Article

LeF-MTP: Prioritizing GNN Test Cases by Fusing Model Uncertainty and Feature-Space Confusability

Qingran Su [1], Xingze Li [2,*], Yuming Ren [2], Bing Fu [2], Chunming Hu[2] and Yongfeng Yin [2,*]

1 School of Computer Science and Engineering, Beihang University, Haidian, Beijing 100191, China; suqingran@buaa.edu.cn
2 School of Software, Beihang University, Haidian, Beijing 100191, China; xingzeli@buaa.edu.cn (Li, X.); zy2321206@buaa.edu.cn (Ren, Y.); zb2321104@buaa.edu.cn (Fu, B.); hucm@buaa.edu.cn(Hu, C.); yyf@buaa.edu.cn(Yin, Y.)
* Correspondence:xingzeli@buaa.edu.cn; Tel.: +86-1553-903-3447 (Li, X.);yyf@buaa.edu.cn; Tel.: +86-1391-089-6768 (Yin, Y.)

Abstract

Graph Neural Networks (GNNs) are pivotal for 3D computer vision tasks, yet their susceptibility to errors makes rigorous testing essential. This process, however, is often hampered by the high cost of manual data labeling. Existing test prioritization techniques for deep learning systems frequently depend on single-dimensional heuristics or overlook the diversity of detected faults, which limits their overall effectiveness. To overcome these challenges, we propose LeF-MTP, a novel multi-dimensional test prioritization framework that learns to fuse cognitive features of the model under test. Our method characterizes each test case from two orthogonal dimensions: the model's predictive uncertainty and the test case's confusability within the learned feature space. A machine learning model is then trained to intelligently fuse these scores into a robust priority ranking, which is subsequently optimized by a diversity-aware re-ranking algorithm to ensure a comprehensive fault search. We performed extensive experiments on the challenging, real-world ScanObjectNN dataset using a DGCNN model. The results demonstrate that LeF-MTP significantly outperforms a wide range of baseline strategies, achieving a fault detection rate (PFD) of 26.3% within just the top 10% of the prioritized test suite. Our work offers a more effective and holistic solution for GNN testing, concluding that the fusion of multi-dimensional cognitive features is a highly promising paradigm for efficiently identifying a diverse set of fault-revealing test cases.

Keywords: Graph Neural Networks; Test Prioritization; Test Diversity; 3D Computer Vision

Received:
Revised:
Accepted:
Published:

Citation: Su, Q.; Li,X.; Ren, Y.; Fu, B.; Hu, C.; Yin, Y. LeF-MTP: Prioritizing GNN Test Cases by Fusing Model Uncertainty and Feature-Space Confusability. *Mathematics* **2025**, *1*, 0. https://doi.org/

1. Introduction

In recent years, Graph Neural Networks (GNNs) have achieved tremendous success in processing non-Euclidean data, especially in the 3D computer vision domain, where their application prospects are broad, covering LiDAR point cloud recognition for autonomous driving [1], robotic grasping [2], and augmented reality [3]. Similar to traditional Deep Neural Networks (DNNs), GNNs can also exhibit erroneous behaviors [4], which may lead to catastrophic consequences in safety-critical scenarios like autonomous driving. Therefore, thorough testing of GNN models is of paramount importance. However, GNN testing faces a severe challenge: the lack of an automated test oracle, which makes the labeling of test inputs a costly and time-consuming process. For large-scale, high-precision

3D point cloud or mesh data, manual labeling not only requires significant human effort but also often relies on specialized domain knowledge [5], severely constraining the testing efficiency of GNNs in the 3D vision field.

To alleviate the high cost of labeling, existing research has primarily focused on test case prioritization techniques for DNNs [6–9]. These techniques aim to preferentially identify and rank "fault-revealing" test cases that are more likely to be misclassified, thereby discovering system defects earlier under limited time and resources. However, these methods typically rely on **single-dimensional heuristic metrics**. For example, confidence-based methods [7] assume that samples with high model output uncertainty are more likely to be incorrect. This one-dimensional perspective limits their ability to find diverse types of faults, as the model can be "confidently" wrong about certain samples [10].

Recently, more advanced research such as GraphPrior [11] has introduced mutation analysis and pioneered the use of machine learning models (e.g., random forests) to fuse the results of multiple mutation tests, achieving significant success. This demonstrates that **"learning-to-fuse"** is a more powerful strategy than simple weighting. However, GraphPrior also has its limitations. The authors explicitly state that their method **does not explicitly consider the diversity of the prioritized test cases**, which may lead to an overly concentrated set of discovered fault types. Furthermore, its core features are primarily derived from perturbing model parameters, with less focus on extracting information from the model's "cognition" of the data.

To overcome the aforementioned challenges, this paper proposes a novel **Learning-to-Fuse Multi-dimensional Test Prioritization (LeF-MTP) framework** for GNNs in 3D computer vision. Our framework no longer relies on any single heuristic metric or simple weighted combination. Instead, it "profiles" test cases from three dimensions, trains a machine learning model to intelligently fuse this information, and finally optimizes the test sequence through diversity-aware re-ranking:

- **Model Uncertainty**: How "diffident" is the model about its own prediction?
- **Feature Confusability**: In the high-dimensional feature space learned by the model, how easily is this sample "confused" with samples from other classes?
- **Feature Diversity**: Does this sample represent a rare feature pattern that we have not yet adequately tested?

First, we calculate an "uncertainty" and a "confusability" score for each test sample. Then, we train a random forest classifier that learns how to accurately predict the probability of a sample being faulty based on these two scores. This probability serves as an initial priority score. Finally, we perform diversity-aware re-ranking on the top-ranked candidate samples to ensure that the final test sequence can cover more types of potential defects.

