiss et al. 2021). Thus, Weiss et al. (Weiss et al. 2021) proposed a refined multi-threaded DSC calculation, which is computationally faster. Therefore, we report computation times for both single-threaded and multi-threaded DSC to highlight this difference. LSCD, in contrast, is inherently simpler and can be efficiently computed using a single-threaded implementation. It only requires calculating distances from class centroids without additional pairwise or covariance-based operations. Since all non-structural metrics require latent space vectors, we precompute and store the latent vectors for all inputs. This ensures that the reported computation time reflects only the cost of metric calculation, enabling a fair comparison across different approaches.

5 Results

5.1 RQ_1 (Correlation Study)

Table 5 reports the performance of DNNs for image classification across all four datasets. The accuracies on the original test datasets are 98.60%, 92.83%, 99.71%, and 74.00% for the MNIST, SVHN, GTSRB, and CIFAR-100 datasets, respectively. As expected, performance drops substantially on the generated corner case data. The mean accuracy on CIFAR-100 corner cases dropped to the lowest 0.53%, while the accuracy on MNIST, SVHN, and GTSRB corner cases decreased to 3.09%, 9.48%, and 1.54%, respectively. This indicates that the generated corner case inputs are more challenging than the original test data.

The dataset quality evaluation using structural test adequacy metrics is expected to yield higher metric values for corner case data since these inputs are more likely to activate additional neuron behaviors that were not observed during the training phase. Table 5 shows that the NC metric evaluated using MNIST test data resulted in a value of 69.49%, which only slightly increased to 73.30% when the corner case dataset (NC) was used instead. In contrast, the k-MNC metric demonstrated a significant and unexpected drop. Its value dropped from 76.76% on the original MNIST test data to just 43.09% when evaluated using the corner case dataset (NC). This drop in k-MNC can be explained by the activation values of neurons that fall outside the typical $[low, high]$ bounds observed during training for corner cases. This is further corroborated by the fact that the NBC and SNAC scores increased from 2.18% and 3.09% to 11.69% and 21.05%, respectively. Overall, NC, NBC, and SNAC tend to increase on corner-case data to varying degrees. In contrast, k-MNC appears less reliable at distinguishing challenging corner case data from the original test distribution.

For the non-structural metrics, higher values are generally expected for corner case data because these inputs are more "surprising" to the deep neural network with respect to the training distribution. This is because inputs with complex features, such as corner case data, are less likely to come from the same distribution as the

Table 5: Summary of performance evaluation of DNNs using Accuracy and test dataset quality assessment using traditional test adequacy metrics and LSCD. The metric values for all metrics except LSCD are in (%).

Dataset	Data Split	Evaluation Metrics								
		Accuracy	**NC**	**k-MNC**	**NBC**	**SNAC**	**LSC**	**DSC**	**MDSC**	**LSCD**
MNIST	Test data	98.60	69.49	76.76	2.18	3.09	46.00	43.00	54. 10	25.27
	Corner Case (using NC)	3.93	73.30	43.09	11.69	21.05	30.00	61.00	82.10	57.17
	Corner Case (using k-MNC)	3.04	73.72	43.64	11.99	21.52	35.00	57.00	84.40	57.79
	Corner Case (using NBC)	3.08	72.88	42.86	11.97	21.39	28.00	59.00	82.30	57.56
	Corner Case (using LSC)	2.82	73.30	43.69	11.73	21.10	29.00	60.00	82.60	57.78
	Corner Case (using DSC)	2.59	72.88	42.06	11.88	21.39	29.00	63.00	84.0	58.15
SVHN	Test data	92.68	57.48	69.44	3.10	3.20	49.00	62.00	60.40	6.14
	Corner Case (using NC)	9.46	71.39	47.10	7.11	8.17	42.00	53.00	73.0	10.73
	Corner Case (using k-MNC)	9.46	72.88	46.82	7.40	8.13	39.00	51.00	73.50	10.74
	Corner Case (using NBC)	9.76	71.54	47.13	7.46	8.29	27.00	49.00	67.40	10.73
	Corner Case (using LSC)	9.62	71.68	47.01	6.87	7.86	40.00	65.00	66.20	10.71
	Corner Case (using DSC)	9.13	72.17	46.85	7.16	8.2	40.00	49.00	71.20	10.74
GTSRB	Test data	99.71	52.17	75.62	3.68	4.28	24.00	58.00	68.50	55.71
	Corner Case (using NC)	1.61	81.98	49.92	17.74	27.52	27.00	77.00	77.80	186.73
	Corner Case (using k-MNC)	1.50	80.95	50.11	17.86	27.70	23.00	80.00	79.0	188.31
	Corner Case (using NBC)	1.33	82.33	50.14	19.13	28.94	39.00	79.00	64.20	189.09
	Corner Case (using LSC)	1.77	79.57	49.74	17.52	27.30	40.00	74.00	77.00	184.54
	Corner Case (using DSC)	1.53	78.95	49.64	17.57	27.32	33.00	74.00	75.90	184.49
CIFAR-100	Test data	74.00	44.89	62.99	2.52	1.74	14.00	15.00	78.00	17.05
	Corner Case (using NC)	0.53	60.92	65.72	9.97	8.35	34.00	50.00	87.60	45.00
	Corner Case (using k-MNC)	0.56	59.42	65.42	9.68	7.99	31.00	53.00	85.00	45.34
	Corner Case (using NBC)	0.54	61.46	68.42	13.4	11.85	24.00	50.00	87.10	45.39
	Corner Case (using LSC)	0.53	59.79	65.51	10.07	8.20	34.00	50.00	87.80	45.31
	Corner Case (using DSC)	0.53	59.25	65.46	9.79	7.98	31.00	50.00	82.80	45.2

training data, which leads to a higher LSC. Similarly, their latent space vectors are often far from the training distribution, placed in sparse or near-decision boundary regions, resulting in a higher DSC and MDSC. Contrary to this expectation, LSC values are lower for most of the corner cases compared to the original test data, except CIFAR-100. For the MNIST dataset, the original test dataset and corner case data (NC) have an LSC of 46 and 30, respectively. A possible reason is that the corner cases in our study were generated using a fuzzing technique that applies global image transformations. Unlike adversarial images, which are meticulously crafted to be "out-of-distribution", these image transformations might still produce images that are more likely to overlap with the original training data's distribution. In contrast, CIFAR-100 behaves differently. LSC values for CIFAR-100 original test data and corner case data (NC) are 14 and 34, respectively. It indicates that the generated corner cases are more "surprising" with respect to the training distribution than the original dataset. This is attributed to its classes exhibiting far greater intra-class variability, and the DNNs are more sensitive to global changes in texture, color, contrast, etc. These factors make the transformed images more likely to fall into low-density regions of the learned distribution.