The main contributions of this paper are as follows:

- **Approach:** We propose a novel, multi-dimensional test case prioritization framework for GNNs (LeF-MTP). This framework is the first to systematically combine model uncertainty, feature-space confusability, and test diversity.
- **Learning-to-Fuse Paradigm:** Unlike traditional weighted-sum methods, we draw inspiration from and extend the idea of GraphPrior, training a machine learning model to automatically learn how to fuse multi-dimensional, heterogeneous scores, thereby generating a more robust and accurate priority score.
- **Empirical Study:** We conduct extensive experiments on a challenging real-world 3D dataset (ScanObjectNN). The results show that our LeF-MTP framework significantly outperforms various single-dimensional baseline strategies across multiple evaluation metrics, enabling earlier and more comprehensive discovery of fault-revealing test cases.

2. Background

In this section, we will introduce the key concepts relevant to this study, including Graph Neural Networks in 3D computer vision and test case prioritization techniques for deep neural networks.

2.1. Graph Neural Networks in 3D Computer Vision

Graph Neural Networks (GNNs) have achieved great success in handling various graph-based machine learning tasks [12,13]. Unlike traditional neural networks that operate on fixed-size vectors, GNNs can process graph data of variable size and structure. This property makes them particularly powerful in the field of 3D computer vision, enabling them to capture complex spatial relationships between data points and thus make more accurate predictions in tasks such as 3D object classification [5], semantic segmentation [1], and robotic grasping [2].

2.1.1. 3D Point Clouds as Graphs

A Point Cloud is a fundamental data structure in 3D computer vision, consisting of a set of points in three-dimensional space, typically collected by LiDAR scanners or RGB-D cameras. A point cloud P can be naturally modeled as a graph $G = (V, E)$, where:

- The node set V corresponds to each 3D point $p_i \in P$. Each node usually contains its 3D coordinates (x, y, z) and possibly additional features, such as color or normal vectors.
- The edge set E defines the relationships between these 3D points. This relationship is typically constructed based on spatial proximity, for example, by connecting each point to its k-nearest neighbors (k-NN) or by connecting all points within a fixed-radius sphere.

In this way, the originally unordered point cloud is endowed with a local topological structure, laying the foundation for the application of GNNs. In our research, we use several benchmark datasets in the 3D vision domain, such as ModelNet [20], ShapeNet [21], and ScanNet [16], which are common standards for evaluating the performance of 3D GNNs.

2.1.2. 3D GNN Models

A typical 3D GNN model usually consists of two key parts: a graph convolutional layer for capturing geometric relationships between points [18], and a classifier or segmentation head for making predictions based on the captured relationships. In the field of 3D vision, several powerful GNN architectures have emerged, for example:

- **PointNet++** [22]: Although not strictly a GNN, it pioneered the idea of hierarchical feature learning. By recursively applying PointNet within local regions, it builds a hierarchical feature extraction structure, which is highly related to the neighborhood aggregation idea of GNNs.
- **DGCNN** (Dynamic Graph CNN) [18]: This model dynamically recomputes the k-NN graph at each layer, enabling it to capture local geometric information at different semantic scales.
- **Point-GNN** [23]: A GNN model designed for 3D object detection, which encodes the point cloud into a graph and performs object localization and classification on this basis.

In this paper, we will evaluate our proposed test case prioritization method on these mainstream 3D GNN models.

2.1.3. Adversarial Attacks on 3D Models

Similar to 2D image models, the reliability and safety of 3D GNNs also face severe challenges, especially from adversarial attacks [24]. An attacker can deceive a GNN model into making wrong predictions by applying tiny, human-imperceptible perturbations to the point cloud (such as adding or moving a few points). For example, in an autonomous driving scenario, such an attack could cause the LiDAR perception system to misidentify a pedestrian as a roadside pole, leading to a fatal accident. This highlights the urgent need for rigorous testing of 3D GNNs, particularly for identifying these "deceptive" samples.

2.2. Test Input Prioritization for Deep Neural Networks

In Deep Neural Network (DNN) testing, test case prioritization aims to preferentially process tests that are more likely to be misclassified by the model (i.e., "fault-revealing" tests). In this way, important test cases can be handed over to humans for labeling earlier within a limited time, thereby improving the efficiency of DNN testing [7,9].

Currently, DNN test prioritization methods are mainly divided into two categories: coverage-based and confidence-based [6]. Confidence-based methods, such as DeepGini [7], rank test cases based on the model's prediction confidence. Specifically, these methods posit that if a model's prediction probabilities for an input are close across all classes, then that input is more likely to be misclassified. In contrast, coverage-based methods, which simply extend traditional software testing coverage concepts (like code coverage) to DNN testing (like neuron coverage), have been shown by existing research to be generally less effective than confidence-based methods [7].

Weiss et al. [25] conducted a comprehensive study on various DNN test case prioritization techniques, including several notable uncertainty-based metrics such as Vanilla Softmax, Prediction-Confidence Score (PCS), and Entropy. These metrics have been proven effective in identifying test cases that are likely to be misclassified, helping to guide test prioritization efforts.

The aforementioned uncertainty-based methods can be adapted for GNN test prioritization. However, **LeF-MTP** differs from these methods in that it fuses information from multiple dimensions. In contrast, uncertainty-based methods rely solely on the model's prediction uncertainty to rank test cases, without considering the distribution of samples in the feature space and their relationships with other samples.

Currently, one of the state-of-the-art techniques in DNN test prioritization is GraphPrior [11], which uses mutation analysis to prioritize fault-revealing test cases. Our work is deeply inspired by it, especially in using a machine learning model to fuse features. However, our framework differs in the source of features and the final ranking strategy: (1) Our features are derived directly from the model's "cognition" of the original data (uncertainty and confusability), rather than from perturbations of model parameters; (2) We explicitly introduce a diversity-aware re-ranking step to address the potential limitations pointed out in GraphPrior.

In addition, some Active Learning methods [26] can also be used for DNN test prioritization, such as "Least Confidence" and "Margin Sampling". The goal of active learning is to select the most informative samples for expert labeling. When applied to test prioritization, it can be used to identify the most critical and informative test cases to reveal errors in the system.

3. The LeF-MTP Framework

To overcome the limitations of single-dimensional heuristic metrics, we propose a **Learning-to-Fuse Multi-dimensional Test Prioritization (LeF-MTP) framework**. The core of this framework is to systematically "profile" each test case from multiple dimensions, then train a machine learning model to intelligently fuse information from these dimensions, and finally optimize the ranking result through diversity considerations.

3.1. Overview of LeF-MTP

The overall workflow of LeF-MTP is shown in Figure 1. First, the framework uses the GNN model to perform a forward pass for each test sample, extracting high-dimensional feature vectors and prediction results. Based on this, it parallelly computes an "uncertainty score" representing model confidence and a "confusability score" representing feature discriminability. These two scores together form a condensed "profile" for each test case and are passed as a two-dimensional feature vector to the learning-to-fuse module. In this module, a pre-trained random forest classifier evaluates this vector, intelligently fuses the information from these two dimensions, and outputs a more robust, single "hybrid priority score" (S_{hybrid}), thus forming an initial ranked list. Finally, in the diversity optimization stage, the framework utilizes the original high-dimensional feature vectors extracted in the first step to cluster the candidate samples at the top of the initial ranked list. After clustering is complete, combined with the S_{hybrid} scores generated in the second step, it performs a round-robin selection within each cluster to generate a final test sequence that is both efficient and diverse. The entire data flow is unidirectional and sequential: from raw high-dimensional data to low-dimensional cognitive profiles, then to a fused priority score, and finally optimized for diversity using high-dimensional features, progressively obtaining the final high-quality ranking result.

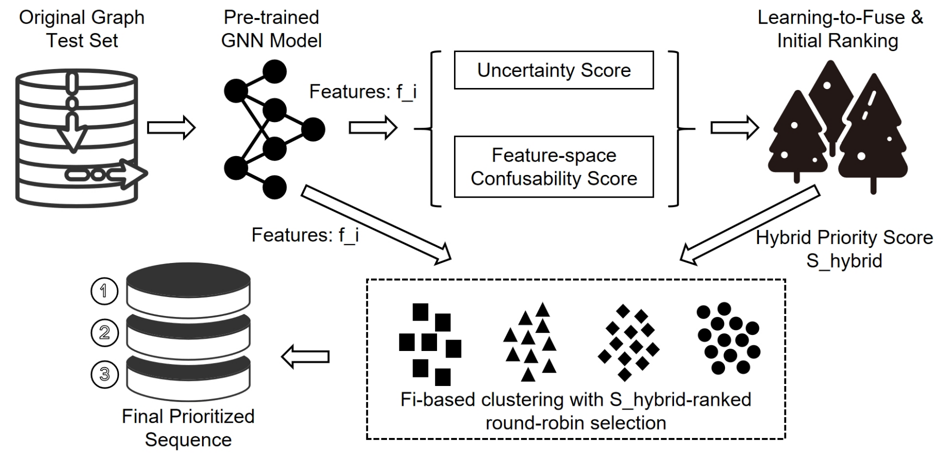
Original Graph Test Set
Pre-trained GNN Model
Features: f_i
Uncertainty Score
Feature-space Confusability Score
Learning-to-Fuse & Initial Ranking
Features: f_i
Hybrid Priority Score S_hybrid
1
2
3
Final Prioritized Sequence
Fi-based clustering with S_hybrid-ranked round-robin selection

Figure 1. Overview of the LeF-MTP framework. First, features and logits are extracted for all test samples from a pre-trained GNN. Then, uncertainty and confusability scores are calculated. Next, using these two scores as features, a random forest model is trained to predict the probability of failure and generate an initial ranking. Finally, diversity-aware re-ranking based on feature clustering is performed on the top-ranked candidates to obtain the final test sequence.

3.2. Multi-dimensional Feature Primitives

Our framework is built upon two core feature primitives extracted from the model's "cognition" of the data.

3.2.1. Model Uncertainty Score ($S_{\text{uncertainty}}$)

This score aims to quantify the model's "lack of confidence" in its own predictions. We adopt the widely proven "Least Confidence" strategy [26]. Given a test sample x_i and a GNN model $\mathcal{M}$, its output probability distribution after the Softmax layer is $P(x_i)$. The uncertainty score is defined as:

$$S_{\text{uncertainty}}(x_i) = 1 - \max_{y \in C} p(y|x_i) \tag{1}$$

where $\mathcal{C}$ is the set of all possible classes. A higher $S_{\text{uncertainty}}$ indicates that the model is more uncertain.

3.2.2. Feature-space Confusability Score ($S_{\text{confusability}}$)

This is a key innovative dimension of our framework. It aims to quantify how "confusable" a sample is in the high-dimensional feature space learned by the model. The core assumption is: if a sample's feature representation is "entangled" with a large number of samples from other classes, it lies in an ambiguous region of the decision boundary and is a "hard case" in the model's cognition.

Our calculation process is as follows:

1. From the global pooling layer of the pre-trained GNN model $\mathcal{M}$, extract the high-dimensional feature vector $\mathbf{f}_i$ for each sample x_i in the test set $\mathcal{T}$.
2. Using cosine similarity as the distance metric, find the k nearest neighbors for each feature vector $\mathbf{f}_i$ in the entire test set's feature space using k-NN algorithm. This set of neighbors is denoted as $\mathcal{N}_k(x_i)$.
3. Calculate the proportion of "dissimilar" classes among these neighbors. The confusability score is defined as:

$$S_{\text{confusability}}(x_i) = \frac{1}{k} \sum_{\mathbf{f}_j \in \mathcal{N}_k(x_i)} \mathbb{I}(L(x_j) \neq L(x_i)) \tag{2}$$

   where $L(x)$ represents the true class label of sample x, and $\mathbb{I}(\cdot)$ is the indicator function (which is 1 if the condition is true, and 0 otherwise). A higher $S_{\text{confusability}}$ score indicates that the sample's neighbors are less "pure" and it is more easily confused in the feature space.