DSC and MDSC generally increase for corner-case inputs, but they exhibit different sensitivities across datasets. On MNIST, DSC rises from 43 on the original test data to 61 on the NC-guided corner cases, while MDSC increases more strongly from 54.10 to 82.10, respectively. Similar increases are observed for GTSRB and CIFAR-100. These increases indicate that corner-case inputs deviate more from the training distribution

and exhibit greater diversity in the latent space than the original test data. However, on SVHN, DSC decreases from 62.00 on the original test set to 53.00 on the NC-guided corner cases, whereas MDSC increases from 60.40 to 73.00. This discrepancy can be attributed to DSC's sensitivity to local nearest-neighbor relationships using Equation 10, which may be affected by outliers or noise in the training distribution. In contrast, MDSC measures deviation with respect to the overall class distribution, resulting in a more stable and consistent increase. These observations suggest that MDSC is less influenced by outliers and can better capture distribution-aware latent-space deviation.

Despite this improvement, MDSC inherently normalizes distances based on covariance. Its values are influenced by covariance-based normalization, and the final coverage score is obtained through bucketing. As a result, changes in MDSC do not always translate into a strong separation between original and corner-case datasets. For example, in SVHN, MDSC increases from 60.40 to 73.00, and on CIFAR-100 from 78.00 to 87.60, indicating only a moderate increase. This reflects the fact that MDSC captures deviation relative to the learned distribution shape, rather than the raw spread of test inputs around class centers. To complement this limitation, we use LSCD, which directly measures the Euclidean dispersion of test inputs with respect to class centroids and does so without covariance-based normalization or bucket discretization.

The results in Table 5 show that LSCD consistently distinguishes between original test data and corner-case datasets. The LSCD values of the MNIST train, test, and corner case data (NC) are 25.10, 25.27, and 57.17, respectively. The similar LSCD values of train and test data suggest that they exhibit similar class dispersion around the derived class centers and follow the learned class distribution. In contrast, the LSCD value of the corner case data is significantly higher than the original train and test data, clearly indicating that the misclassified data has higher latent space dispersion than the correctly classified data. The LSCD values for SVHN train, test, and corner case data (NC) are 6.08, 6.14, and 10.73, respectively. These increases suggest that corner case inputs are more dispersed from the learned class regions in latent space.

While both MDSC and LSCD capture deviations between original and corner case datasets, they reflect different aspects of latent space behavior. MDSC emphasizes distribution-aware deviation, whereas LSCD provides a direct measure of dispersion relative to class centers. We further analyze the relative effectiveness of these metrics using correlation with mutation score.

Table 6 and Table 7 present the Pearson and Spearman's rank correlation between the studied test adequacy metrics and mutation score across the four datasets, respectively. The varying number of mutant DNNs results from various factors when applying MOs. For example, using an overlapping output class mutation operator on a dataset with a large number of output classes, such as the CIFAR-100 dataset, can yield more mutant models. Furthermore, we recognized that the LSC, DSC, and MDSC calculations are sensitive to the bucket size k used in Equation 13. To mitigate any potential bias in the correlation study, we compute their correlations across bucket sizes in the range $[100, 1000]$ with a step size of 100 and report the mean correlation values to reduce bucket-specific bias.

Table 6: Correlation Analysis using **Pearson Correlation Coefficient** of structural and non-structural Coverage Criteria vs. Mutation Score (MS) per Dataset.

Dataset	No. of Mutant DNNs	NC v/s MS		k-MNC v/s MS		NBC v/s MS		SNAC v/s MS	
		Corr.	p-value	Corr.	p-value	Corr.	p-value	Corr.	p-value
MNIST	45	0.040	0.823	-0.096	0.588	0.029	0.941	0.111	0.549
SVHN	46	0.168	0.509	-0.138	0.470	0.224	0.353	0.189	0.196
GTSRB	51	-0.068	0.549	-0.051	0.745	-0.278	0.196	-0.321	0.078
CIFAR-100	91	0.105	0.352	0.065	0.627	0.013	0.902	0.032	0.784
		LSC v/s MS		DSC v/s MS		MDSC v/s MS		LSCD v/s MS	
		Corr.	p-value	Corr.	p-value	Corr.	p-value	Corr.	p-value
MNIST	45	0.368	0.039	0.545	0.039	0.594	0.039	0.613	0.039
SVHN	46	0.102	0.196	-0.008	0.824	0.485	0.039	0.866	0.039
GTSRB	51	0.363	0.039	0.094	0.118	0.415	0.039	0.957	0.039
CIFAR-100	91	0.284	0.039	0.427	0.039	0.527	0.039	0.943	0.039

The structural metrics show no meaningful relationship with the mutation score. Table 6 shows that the average Pearson correlation of NC, NBC, and SNAC across all datasets is 0.06, -0.055, and -0.003, respectively. These findings are also valid for k-MNC. Although some structural metrics show higher values on corner case datasets, this increase does not translate into a meaningful correlation with mutation score across diverse fault conditions. Consequently, the activation patterns of DNNs do not seem to be a reliable indicator of a dataset's effectiveness under various fault conditions.

Among the non-structural metrics, LSC and DSC exhibit only weak positive Pearson correlations with mutation score, with mean values of 0.279 and 0.264, respectively. Although LSC did not consistently differentiate between original test data and corner cases, it showed a weak positive correlation with mutation scores. This might be because the likelihood is estimated differently under different fault conditions. On the other hand, DSC shows a slightly lower correlation than LSC. This could be because it is more sensitive to outliers, which tend to be more common in larger networks and datasets. MDSC improves upon DSC by measuring deviation with respect to the class distribution rather than to individual training samples. This results in a more stable correlation pattern, with Pearson correlations of 0.594, 0.485, 0.415, and 0.527 on MNIST, SVHN, GTSRB, and CIFAR-100, respectively, corresponding to a mean Pearson correlation of 0.505.

Notably, LSCD shows the strongest and most stable performance, with high and statistically significant correlations across all datasets, ranging from 0.613 on MNIST to 0.943 on CIFAR-100. This consistency holds despite differences in dataset complexity, number of classes, and model architecture. This suggests that directly measuring the dispersion of test inputs around class centroids provides a stronger linear relationship with MS. In other words, while MDSC captures whether inputs deviate from the learned class distribution, LSCD more directly reflects how broadly test inputs

Table 7: Correlation analysis using **Spearman correlation coefficient** for structural and non-structural coverage criteria versus Mutation Score (MS) per dataset.

Dataset	No. of Mutant DNNs	NC v/s MS		k-MNC v/s MS		NBC v/s MS		SNAC v/s MS	
		Corr.	p-value	Corr.	p-value	Corr.	p-value	Corr.	p-value
MNIST	45	0.025	0.883	-0.099	0.558	-0.006	0.974	0.128	0.451
SVHN	46	0.063	0.726	0.080	0.659	0.283	0.111	0.221	0.217
GTSRB	51	-0.067	0.677	-0.053	0.741	-0.116	0.471	-0.223	0.161
CIFAR-100	91	0.178	0.109	0.036	0.746	-0.023	0.838	0.023	0.835
		LSC v/s MS		DSC v/s MS		MDSC v/s MS		LSCD v/s MS	
		Corr.	p-value	Corr.	p-value	Corr.	p-value	Corr.	p-value
MNIST	45	0.382	0.000	0.351	0.000	0.418	0.000	0.374	0.000
SVHN	46	0.072	0.254	-0.083	0.192	0.402	0.000	0.715	0.000
GTSRB	51	0.296	0.000	0.092	0.120	0.212	0.000	0.615	0.000
CIFAR-100	91	0.246	0.000	0.225	0.000	0.286	0.000	0.327	0.000

explore sparse and previously untested regions of the latent space, which appears to align more closely with fault-revealing capability. On average, LSCD shows a strong positive correlation of 0.844 with the mutation score (the highest among all studied test adequacy metrics) across all datasets.