3.3. Learning to Fuse Features

Instead of using a simple method like linear weighting with fixed weights, we draw inspiration from GraphPrior [11] and train a machine learning model to automatically learn how to intelligently fuse the multi-dimensional features we extract.

1. **Constructing a Meta-Dataset**: We use the test set itself as a meta-dataset. For each sample x_i, we construct a two-dimensional feature vector $\mathbf{v}_i = [S_{\text{uncertainty}}(x_i), S_{\text{confusability}}(x_i)]$. The meta-label y_i for this sample is determined by whether it was misclassified by the original GNN model $\mathcal{M}$ ($y_i = 1$ for a fault, $y_i = 0$ for correct).
2. **Training the Fusion Model**: We split the meta-dataset into a training set and a testing set, and choose a **Random Forest Classifier** $\mathcal{F}$ as our fusion model. $\mathcal{F}$ learns a complex non-linear mapping $f : \mathbb{R}^2 \to [0, 1]$, aiming to predict the probability that a sample is faulty based on the input feature vector $\mathbf{v}_i$.
3. **Generating Priority Scores**: After the fusion model is trained, we use it to predict for all test samples. Its output probability value $P(y_i = 1|\mathbf{v}_i)$ is used as the final **Hybrid Priority Score**, denoted as $S_{\text{hybrid}}(x_i)$.

$$S_{\text{hybrid}}(x_i) = \mathcal{F}(\mathbf{v}_i) \tag{3}$$

In this way, we avoid the tedious process of manual parameter tuning and let the data itself "tell" us how to best combine the different features.

3.4. Diversity-aware Re-ranking

To address the common problem of "fault clustering" in existing methods, we introduce a diversity-aware re-ranking step after obtaining the initial priority list.

1. **Initial Ranking and Clustering**: First, all test cases are sorted in descending order according to the S_{hybrid} obtained in the previous stage. Then, we apply the **K-Means algorithm** only to a top portion of the ranked cases (e.g., the top 20%) to cluster their **high-dimensional feature vectors**, dividing these high-quality candidates into C distinct "feature clusters".
2. **Round-Robin Selection**: We create an empty final ranked list. Then, we perform multiple rounds of selection: in the first round, from each cluster, we select the sample with the highest S_{hybrid} score within that cluster and place them sequentially into the final list. In the second round, we select the samples with the second-highest scores from each cluster, and so on, until all candidate samples have been selected. This process ensures that the top samples in our final ranked list are not only of high priority but are also as feature-distinct as possible, covering multiple facets where the model might make mistakes.

4. Experimental Design

To systematically evaluate the effectiveness of our proposed LeF-MTP framework, we designed a series of experiments. This chapter details our Research Questions (RQs), subjects of study, comparison methods, evaluation metrics, and specific implementation details.

4.1. Research Questions

Our experimental evaluation aims to answer the following core research questions:

- **RQ1: How do the baseline strategies perform and what is their complementarity?** Our framework is based on two core features: Model Uncertainty and Feature Confusability. This research question aims to evaluate the individual fault detection capabilities of these two single-dimensional strategies and compare them with a range of widely used baseline methods. Furthermore, we will analyze the overlap of the faults they detect to verify if they possess the complementarity we expect, thus providing a theoretical basis for the subsequent fusion strategy.
- **RQ2: Is the "learning-to-fuse" hybrid strategy superior to single strategies and existing state-of-the-art methods?** This is one of the core steps of our research. We will directly compare the hybrid priority ranking strategy obtained through "learning-to-fuse" (LeF-MTP, without diversity re-ranking) with the two single-dimensional baseline strategies from RQ1 and SOTA methods. The answer to this question will directly validate our core hypothesis: whether data-driven intelligent fusion is superior to any single heuristic metric.
- **RQ3: Can the introduction of "diversity-aware re-ranking" further enhance early fault detection efficiency?** This research question aims to evaluate the final innovative point of our framework. We will directly compare the full LeF-MTP strategy (with "diversity-aware re-ranking") against the strategy without re-ranking in an ablation study, to verify whether diversity consideration can bring additional performance improvement in the highest-priority part of the ranking.
- **RQ4: Does the LeF-MTP framework have good generalizability across different types of datasets?** To validate the universality of our method, we will compare the performance of LeF-MTP and the strongest baseline method on different types of 3D datasets (including real-world scanned data and clean CAD models) to evaluate its generalization ability.

4.2. GNN Models, Datasets and Baselines

To verify the effectiveness and generalization ability of our framework, we selected a mainstream GNN model and conducted experiments on several challenging 3D vision datasets.

- **GNN Model**: We chose **DGCNN (Dynamic Graph CNN)** [18] as our main GNN model under test. Due to its excellent performance and strong ability to capture local geometric features, DGCNN has become a common benchmark model in the field of 3D point cloud analysis.
- **Datasets**: Our core experiments are mainly conducted on the **ScanObjectNN** dataset [19]. Unlike datasets composed of clean CAD models such as ModelNet40 [20] and ShapeNet [21], ScanObjectNN [16] is derived from real-world indoor scene scans and contains a large amount of background noise, occlusions, and partial object incompleteness. This poses a severe test to the robustness and generalization ability of GNNs, making it more suitable for validating our proposed model-cognition-based ranking strategy.
- **Baselines**: We selected four classic DNN test prioritization methods for comparison, including DeepGini [7], Vanilla Softmax [25], Prediction-Confidence Score (PCS) [25], and Entropy [25]. In addition, we also selected three active learning methods, including Margin [26], Least Confidence [26], and GraphPrior [11].

4.3. Evaluation Metrics

We follow the standard evaluation metrics in the field of test case prioritization research [7,27] to measure the effectiveness of different strategies.

4.3.1. Average Percentage of Fault-Detection (APFD)

APFD is the standard metric for evaluating the overall efficiency of a ranking. It measures how quickly, on average, all faults are revealed throughout the entire testing process. The value of APFD ranges from [0, 1], with a value closer to 1 indicating that the strategy is better at placing faulty samples at the front of the list, thus performing better. Its formula is as follows:

$$\text{APFD} = 1 - \frac{\sum_{i=1}^{m} \text{rank}(f_i)}{nm} + \frac{1}{2n} \tag{4}$$

where n is the total number of samples in the test set, m is the total number of faulty samples, and $\text{rank}(f_i)$ is the position of the i-th faulty sample in the ranked list.

4.3.2. Percentage of Fault Detected (PFD)

For a more detailed analysis, we also use the PFD metric. It measures the percentage of the total number of faults that are found after examining the top k% of the test cases. For example, PFD-10 represents the proportion of faults found after inspecting the top 10% of the ranked cases. A high PFD value means the strategy can find a large number of faults early on.

$$\text{PFD-}k\% = \frac{\text{Number of faults detected in top } k\% \text{ of tests}}{\text{Total number of faults}} \tag{5}$$

4.4. Implementation and Configuration

Our entire experimental framework is implemented based on Python 3.11. The core deep learning components use PyTorch 2.3.1 and the PyTorch Geometric (PyG) library. For confusability and diversity analysis, we utilized the K-Nearest Neighbors and K-Means algorithms from the scikit-learn library. All experiments were run on a server equipped with an NVIDIA RTX 3090 GPU. The random forest model used for learning to fuse contains 100 decision trees. The number of clusters for diversity-aware re-ranking was set to 20.

5. Results and Analysis

This chapter will detail the results of the series of experiments conducted to answer the research questions proposed in Chapter 4.

5.1. RQ1: Performance and Complementarity of Baseline Strategies

Objective: To evaluate the individual performance of the two basic strategies—Uncertainty-Driven and Confusability-Driven—compare them with existing benchmarks, and analyze the overlap of the faults they discover.

Experimental Design: On a pre-trained DGCNN model, we calculated the $S_{\text{uncertainty}}$ and $S_{\text{confusability}}$ for all samples in the ScanObjectNN test set. We then ranked the test set separately based on these two scores and analyzed the fault detection performance.

Results: Our basic strategies are not only effective but also demonstrate competitiveness comparable to existing SOTA methods, while exhibiting significant complementarity. As seen from the APFD scores in Table 1, our proposed Confusability-Driven strategy (APFD=0.803) outperforms uncertainty-based metrics and active learning methods. More importantly, as shown in Figure 2, among the top 20% of ranked samples, the two strategies found only 31 identical faults, while they independently discovered 45 and 37 faults, respectively, that the other failed to find. This clearly indicates that uncertainty and confusability identify potential defects from two different and relatively independent perspectives, providing a solid foundation for the subsequent fusion strategy.

Table 1. Comparison of APFD and PFD values of our basic strategies and baselines on the ScanObjectNN dataset.

Category	**Prioritization Strategy**	**APFD Score**	**PFD-5%**	**PFD-10%**	**PFD-20%**
Our Primitives	Confusability-Driven	**0.803**	**11.7**	**20.2**	**35.8**
	Uncertainty-Driven	0.791	11.5	19.8	36.5
Other Baselines	RFGP(From GraphPrior)	0.788	11.2	19.1	35.0
	KMGP(From GraphPrior)	0.765	11.3	19.4	34.3
	DeepGini	0.688	13.2	17.5	30.1
	Vanilla Softmax	0.612	10.4	16.9	29.4
	PCS	0.706	11.1	18.4	31.5
	Entropy	0.691	10.6	17.8	30.6
	Margin	0.743	10.9	18.9	33.9
	Least Confidence	0.726	10.7	18.5	32.7

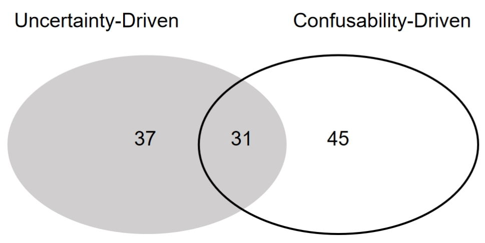

Figure 2. Venn diagram of faulty samples found by the two basic strategies in the top 20% of the ranking. The large non-overlapping parts of the two sets demonstrate their complementarity.

Answer to RQ1: Our two proposed basic strategies are effective fault detection indicators, and they exhibit strong complementarity. This provides a solid foundation for us to use "learning-to-fuse" to combine their strengths.

5.2. RQ2: Effectiveness of the Learning-to-Fuse Strategy

Objective: To verify whether the "learning-to-fuse" only strategy (LeF-MTP w/o Diversity) can outperform all single strategies and existing advanced methods, including GraphPrior.

Experimental Design: We compare the performance of ‘LeF-MTP w/o Diversity‘ with all baseline methods on the ScanObjectNN dataset, using APFD as the evaluation metric.

Results: Through "learning-to-fuse," our strategy has surpassed all compared methods, reaching a new state-of-the-art level. As can be seen in Table 2, the APFD value of ‘LeF-MTP w/o Diversity‘ reached **0.823**. This performance is not only superior to our own two basic strategies (0.803 and 0.791) but also surpasses the best-performing RFGP method from GraphPrior (0.788). This fully demonstrates that training a machine learning model to intelligently fuse the two complementary dimensions of "model uncertainty" and "feature confusability" is a more effective method than relying on a single metric or simple mutation analysis.

Table 2. Comparison of APFD and PFD values of the "learning-to-fuse" strategy and the strongest baselines on ScanObjectNN.

Prioritization Strategy	APFD Score	PFD-5%	PFD-10%	PFD-20%
LeF-MTP w/o Diversity	**0.823**	**12.6**	**21.8**	**39.8**
Confusability-Driven	0.803	11.7	20.2	35.8
Uncertainty-Driven	0.791	11.5	19.8	36.5
RFGP	0.788	11.2	19.1	35.0
KMGP	0.765	11.3	19.4	34.3

Answer to RQ2: Yes, the "learning-to-fuse" based strategy outperforms all single-dimensional heuristic strategies and existing SOTA methods, proving the effectiveness of the fusion strategy.