Table 7 complements this analysis using Spearman rank correlation, which evaluates whether the relative ordering of test datasets induced by each metric is preserved with respect to mutation score. This is important because a useful proxy for mutation score should not only correlate numerically with MS, but should also preserve its ranking of dataset quality.

The structural metrics again show negligible rank correlation with mutation score. In contrast, the non-structural metrics showed a consistent trend in relative performance. LSC, DSC, MDSC, and LSCD achieve mean Spearman correlation coefficients of 0.249, 0.146, 0.329, and 0.507, respectively, across all datasets. Thus, LSCD yields the highest average rank correlation overall, indicating that it best preserves the relative ordering of datasets with respect to mutation score.

A closer look reveals an interesting distinction between MDSC and LSCD. On MNIST, MDSC attains the highest Spearman correlation among the non-structural metrics (0.418), slightly exceeding LSCD (0.374). This may suggest that covariance-aware deviation can be effective on simpler datasets with relatively well-behaved latent structure. However, as dataset complexity increases, the advantage of LSCD becomes more pronounced. On SVHN, GTSRB, and CIFAR-100, LSCD substantially outperforms MDSC in rank correlation, with Spearman coefficients of 0.715, 0.615, and 0.327, respectively, compared to 0.402, 0.212, and 0.286 for MDSC. These results indicate that while MDSC remains a strong distribution-aware baseline, LSCD provides a higher average rank association with MS.

To assess whether the observed relationship with mutation score depends on class conditioning, we additionally evaluated LSCD and MDSC using predicted classes.

Table 8: Robustness check using predicted-class variants of LSCD and MDSC. The table reports the mean Pearson and Spearman correlations with the mutation score across all datasets.

Metric	Mean Pearson Correlation	Mean Spearman Correlation
MDSC (using predicted classes)	0.505	0.329
LSCD (using predicted classes)	0.706	0.513

As shown in Table 8, the overall ranking remained unchanged: LSCD continued to show a stronger relationship with mutation score than MDSC on average when using predicted classes instead of ground-truth classes. In particular, LSCD achieved mean Pearson and Spearman correlations of 0.706 and 0.513, compared with 0.505 and 0.329 for MDSC. This suggests that LSCD remains strongly related to mutation score both when ground-truth labels are available and when they are not, consistent with its role as a direct measure of latent-space dispersion around class centroids.

Taken together, the primary Pearson and Spearman analyses and the additional robustness check using predicted class followed a clear performance hierarchy. LSC and DSC generally show weak and inconsistent correlations with mutation score across datasets. MDSC provides a more stable estimate than DSC and shows moderate correlation by capturing deviations with respect to the learned class distribution. However, LSCD consistently achieves the highest correlation values across both Pearson and Spearman analyses. In the Pearson correlation, LSCD exhibits strong positive correlations across all datasets (ranging from 0.613 to 0.943), indicating a strong linear relationship with mutation score. Similarly, under Spearman correlation, LSCD achieves the highest average rank correlation, suggesting that it most reliably preserves the ordering of dataset quality with respect to mutation score.

In terms of statistical significance, most non-structural metrics (LSC, DSC, MDSC, and LSCD) achieve p-values below the conventional thresholds of 0.05, indicating that their observed correlations are unlikely to be due to chance. LSCD is the only metric that consistently achieves both high correlation values and statistically significant results across all datasets in both Pearson and Spearman analyses. The similarity of p-values across metrics (e.g., constant values such as 0.039 or 0.000) likely arises from the limited sample size and the discrete nature of the underlying test statistics implemented using SciPy (Virtanen et al. 2020). Therefore, the distinguishing factor among the metrics is not solely statistical significance, but rather the magnitude and consistency of the correlation coefficients.

Answer to RQ1: Correlation Study

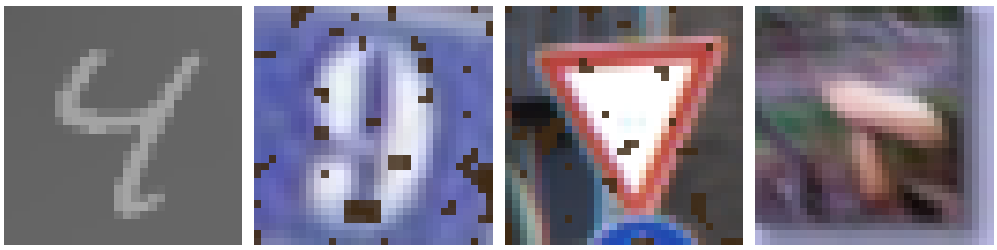
Structural coverage metrics show no meaningful correlation with mutation score. Among non-structural metrics, LSC and DSC exhibit weak correlations with MS. MDSC shows a more stable, moderate Pearson correlation of 0.505. LSCD achieves a significantly stronger correlation with MS under both Pearson (0.844) and Spearman (0.507) correlation coefficients, and consistently attains statistical significance across datasets.

5.1.1 $RQ_{1.1}$ (Corner Case Validity)

The inclusion of poor-quality corner case data poses a potential threat to the reliability of our correlation study in RQ1, as noisy data can also result in a higher MS and test adequacy metrics, especially latent-space-based metrics. We generated a substantial number of corner cases through fuzzing: 72,211 for MNIST, 71,551 for SVHN, 66,495 for GTSRB, and 467,358 for CIFAR-100, which we used to analyze the correlation between test adequacy metrics and MS. Given the scale of sampled corner cases, manually verifying the validity of these images is infeasible, and collecting equivalent real-world samples is also impractical. Therefore, we rely on fuzzing combined with an automated validator to assess the quality of the generated corner cases and ensure the robustness of our correlation study.

Some representative examples of the generated corner cases using fuzzing are shown in Figure 5. For instance, the original GTSRB "Yield" traffic sign is misclassified as "Traffic Signals" after applying spatter noise Figure 5c, while remaining clearly recognizable to a human observer. Similarly, in other datasets, the digit 4 is misclassified as 8 after applying fog on MNIST, the digit 9 is misclassified as 4 after applying spatter noise on SVHN, and a mushroom is misclassified as a trout after applying motion blur on CIFAR-100. These examples illustrate that the fuzzing campaign produces realistic and semantically meaningful corner cases that expose misclassification behavior.

To systematically verify this at scale, all generated corner cases were assessed using our automated validator, and only those deemed valid were retained for the correlation study in RQ1. Table 9 illustrates the autoencoder-validity results for corner case images generated using fuzzing. The table reports the number of valid and invalid images along with the validity rate. Across all datasets, the average validity rate is 99.50%, with very few corner cases reported as invalid, which supports the findings of the correlation study discussed in RQ1.



(a) MNIST (b) SVHN (c) GTSRB (d) CIFAR-100

Fig. 5: Sample Corner-case images generated using fuzzing used for our correlation analysis in RQ1.

Table 9: Answer to RQ1.1: Corner Case validity assessment using SelfOracle (Stocco et al. 2020).

Dataset	Coverage Criteria	Corner Case Data Evaluation Metrics			
		Total No. of Images	No. of Valid Images	No. of Invalid Images	Validity (%)
MNIST	NC	14,129	14,070	59	99.58
	k-MNC	12,868	12,779	89	99.30
	NBC	14,006	13,946	60	99.57
	LSC	15,610	15,508	102	99.34
	DSC	15,598	15,464	134	99.14
SVHN	NC	14,390	14,307	83	99.42
	k-MNC	14,216	14,126	90	99.36
	NBC	14,339	14,256	83	99.42
	LSC	14,306	14,243	63	99.55
	DSC	14,300	14,224	76	99.46
GTSRB	NC	13,327	13,305	22	99.83
	k-MNC	14,476	14,451	25	99.82
	NBC	14,249	14,216	33	99.76
	LSC	12,116	12,087	29	99.76
	DSC	12,327	12,297	30	99.75
CIFAR-100	NC	90,933	90,284	649	99.28
	k-MNC	95,790	95,220	570	99.40
	NBC	85,063	84,563	500	99.41
	LSC	97,323	96,745	578	99.40
	DSC	98,249	97,655	594	99.39
Average	-	-	-	-	**99.50**

Answer to RQ1.1: Corner Case Validity

The identified corner case images used to study correlation exhibit a high validity rate for all datasets (99.50% on average), according to a state-of-the-art automated input validator.

5.2 RQ_2 (Computational Time Required)

We used an Ubuntu 22.04 machine with an AMD Ryzen 2920X 12-Core Processor, NVIDIA GeForce RTX 2070 SUPER, and 128GB RAM in our experiments. We monitor the end-to-end computational time required for all test adequacy metrics calculations across all mutant models and data splits. For DSC, we computed computational time under both single and multi-thread settings.

Among the structural coverage metrics, NC demonstrated the highest computational efficiency. Its asymptotic complexity is $\mathcal{O}(M \cdot T_{test})$, where M is the total number of neurons in the DNN under test and T_{test} is the number of test inputs. This is attributed to its simple methodology, which verifies whether a neuron's activation

value exceeds a predefined threshold. As shown in Table 10, NC required only 4 minutes for the MNIST dataset due to LeNet-5's simpler architecture, whereas the deeper VGG-16 architecture used for CIFAR-100 increased the runtime to 7 hours and 23 minutes. In contrast, k-MNC proved to be the most computationally intensive, with a complexity of $\mathcal{O}(M \cdot T_{test} \cdot k)$. This increase is due to the requirement of binning each activation into one of k predefined sections. Consequently, k-MNC runtimes ranged from approximately 40 minutes for MNIST to over 3 days for CIFAR-100. In contrast to the k-MNC, NBC monitors both the upper- and lower-boundary corner cases that fall outside the k-sections established during training. SNAC simplifies this by focusing exclusively on upper-boundary activations, theoretically requiring only half as many boundary-checking operations as NBC. As presented in Table 10, NBC takes 35 minutes while SNAC takes only 14 minutes in the MNIST dataset. This trend is similar across all evaluated datasets.

Unlike structural coverage criteria, non-structural criteria are based on latent space vectors extracted from the final output layer of DNN under test. The computation of LSC requires performing Kernel Density Estimation (KDE) across all mutant models, which carries an asymptotic complexity of $\mathcal{O}(D \cdot T_{train} \cdot T_{test})$, where D represents the latent vector dimensionality, and T_{train} and T_{test} represent the sizes of the training and test sets, respectively. The LSC calculation requires approximately 15 hours across all datasets. We can see from Table 10 that LSC computation for SVHN and GTSRB takes 2 hours 18 minutes and 1 hour 20 minutes, respectively. This is because the SVHN training dataset is larger than the GTSRB dataset. The original DSC implementation (Kim et al. 2019) employed single-threaded computation, calculating image-wise DSC by measuring distance against the entire training set using Equation 7 and Equation 8. As shown in Table 10, single-threaded execution to compute DSC takes the most computational time among all test adequacy metrics, ranging to 325 hours for CIFAR-100. The multi-threaded implementation significantly optimized DSC computation, providing an average speedup of 17.7 times over the original single-threaded approach (Weiss et al. 2021). Despite parallelization, both DSC implementations have the same $\mathcal{O}(D \cdot T_{train} \cdot T_{test})$ complexity as LSC, since they must perform exhaustive distance comparisons between the training and test distributions. This is evident from Table 10 that even multi-threaded DSC calculations take 41 hours 36 minutes for datasets like CIFAR-100.

LSCD is designed to be simplistic and computationally more efficient by streamlining calculations within the latent space. Unlike LSC and DSC, it neither requires KDE estimation across all models nor distance calculation to each training data latent space vector. It only involves distance calculations to a single class center, reducing the complexity to $\mathcal{O}(D \cdot T_{test})$. By calculating distances only to the class centroid rather than to each training vector, the vectorized implementation allows LSCD computation across all mutants and datasets in just under 1 hour. It took only 41 seconds for the MNIST dataset and just 38 minutes for the more complex CIFAR-100 dataset. MDSC exhibits comparable computational efficiency, as it also relies on distance calculations based on centroids and a covariance matrix. As shown in Table 10, both LSCD and MDSC consistently achieve the fastest execution times across all datasets. This highlights that both metrics are computationally well-suited for large-scale evaluation.

Table 10: Answer to RQ2: Runtime comparison (hours:minutes:seconds) for calculating traditional test adequacy criteria and LSCD. DSC calculations are performed in both single and multi-thread implementations. Colors indicate relative performance: (green = fastest (LSCD and MDSC), gray = second-fastest metric (best-performing among existing baselines), and red = the slowest execution per dataset).

Dataset	LSCD	MDSC	NC	LSC	DSC (MT)	SNAC	NBC	k-MNC	DSC (ST)
MNIST	00:00:41	00:00:58	00:04:14	01:11:36	00:53:16	00:14:50	00:35:55	00:40:15	24:02:48
SVHN	00:00:51	00:01:11	00:16:45	02:18:40	01:22:36	01:55:44	02:36:49	03:42:09	36:53:25
GTSRB	00:02:19	00:02:25	00:18:20	01:20:21	02:36:09	02:47:38	03:18:24	05:14:15	23:05:33
CIFAR-100	00:38:12	00:43:49	07:23:36	11:45:49	41:36:46	93:09:00	126:32:05	225:03:24	325:51:24

However, a key difference lies in their memory requirements. MDSC requires the estimation and storage of a covariance matrix, which scales with both the latent space dimensionality and the number of output classes. As model complexity and output dimensionality increase, this storage overhead grows significantly. In contrast, LSCD only requires storing class centroids, making its memory footprint substantially lower and independent of covariance estimation.