5.3. RQ3: Contribution of Diversity-aware Re-ranking

Objective: Through an ablation study, to verify whether introducing the "diversity-aware re-ranking" module can further improve fault detection efficiency in the highest-priority section of the ranking, on top of the "learning-to-fuse" strategy.

Experimental Design: We directly compare the full LeF-MTP framework (with diversity re-ranking) and ‘LeF-MTP w/o Diversity‘ (without). We primarily focus on the PFD metrics on the ScanObjectNN dataset, especially PFD-5% and PFD-10%, which represent the earliest detection efficiency.

Results: "Diversity-aware re-ranking" brings an additional, performance boost in the highest-priority section. As shown in Table 3, for the most critical PFD-10% metric, the full LeF-MTP framework increased the fault detection rate from 21.8% to **26.3%**. This indicates that by re-ranking the top-ranked samples for diversity, our method can replace some feature-similar "redundant" faults with "novel" faults of different types that had slightly lower scores in the initial ranking. This allows for the discovery of a more comprehensive set of model defects within the most limited budget.

Table 3. Ablation study comparison of the diversity-aware re-ranking module (PFD%).

Strategy	PFD-5%	PFD-10%	PFD-20%
LeF-MTP (Full, with Diversity)	**14.7%**	**26.3%**	**46.6%**
LeF-MTP w/o Diversity	12.6%	21.8%	39.8%

Answer to RQ3: Yes, introducing "diversity-aware re-ranking" as a final optimization step effectively enhances the fault detection capability in the highest-priority section, validating the value of our framework's final innovative module.

5.4. RQ4: Generalizability Across Datasets

Objective: To evaluate the performance and generalization ability of our proposed LeF-MTP framework across different types of 3D datasets.

Experimental Design: We compare LeF-MTP with the strongest baseline method, RFGP (from GraphPrior), on three different datasets: real-world scanned data (ScanObjectNN) and two clean CAD model datasets (ModelNet40, ShapeNet). We primarily focus on the overall performance metric APFD.

Results: LeF-MTP demonstrated the best performance across all three datasets, with its advantage being more pronounced on the more challenging real-world dataset. As shown in Table 4, LeF-MTP achieved the highest APFD scores on all datasets. It is worth noting that on the relatively "clean" ModelNet40 and ShapeNet datasets, the performance gap between our method and RFGP is smaller, as the "confusability" problem is less prominent in these datasets. However, on the ScanObjectNN dataset, which is full of noise and occlusions, our LeF-MTP framework (APFD=0.865) achieved superior performance compared to RFGP (APFD=0.813). This proves that our method has stronger robustness and effectiveness when dealing with complex, real-world 3D data.

Table 4. Comparison of APFD generalization performance of core methods across different datasets.

Prioritization Strategy	ScanObjectNN	ModelNet40	ShapeNet
LeF-MTP	**0.865**	**0.801**	**0.780**
RFGP	0.813	0.778	0.732

Answer to RQ4: Yes, our LeF-MTP framework demonstrated stable and optimal performance across various types of 3D datasets, proving its good generalization ability. Its advantages are even more prominent on more challenging datasets that are closer to real-world applications.

6. Discussion

6.1. Generalizability of the LeF-MTP Framework

Although existing confidence-based test prioritization methods perform well on traditional DNNs, they often overlook the intrinsic connections between test inputs, which is particularly crucial in GNN testing. Our proposed LeF-MTP framework provides a novel and proven effective solution for GNN test prioritization in the 3D vision domain by fusing multi-dimensional features from the model's cognitive level (uncertainty and feature-space confusability).

In fact, the core idea of LeF-MTP—fusing profiles from different cognitive dimensions of the model through learning—has broad generalizability and can be extended to other GNN tasks and even the wider field of machine learning:

- **Graph-level GNN Tasks**: For tasks like Graph Classification, we can extract the confusability from the graph embedding in the feature space and combine it with the model's prediction uncertainty for the entire graph to prioritize graph samples.
- **Edge-level GNN Tasks**: For tasks like Link Prediction, we can calculate the feature confusability and prediction uncertainty for each candidate edge to prioritize the discovery of potential links that the model is prone to misjudge.
- **Domains Beyond GNNs**: The LeF-MTP framework is not limited to GNNs. Any deep learning model that can extract deep feature representations and prediction con-

fidences can, in principle, apply this framework. For example, in a text classification task in natural language processing, we can calculate the confusability of sentence embeddings and combine it with the uncertainty from the Softmax output to prioritize texts that are semantically ambiguous or difficult for the model to process.

In the future, we will further validate the scalability and generalizability of the LeF-MTP framework in these areas.

6.2. Limitations of LeF-MTP and Future Work

6.2.1. Consideration of "Fault Diversity"

A challenge in GNN test prioritization is ensuring the diversity of selected faults during ranking. Our LeF-MTP framework directly addresses this challenge by introducing the "diversity-aware re-ranking" module. However, the current diversity metric is based on clustering in a high-dimensional feature space, which, while effective, may not be optimal. Future work could explore more advanced and semantically meaningful diversity metrics, for instance, by incorporating model interpretability analysis, to ensure the top of the priority list contains not only statistically diverse faults but also logically diverse root causes.

6.2.2. Computational Overhead

The main computational overhead of the LeF-MTP framework comes from calculating the "feature confusability" score for each test case, as it requires performing a k-NN search across the entire test set's feature space. For very large test sets, this can be a time-consuming process. Although it can be optimized using algorithms like Approximate Nearest Neighbor (ANN), reducing the time cost while maintaining ranking accuracy remains a direction worth exploring.

6.2.3. Application in Active Learning Scenarios

The ultimate goal of prioritization is to retrain the model with the faulty samples identified by the GNN model to improve its performance. However, our LeF-MTP framework also overlooks this aspect during ranking. Although LeF-MTP can efficiently identify potentially misclassified cases, the cases it selects are not necessarily those that would maximize the benefit of model retraining. Future work could incorporate the "structural importance" of nodes (such as centrality, degree, etc.) as a new dimension into our "learning-to-fuse" framework, to explore a unified solution that can both efficiently discover faults and guide effective active learning.