Their efficiency is particularly notable when compared to traditional structural metrics, as they maintain their computational efficiency even as architectural depth and dataset complexity increase. While NC remains the most efficient among structural metrics, it is still hindered by the architecture complexity. As computational time increases with deeper architectures like VGG-16, it takes over 7 hours to calculate on CIFAR-100. In contrast, though non-structural metrics like LSC and DSC are significantly faster than intensive structural criteria like k-MNC, NBC, or SNAC, they remain an order of magnitude slower than LSCD and MDSC. They demonstrate the best computational performance, achieving an average speedup of approximately 10.4 times over NC, the most efficient structural metric. It also reduces the overhead of k-MNC from several days to under an hour. Overall, LSCD and MDSC both provide highly efficient computation compared to traditional structural and other surprise-based metrics. However, LSCD offers a simpler and more scalable alternative due to its reduced storage requirements and absence of covariance-based computations.

Answer to RQ2: Computational Time Required

LSCD and MDSC both demonstrate a substantial improvement in computational efficiency over all other evaluated test adequacy criteria. Across the studied datasets, they are approx. 10.4 times faster than NC, the most efficient among structural criteria, and approximately 78.6 times faster than LSC, which is the fastest among non-structural criteria.

5.3 Discussion

It is important to note that LSCD is not expressed on a fixed 0–100% scale, unlike other coverage metrics. Instead, it directly represents the magnitude of class dispersion

in the latent space of a trained DNN. A key reason for this is the interplay between the complexity of DNN architecture and the number of output classes. For example, MNIST with 10 output classes and simpler architectures, such as LeNet-5, shows LSCD values up to 58.15. SVHN also has 10 classes, but its spatial transformer network produces a more compact, structured latent space, yielding smaller LSCD values (up to 10.74). As the number of classes increases, even with similar architectures, the latent space must be divided into more class-specific regions, naturally leading to larger LSCD values. This is evident in the GTSRB, where corner-case data results in LSCD values reaching 188.73. Conversely, deeper and more expressive architectures can counterbalance this effect by learning more compact latent representations. This explains why VGG-16 on CIFAR-100 shows LSCD values up to 45 for corner case data despite having 100 output classes. Overall, these observations demonstrate that LSCD naturally adapts to both the architecture and the dataset characteristics. It provides a more interpretable and continuous measure of how far test inputs deviate from learned class regions. In particular, LSCD avoids the need for discretization into buckets, as required by metrics such as LSC, DSC, and MDSC, thereby eliminating sensitivity to hyperparameters such as bucket size.

Latent space-based metrics also offer practical advantages over structural coverage criteria, which require full access to the DNN's internal architecture. DSC suffers from high computational overhead and sensitivity to outliers because it relies on pairwise distances, whereas LSC relies on density estimation, which varies across different fault conditions. Thus, among latent-space metrics, MDSC and LSCD both offer computationally efficient, stable, and robust alternatives to all other criteria. MDSC captures deviations from the learned distribution by incorporating a covariance matrix, yielding a stable, distribution-aware measure of surprise. LSCD complements this perspective by directly quantifying geometric dispersion with respect to class centroids, without covariance normalization. This distinction leads to differences in interpretability: MDSC reflects how unusual an input is relative to the learned distribution, whereas LSCD reflects how broadly test inputs explore the latent space. Taken together, these results indicate that LSCD and MDSC provide complementary insights into latent-space behavior. This combination enables a more comprehensive evaluation of test dataset quality, particularly in identifying inputs that explore sparse and previously untested regions of the latent space.

This interpretation is further supported by the additional robustness check using predicted classes for LSCD and MDSC calculations. LSCD remains more strongly related to mutation score than MDSC on average. This suggests that the usefulness of LSCD is preserved both when ground-truth labels are available and when they are not.

The relevance of post-training MOs is still an ongoing research problem (Jahangirova and Tonella 2020; Abbasishahkoo et al. 2024), as they directly modify the internal structure of a trained DNN and are less representative of realistic faults in DNNs. Therefore, we used only pre-training MOs in our study. Importantly, our correlation analysis shows that LSCD maintains a meaningful correlation with mutation score, highlighting its potential as a practical proxy for MS in assessing test dataset quality. This enables a reduction in reliance on computationally expensive mutation testing while still supporting effective evaluation of a test dataset's ability to reveal faults.

5.4 Threats to Validity

5.4.1 Internal validity

The increase in various test adequacy metrics depends on the transformation techniques used during mutations in CGF. We have carefully chosen the domain-specific transformations that could occur in real-world situations and are suitable for DNNs in computer vision tasks. These metric scores could change if other transformations are applied.

Another potential threat to validity arises from variability in wall-clock runtime caused by system-level factors, such as disk I/O latency and file caching. In particular, DNN and dataset loading may introduce a warm-up bias, whereby later mutants benefit from cached resources and appear to execute faster than earlier ones. To mitigate this concern, we complement empirical runtime measurements with asymptotic complexity analysis, ensuring that our conclusions reflect algorithmic efficiency rather than incidental implementation or hardware artifacts. Although environmental conditions may influence absolute execution times, the observed reductions in theoretical computational complexity provide a hardware-independent, robust explanation for the efficiency gains observed in our experiments.

5.4.2 External validity / Generalizability

In our evaluations, we focused on image classification task, which could pose a threat to the generalizability of our results. We intentionally selected well-established benchmark architectures, such as LeNet, Spatial Transformer Networks, and VGG-16 (Lecun et al. 1998; Hira 2026; Simonyan and Zisserman 2014), and widely used datasets of varying complexity and real-world relevance in order to provide a reliable basis for studying the relationship between test adequacy metrics and mutation score in this setting. We deliberately avoided datasets like CIFAR-10 due to their low-quality images and ambiguous labeling (Karpathy; Toneva et al. 2019). This could hinder the recognizability of corner cases generated by fuzzing and correlation study. We did not include ImageNet (Deng et al. 2009) in our evaluations. Prior works commonly evaluate only subsets of ImageNet classes due to the substantial computational and storage demands of full-scale ImageNet training and testing (Humbatova et al. 2021; Ma et al. 2018; Xie et al. 2019; Zohdinasab et al. 2026). Our experiments utilize CIFAR-100, with all 100 diverse object categories, which provides a widely used and computationally feasible proxy for studying high-class-count image classification. It captures the scale and complexity required to evaluate adequacy metrics without the prohibitive cost of ImageNet.

While our evaluation focused on image classification, we are confident that the underlying principles of traditional adequacy metrics and our proposed LSCD metric are not tied to any specific architecture or dataset. Broader validation across additional architectures, datasets, and application domains remains an important direction for future work, with the main adaptation being the selection of an appropriate latent-space representation for the task under consideration.