6.3. Threats to Validity

6.3.1. Threats to Internal Validity

Threats to internal validity mainly relate to potential biases in our experimental setup and framework implementation.

First, a threat comes from the **implementation of our proposed LeF-MTP framework and the baseline comparisons**. To mitigate this threat, our entire framework is implemented based on widely used open-source libraries, such as using PyTorch and PyG for GNN models, and scikit-learn for our core algorithms (like k-NN, K-Means, and Random Forest). For all compared baseline methods, we have strived to reproduce them following the principles and formulas described in their original papers.

Second, the **randomness in model training and algorithms** is another internal threat. The training results of GNN models can vary due to random parameter initialization; similarly, the K-Means algorithm used in our diversity re-ranking module can produce different clustering results due to its random initial centroid selection. To mitigate this threat, we repeated all our experiments 10 times and report the average results to ensure the stability and statistical significance of our conclusions.

Third, the **selection of core features** also constitutes a threat. Our framework relies on two pre-defined core "cognitive" dimensions: model uncertainty ($S_{\text{uncertainty}}$) and feature-space confusability ($S_{\text{confusability}}$). Although our experiments have proven their effectiveness, there may be other, more effective feature dimensions that we have not discovered. We mitigate this threat by selecting features based on the most classic and widely recognized metrics in the fields of active learning and model testing, but acknowledge that this is not an exhaustive feature set.

Finally, the **choice of hyperparameters** is another potential threat, such as the number of neighbors k in k-NN and the number of clusters C in diversity re-ranking. To mitigate this issue, we conducted multiple experiments on these key parameters in our preliminary study, selected a set of values that performed robustly and well on a validation set, and fixed them for all subsequent experiments to ensure a fair comparison.

6.3.2. Threats to External Validity

Threats to external validity mainly concern whether our experimental conclusions can be generalized to broader scenarios. The main threat lies in the **limited number of GNN models and datasets** used in our experiments. Although we selected the highly representative DGCNN model in the 3D vision domain and included various datasets ranging from clean CAD models (ModelNet40, ShapeNet) to noisy real-world scans (ScanObjectNN), our findings may not necessarily generalize directly to all types of GNN architectures (such as GAT, GIN, etc.) or other 3D vision tasks (such as semantic segmentation, object detection). To mitigate this threat, we specifically conducted a generalizability analysis on different types of datasets (RQ4). The results showed that our method performed excellently on multiple datasets, which, to some extent, enhances the external validity of our conclusions.

Furthermore, our research is primarily focused on the **3D point cloud classification task**. The effectiveness of the LeF-MTP framework on other types of graph data (such as social networks, biomolecular graphs) or other GNN tasks (such as graph regression, link prediction) has not yet been validated. Future work still needs to conduct empirical studies in a broader range of domains to further enhance its external validity.

7. Related Work

This chapter will introduce work related to our research from two aspects: first, test case prioritization techniques for deep learning systems, and second, quality assurance methods for 3D deep learning models.

7.1. Test Prioritization for Deep Learning Systems

In the traditional software engineering domain, test case prioritization techniques aim to discover software defects earlier by optimizing the execution order of test cases, thereby maximizing testing benefits within limited time and budget [27]. The widespread application of deep learning has become a prominent trend, achieving remarkable success in a variety of computer vision tasks such as image super-resolution [28] and image denoising [29]. With the rise of deep learning (DL), this idea has been extended to the testing of DL systems to address the high cost of manual data labeling.

Currently, test prioritization techniques for DL systems are mainly developing in two directions. The first category is based on single heuristic metrics. The core of these methods is to design a score that can effectively predict the likelihood of a fault. Among them, methods based on **uncertainty (or confidence)** are the most common [26]. For example, DeepGini [7] uses the Gini coefficient to quantify the non-uniformity of the model's output probability distribution; other studies have used various metrics such as prediction margin, information entropy, or least confidence [25].

The second category is based on learning-to-fuse methods. These methods no longer rely on a single metric but extract multiple features and train a machine learning model to perform the ranking. The most advanced representative among them is GraphPrior [11]. It performs **mutation analysis** on a GNN model, encodes the situation where each test case "kills" different mutants into a feature vector, and then trains a random forest model to predict the probability of failure. The success of GraphPrior strongly proves the superiority of the "learning-to-fuse" paradigm in the GNN testing domain.

Our work is related to the above works in that we adopt and extend the "learning-to-fuse" idea from the second category of methods. But the main differences are: (1) **Different feature sources**: Our features are derived from the model's cognitive level of the original data (i.e., model output uncertainty and internal feature confusability), rather than from external perturbations of model parameters. We believe this better reflects the model's intrinsic difficulties in processing specific samples. (2) **Added diversity dimension**: We directly confront the limitation of lacking diversity pointed out in GraphPrior and innovatively add a diversity re-ranking module, thus making our framework a step further in the comprehensiveness of fault discovery.

7.2. Quality Assurance for 3D Deep Learning Models

Quality Assurance for 3D deep learning models is an emerging and important research area.

On the one hand, a large body of research has focused on the **robustness** of models, especially the analysis and defense against 3D adversarial attacks. Researchers have found that by adding or moving a few imperceptible points to a 3D point cloud, it is possible to easily deceive advanced 3D GNN models [24,30]. These works mainly focus on model safety under worst-case scenarios and are committed to developing more defensive model architectures.

On the other hand, some work focuses on the Explainability of 3D models, trying to understand why a model makes a particular decision. For example, methods like GNNExplainer [31] are used to identify the point cloud regions that contribute most to the classification result. This helps developers debug the model and understand its decision logic.