6 Related Work

6.1 Test Adequacy Criteria for DNNs

Various test adequacy criteria and test input generators based on them have been proposed for DNNs (Kim et al. 2019; Pei et al. 2019; Ma et al. 2018; Xie et al. 2019; Zhang et al. 2018; Tian et al. 2018). Structural coverage criteria such as neuron coverage have been shown less useful for test data quality measurement (Harel-Canada et al. 2020). On the other hand, latent space-based test adequacy metrics such as surprise coverage enjoy wider applicability. Likelihood-based and distance-based surprise coverage quantify how "surprising" a test input is relative to the training data distribution (Kim et al. 2019). Mahalanobis distance-based surprise coverage further extends this idea by incorporating class-wise distribution statistics, including covariance, enabling a distribution-aware measurement of deviation in latent space (Kim et al. 2023). As studied by Weiss et al. in (Weiss et al. 2021), surprise coverage has wide applicability and has already been cited in more than 105 scientific papers. Due to its computational complexity, refinement and computationally efficient versions of DSC have also been proposed (Weiss et al. 2021). However, most existing studies on surprise coverage primarily evaluate its effectiveness using adversarial inputs to analyze model behavior beyond original test data. Although this provides useful insights into robustness, such inputs may not fully reflect the characteristics of real-world input transformation.

Mani et al. (Mani et al. 2019) proposed the Centroid Positioning (CP) metric to describe the total number of points located outside the train class cluster threshold by a radius value. The calculation of the CP metric is highly sensitive to these threshold values. In our preliminary experiments, we found that changing the radius value changes the CP values constantly. Also, the determination of the radius value is not described in detail in the original work (Mani et al. 2019). In this work, we use LSCD to operationalize direct latent-space dispersion and study whether this signal approximates mutation score more effectively than existing adequacy metrics under the studied settings. Unlike LSC, DSC, and MDSC, LSCD does not rely on density estimation, pairwise comparisons, or covariance-based normalization, and therefore provides a direct and interpretable characterization of latent-space structure. Rather than focusing solely on deviation from the distribution, LSCD emphasizes how broadly test inputs explore class regions.

To the best of our knowledge, prior work has not systematically studied the relationship between test adequacy metrics and mutation score. Such analysis is important because the mutation score provides a direct measure of a dataset's ability to reveal faults in DNNs. Understanding how existing metrics correlate with mutation scores can help identify computationally efficient alternatives for assessing test dataset quality.

Our study differs from prior work in two key aspects. First, we evaluate LSCD and existing metrics using corner-case data generated through fuzzing, which better reflects realistic scenarios compared to adversarial inputs (Zhang et al. 2022; Li et al. 2022). Second, we analyze the correlation between these metrics and mutation score, providing empirical evidence of their effectiveness as proxies to MS for test dataset quality. Overall, this work positions LSCD not as a replacement for existing latent-space metrics such as MDSC, but as a complementary measure that emphasizes

dispersion rather than distributional deviation. Together, these perspectives enable a more comprehensive understanding of test dataset quality in DNNs.

6.2 Mutation Testing for DNNs

Mutation testing is well-studied for DL systems inspired by mutation testing in software systems (Tufano et al. 2020; Hu et al. 2019; Humbatova et al. 2021). DeepMutation, DeepMutation++, and DeepCrime (Tufano et al. 2020; Hu et al. 2019; Humbatova et al. 2021) successfully proposed frameworks to create mutant models of trained DNNs and compute MS. The Mutation score is used to rank various test suites based on their ability to reveal the faults in mutant models. DeepMutation++ (Hu et al. 2019) uses post-training MOs, which lack realism to real-world applications as studied in (Humbatova et al. 2021; Abbasishahkoo et al. 2024). To ensure that the findings in our study were relevant to real-world DNN behavior, we focused on realistic pre-training MOs. In DeepCrime (Humbatova et al. 2021), the authors derived the pre-training MOs based on the real-world fault categories and computed MS. The choice of MOs to inject the faults and the availability of various formulas to compute MS make it more challenging to adapt it for practical uses. On top of that, the computationally expensive re-training steps involved in generating mutant model raise a serious question about their usefulness. An effective alternative to Mutation Score (MS) needs to be explored before it can be reliably used in other functions, such as when TEASMA uses it to predict fault detection rates (Abbasishahkoo et al. 2024). Thus, we studied the correlation of MS with other test adequacy metrics. We introduced and evaluated LSCD as a promising approximation to MS for test dataset quality assessment. In our study, we found that LSCD shows a stronger association with MS than the evaluated metrics under the studied settings. The taxonomy of real faults in DL systems (Humbatova et al. 2020) describes 92 unique faults arising from various sources leading to the faulty behaviors of DNNs. We chose 15 unique faults and created mutant models using corresponding pre-training MOs for our study.

6.3 Test Data Generation for DNNs

The existence of a labeled and robust test oracle is very limiting. This has fostered the development of many automated test data generation techniques for DNN like DeepHunter (Xie et al. 2019), DeepXplore (Pei et al. 2019), DeepGauge (Ma et al. 2018), and XMutant (Chen et al. 2025). These methods are used in industry (Zhang et al. 2022; Li et al. 2022) to test the robustness properties of DNN models. This means identifying corner cases in which DNN exhibits poor performance. In our work, we use the CGF technique similar to DeepHunter to sample the corner cases for test data quality enhancement. While DeepHunter transforms the input image by using metamorphic relation strategies based on affine and pixel value transformations (Xie et al. 2019), in our paper, we use domain-specific transformations such as weather transformations (e.g., fog, rain), image transformations (e.g., saturate, contrast), blurring effects (e.g., defocus blur, gaussian blur) (Hendrycks and Dietterich 2019) which are more realistic for the image classification task. Also, we reported the validity of sampled corner

cases assessed by the SOTA automated validator which is missing in DeepHunter (Xie et al. 2019).

7 Conclusion and Future Work

Test dataset quality plays a pivotal role in trusting the performance of DNNs, as reflected in evaluation metrics. It is common to rely on two main types of criteria for test dataset quality assessment: test adequacy metrics and mutation score. Identifying an efficient and effective criterion for test dataset quality assessment remains an open research topic. Traditional test adequacy metrics do not require generating mutant models, unlike the mutation score, but their ability to quantify test datasets effectively across various fault conditions remains unexplored. In contrast, mutation score measures test dataset quality based on their fault-revealing nature. Despite its promising insights, the practical application of mutation scores is limited due to the significant computational overhead required to generate mutants.

In this paper, we evaluated various test adequacy metrics to determine whether they could serve as potential approximations of MS. Our results show that traditional structural metrics struggle to capture meaningful differences between original and corner-case data distributions and exhibit no meaningful correlation with MS. In contrast, latent-space based metrics provide a more informative measure of test data quality. Among these, both MDSC and LSCD demonstrate a meaningful relationship with mutation score. MDSC captures distribution-aware deviations in the latent space and shows moderate correlation across studied datasets. On the other hand, LSCD consistently achieves a stronger correlation with mutation score, reflected in Pearson and Spearman coefficients of 0.844 and 0.507, respectively, while remaining computationally efficient. LSCD and MDSC's computational time across 233 mutant models and all datasets was approx. 10.4 times faster than NC (the most efficient structural coverage metric) and 78.6 times faster than LSC (the most efficient non-structural coverage metric). Unlike traditional metrics, LSCD does not require full access to the DNN under test and is free from covariance estimation and bucket-based discretization like MDSC, making it simple to apply across different architectures and datasets.

Overall, our results suggest that LSCD and MDSC offer complementary perspectives on test dataset quality. While MDSC measures how unusual inputs are with respect to the learned distribution, LSCD directly captures how broadly test inputs explore latent space. Together, these properties highlight their potential as a practical and efficient approximation to mutation testing, reducing reliance on computationally expensive mutant generation while still enabling effective assessment of a dataset's fault-revealing capability.

In future work, we aim to evaluate the effectiveness of LSCD and compare it with conventional fuzzing techniques for test generation. In this study, we refrained from using LSCD to guide fuzzing, as employing the same criteria for both test generation and evaluation would influence the results. We will investigate using LSCD to detect biases in DNN and drive retraining strategies that mitigate such problems. Finally, we aim to investigate the capability of LSCD to characterize boundaries in the latent space of DNNs (Weißl et al. 2025; Riccio and Tonella 2020; Merabishvili et al. 2026),

and foundation models such as diffusion models (Weißl et al. 2026; Guo et al. 2018), and LLMs (Sorokin et al. 2026; Giebisch et al.; Yoon et al. 2025).

Funding

The authors did not receive support from any organization for the submitted work.

Competing Interests

Alexander Pretschner serves as an Editorial Board Member of the Software Quality Journal. In accordance with the journal's editorial policies, he will not be involved in the editorial handling, reviewer selection, or decision-making process for this manuscript. The remaining authors declare that they have no competing interests.

Author Contributions

Vivek V. Vekariya conceived the study, designed and implemented the approach, conducted the experiments, analyzed the results, and prepared the original manuscript draft. Mojdeh Golagha, Andrea Stocco, and Alexander Pretschner contributed to study design, interpretation of results, manuscript review and revision, and supervision. All authors reviewed and approved the final manuscript.

Data Availability

The software artifacts and the data used for analysis are publicly available (2026).

References

An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. (2015)

Abbasishahkoo, A., Dadkhah, M., Briand, L., Lin, D.: TEASMA: A Practical Methodology for Test Adequacy Assessment of Deep Neural Networks (2024). https://arxiv.org/abs/2308.01311

Bengio, Y., Mesnil, G., Dauphin, Y., Rifai, S.: Better Mixing via Deep Representations (2012). https://arxiv.org/abs/1207.4404

Bojarski, M., Testa, D.D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to end learning for self-driving cars. CoRR **abs/1604.07316** (2016)

Chen, X., Biagiola, M., Riccio, V., d'Amorim, M., Stocco, A.: XMutant: XAI-based Fuzzing for Deep Learning Systems. Empirical Software Engineering (2025)

Choi, S.C., Wette, R.: Maximum likelihood estimation of the parameters of the gamma distribution and their bias. Technometrics **11**(4), 683–690 (1969) https://doi.org/10.1080/00401706.1969.10490731

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)

Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)

Goodfellow, I.J., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, MA, USA (2016). http://www.deeplearningbook.org

Giebisch, R., Friedl, K.E., Sorokin, L., Stocco, A.: Automated factual benchmarking for in-car conversational systems using large language models. In: Proceedings of the 36th IEEE Intelligent Vehicles Symposium. IV '25. IEEE

Guo, J., Jiang, Y., Zhao, Y., Chen, Q., Sun, J.: 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, pp. 739–743 (2018)

Harel-Canada, F., Wang, L., Gulzar, M.A., Gu, Q., Kim, M.: 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. ESEC/FSE 2020, pp. 851–862. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3368089.3409754 . https://doi.org/10.1145/3368089.3409754

Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: International Conference on Learning Representations (ICLR) (2019)

Hira, P.: pytorch-implementation-of-gtsrb-classification-challenge. https://github.com/poojahira/gtsrb-pytorch. Accessed: 2026 (2026)

Humbatova, N., Jahangirova, G., Bavota, G., Riccio, V., Stocco, A., Tonella, P.: Taxonomy of real faults in deep learning systems. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. ICSE '20, pp. 1110–1121. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3377811.3380395 . https://doi.org/10.1145/3377811.3380395

Humbatova, N., Jahangirova, G., Tonella, P.: Deepcrime: mutation testing of deep learning systems based on real faults. In: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA 2021, pp. 67–78. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3460319.3464825 . https://doi.org/10.1145/3460319.3464825

Hu, Q., Ma, L., Xie, X., Yu, B., Liu, Y., Zhao, J.: Deepmutation++: A mutation testing framework for deep learning systems. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1158–1161 (2019). https://doi.org/10.1109/ASE.2019.00126

Huang, S.H., Papernot, N., Goodfellow, I.J., Duan, Y., Abbeel, P.: Adversarial attacks on neural network policies. CoRR **abs/1702.02284** (2017) 1702.02284

Julian, K.D., Kochenderfer, M.J., Owen, M.P.: Deep neural network compression for aircraft collision avoidance systems. CoRR **abs/1810.04240** (2018) arXiv:1810.04240

Jahangirova, G., Tonella, P.: An empirical evaluation of mutation operators for deep learning systems. In: 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), pp. 74–84 (2020). https://doi.org/10.1109/ICST46399.2020.00018

Karpathy, A.: Lessons learned from manually classifying CIFAR-10. https://karpathy.github.io/2011/04/27/manually-classifying-cifar10/

Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (2017)

Kim, J., Feldt, R., Yoo, S.: Guiding deep learning system testing using surprise adequacy. In: Proceedings of the 41st International Conference on Software Engineering. ICSE '19, pp. 1039–1049. IEEE Press, Piscataway, NJ, USA (2019). https://doi.org/10.1109/ICSE.2019.00108 . https://doi.org/10.1109/ICSE.2019.00108

Kim, J., Feldt, R., Yoo, S.: Evaluating surprise adequacy for deep learning system testing. ACM Trans. Softw. Eng. Methodol. **32**(2) (2023) https://doi.org/10.1145/3546947

Kaminwar, S., Goschenhofer, J., Thomas, J., Thon, I., Bischl, B.: Structured Verification of Machine Learning Models in Industrial Settings. Big Data, Mary Ann Liebert, Inc. (2021). https://doi.org/10.1089/big.2021.0112 . https://doi.org/10.1089/big.2021.0112

Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report (2009)

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998) https://doi.org/10.1109/5.726791

Li, S., Guo, J., Lou, J.-G., Fan, M., Liu, T., Zhang, D.: Testing machine learning systems in industry: An empirical study. In: 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 263–272 (2022). https://doi.org/10.1109/ICSE-SEIP55303.2022.9793981

Lambertenghi, S.C., Leonhard, H., Stocco, A.: Benchmarking image perturbations for testing automated driving assistance systems. In: Proceedings of the IEEE 18th IEEE International Conference on Software Testing, Verification and Validation. ICST '25, p. 12. IEEE. https://arxiv.org/abs/2501.12269

Maryam, Biagiola, M., Stocco, A., Riccio, V.: Benchmarking Generative AI Models for Deep Learning Test Input Generation. In: Proceedings of 18th IEEE International Conference on Software Testing, Verification and Validation. ICST '25, p. 12. IEEE. https://arxiv.org/abs/2412.17652

Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., Liu, Y., Zhao, J., Wang, Y.: DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems, pp. 120–131. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3238147.3238202

Mani, S., Sankaran, A., Tamilselvam, S., Sethi, A.: Coverage testing of deep learning models using dataset characterization. CoRR **abs/1911.07309** (2019) 1911.07309

Merabishvili, G., Weissl, O., Stocco, A.: Latent regularization in generative test input generation. In: Proceedings of the 48th International Conference on Software Engineering Workshops (2026)

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.: Reading digits in natural images with unsupervised feature learning. (2011). https://api.semanticscholar.org/CorpusID:16852518

Pei, K., Cao, Y., Yang, J., Jana, S.: Deepxplore: Automated whitebox testing of deep learning systems. Commun. ACM **62**(11), 137–145 (2019) https://doi.org/10.1145/3361566

Panichella, A., Liem, C.C.S.: What Are We Really Testing in Mutation Testing for Machine Learning? A Critical Reflection (2021). https://arxiv.org/abs/2103.01341

Replication Package. https://github.com/tum-i4/latent_space_class_dispersion (2026)

Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., Tonella, P.: Testing Machine Learning based Systems: A Systematic Mapping. Empirical Software Engineering (2020)

Riccio, V., Tonella, P.: Model-based exploration of the frontier of behaviours for deep learning system testing. ESEC/FSE, pp. 876–888 (2020)

Riccio, V., Tonella, P.: When and why test generators for deep learning produce invalid inputs: an empirical study. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1161–1173 (2023). https://doi.org/10.1109/ICSE48619.2023.00104

Spearman, C.: The proof and measurement of association between two things. The American Journal of Psychology **15**(1), 72–101 (1904)

Stocco, A., Pulfer, B., Tonella, P.: Mind the Gap! A Study on the Transferability of Virtual vs Physical-world Testing of Autonomous Driving Systems. IEEE Transactions on Software Engineering (2022) https://doi.org/10.1109/TSE.2022.3202311

Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In: IEEE International Joint Conference on Neural Networks, pp. 1453–1460 (2011)

Sorokin, L., Vasilev, I., Friedl, K.E., Stocco, A.: STELLAR: A Search-Based Testing Framework for Large Language Model Applications (2026). https://arxiv.org/abs/2601.00497

Stocco, A., Weiss, M., Calzana, M., Tonella, P.: Misbehaviour prediction for autonomous driving systems. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. ICSE '20, pp. 359–371. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3377811.3380353 . https://doi.org/10.1145/3377811.3380353

Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

Tufano, M., Kimko, J., Wang, S., Watson, C., Bavota, G., Di Penta, M., Poshyvanyk, D.: Deepmutation: a neural mutation tool. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings. ICSE '20, pp. 29–32. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3377812.3382146 . https://doi.org/10.1145/3377812.3382146

Tian, Y., Pei, K., Jana, S., Ray, B.: Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th International Conference on Software Engineering. ICSE '18, pp. 303–314. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3180155.3180220 . https://doi.org/10.1145/3180155.3180220

Toneva, M., Sordoni, A., Combes, R.T., Trischler, A., Bengio, Y., Gordon, G.J.: An Empirical Study of Example Forgetting during Deep Neural Network Learning (2019). https://arxiv.org/abs/1812.05159

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy

1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020) https://doi.org/10.1038/s41592-019-0686-2

Weißl, O., Abdellatif, A., Chen, X., Merabishvili, G., Riccio, V., Kacianka, S., Stocco, A.: Targeted deep learning system boundary testing. ACM Transactions on Software Engineering and Methodology (2025)

Weiss, M., Chakraborty, R., Tonella, P.: A Review and Refinement of Surprise Adequacy . In: 2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest), pp. 17–24. IEEE Computer Society, Los Alamitos, CA, USA (2021). https://doi.ieeecomputersociety.org/10.1109/DeepTest52559.2021.00009

Wikipedia: Pearson correlation coefficient — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient. [Online; accessed 17-July-2024] (2024)

Wand, M.P., Jones, M.C.: Kernel Smoothing, 1st edn. Chapman and Hall/CRC, ??? (1994). https://doi.org/10.1201/b14876

Weißl, O., Riccio, V., Kacianka, S., Stocco, A.: HyperNet-Adaptation for Diffusion-Based Test Case Generation (2026). https://arxiv.org/abs/2601.15041

Wong, A., Wang, X.Y., Hryniowski, A.: How much can we really trust you? towards simple, interpretable trust quantification metrics for deep neural networks. CoRR **abs/2009.05835** (2020) 2009.05835

Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J., See, S.: DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks, pp. 146–157. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3293882.3330579

Yan, M., Chen, J., Cao, X., Wu, Z., Kang, Y., Wang, Z.: Revisiting deep neural network test coverage from the test effectiveness perspective. Journal of Software: Evolution and Process **36**(4), 2561 (2024) https://doi.org/10.1002/smr.2561 https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2561

Yoon, J., Kim, S., Feldt, R., Yoo, S.: Clotho: Measuring Task-Specific Pre-Generation Test Adequacy for LLM Inputs (2025). https://arxiv.org/abs/2509.17314

Yuan, Y., Pang, Q., Wang, S.: You can't see the forest for its trees: Assessing deep neural network testing via neural coverage. CoRR **abs/2112.01955** (2021) 2112.01955

Zhao, J., Feng, Q., Wu, P., Lupu, R.A., Wilke, R.A., Wells, Q.S., Denny, J.C., Wei, W.-Q.: Learning from longitudinal data in electronic health record and genetic data to improve cardiovascular event prediction. Scientific Reports **9**(1), 717 (2019) https:

//doi.org/10.1038/s41598-018-36745-x

Zhang, J.M., Harman, M., Ma, L., Liu, Y.: Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering **48**(1), 1–36 (2022) https://doi.org/10.1109/TSE.2019.2962027

Zohdinasab, T., Riccio, V., Tonella, P.: Automated feature extraction for testing deep learning systems through illumination search. Software Testing, Verification and Reliability **36**(1-2), 70019 (2026) https://doi.org/10.1002/stvr.70019 https://onlinelibrary.wiley.com/doi/pdf/10.1002/stvr.70019

Zhang, Q., Wang, H., Lu, H., Won, D., Yoon, S.W.: Medical image synthesis with generative adversarial networks for tissue recognition. In: 2018 IEEE International Conference on Healthcare Informatics (ICHI) (2018). IEEE

Zhang, Z., Wang, Y., Yao, Y., Wang, Z., Huang, Z.: A fine-grained evaluation of mutation operators to boost mutation testing for deep learning systems. Empirical Software Engineering **30**(63) (2025) https://doi.org/10.1007/s10664-025-10613-5

Zhang, M., Zhang, Y., Zhang, L., Liu, C., Khurshid, S.: Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 132–142 (2018). https://doi.org/10.1145/3238147.3238187