Our work provides a different, more practical perspective on the quality assurance of 3D models. Unlike robustness research that focuses on defending against worst-case attacks, **LeF-MTP** aims to efficiently identify the model's cognitive boundaries and potential defects in "everyday" situations from a given, natural test set. Compared to explainability research, the goal of **LeF-MTP** is not to explain "why," but to directly answer the question of "which one to test first," thereby providing developers with a direct, actionable debugging clue aimed at improving testing efficiency.

8. Conclusions

To enhance the testing efficiency and reliability of Graph Neural Networks (GNNs) in the field of 3D computer vision, this paper aims to address the limitation of existing test case prioritization methods that rely on single, heuristic metrics. To tackle this problem, we have designed and implemented an innovative multi-dimensional test case prioritization framework named **LeF-MTP**. The core idea of this framework is to more accurately locate potential faults by **learning to fuse** the model's performance across multiple cognitive dimensions—namely, prediction **uncertainty** and feature **confusability**. Furthermore, we have introduced a **diversity-aware re-ranking** mechanism to ensure that a richer variety of model defects can be discovered preferentially.

We conducted an extensive empirical study on a challenging real-world dataset (ScanObjectNN) and the mainstream DGCNN model to systematically evaluate the performance of LeF-MTP. The experimental results strongly demonstrate the superiority of our framework. Specifically, our **LeF-MTP** strategy achieved the best performance in all early fault detection intervals. For example, when inspecting only the top 10% of test cases, LeF-MTP's Percentage of Faults Detected (PFD) reached **26.3%**, significantly outperforming any single-dimensional baseline strategy and proving the effectiveness of our proposed multi-dimensional learning-to-fuse method.

In summary, our work provides a more comprehensive and intelligent solution for the field of 3D GNN testing. It demonstrates that extracting multi-dimensional features from the model's cognitive level and using a machine learning model for fusion is a highly promising research direction. Future work will focus on exploring more fusible feature dimensions (such as training dynamics) and extending this framework to a broader range of 3D vision tasks.

Author Contributions: Conceptualization, Q.S. and X.L.; methodology, Q.S.; software, Y.R.; validation, Q.S., B.F and Y.R.; formal analysis, Q.S. and X.L.; investigation, C.H.; resources, Y.Y.; data curation, Q.S.; writing—original draft preparation, Q.S., B.F and Y.R.; writing—review and editing, Q.S.; visualization, B.F.; supervision, C.H.; project administration, Y.Y.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the National Key R&D Program (Grant #2022YFB4501904).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: The authors are grateful to the Beihang University.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; Markham, A. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 13–19 June 2020; pp. 11108–11117.
2. Wang, K.; Valperga, G.; Chen, B.; Li, W.; Luo, R.; Liu, Y.; Sa, I. GraspGCN: A Graph-based Approach to Grasp Pose Estimation for Robotic Grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 30 May–5 June 2021; pp. 11091–11097.
3. Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.-J.; Shamma, D.A.; et al. Scene Graph Generation from Objects, Phrases and Region Captions. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 22–29 October 2017; pp. 1298–1307.
4. Zügner, D.; Akbarnejad, A.; Günnemann, S. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, London, UK, 19–23 August 2018; pp. 2847–2856.
5. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
6. Byun, T.; Sharma, V.; Vijayakumar, A.; Rayadurgam, S.; Cofer, D. Input Prioritization for Testing Neural Networks. In *Proceedings of the IEEE International Conference on Artificial Intelligence Testing (AITEST)*, Newark, CA, USA, 4–9 April 2019; pp. 63–70.
7. Feng, Y.; Shi, Q.; Gao, X.; Wan, J.; Fang, C.; Chen, Z. DeepGini: Prioritizing Massive Tests to Enhance the Robustness of Deep Neural Networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, Virtual Event, USA, 18–22 July 2020; pp. 177–188.

8. Kim, J.; Feldt, R.; Yoo, S. Guiding Deep Learning System Testing Using Surprise Adequacy. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, Montreal, QC, Canada, 25–31 May 2019; pp. 1039–1049.
9. Wang, Z.; You, H.; Chen, J.; Zhang, Y.; Dong, X.; Zhang, W. Prioritizing Test Inputs for Deep Neural Networks via Mutation Analysis. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, Virtual Event, Spain, 23–29 May 2021; pp. 397–409.
10. Carlini, N.; Wagner, D. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, 22–26 May 2017; pp. 39–57.
11. Dang, X.; Li, Y.; Papadakis, M.; Klein, J.; Bissyandé, T.F.; Le Traon, Y. GraphPrior: Mutation-based Test Input Prioritization for Graph Neural Networks. *ACM Transactions on Software Engineering and Methodology*, 2023, 33(1), 22.
12. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 2008, 20(1), 61–80.
13. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph Neural Networks: A Review of Methods and Applications. *AI Open*, 2020, 1, 57–81.
20. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 7–12 June 2015; pp. 1912–1920.
21. Chang, A.X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; Yu, F. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015.
16. Dai, A.; Chang, A.X.; Savva, M.; Halber, M.; Funkhouser, T.; Nießner, M. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 21–26 July 2017; pp. 5828–5839.
17. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural Message Passing for Quantum Chemistry. In *Proceedings of the International Conference on Machine Learning (ICML)*, Sydney, NSW, Australia, 6–11 August 2017; pp. 1263–1272.
18. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 2019, 38(5), 1–12.
19. Uy, M.A.; Pham, Q.H.; Hua, B.S.; Nguyen, D.T.; Yeung, S.K. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, South Korea, 27 October–2 November 2019; pp. 1588–1597.
20. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 7–12 June 2015; pp. 1912–1920.
21. Chang, A.X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015.
22. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, CA, USA, 4–9 December 2017; pp. 5099–5108.
23. Shi, W.; Rajkumar, R. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 13–19 June 2020; pp. 1711–1719.
24. Xiang, C.; Zhang, C.; Bhargava, P.; Jaiswal, A.; Metaxas, D.N. Generating Adversarial Point Clouds with Point-Voxel Manifold. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea, 27 October–2 November 2019; pp. 640–648.
25. Weiss, M.; Tonella, P. Simple Techniques Work Surprisingly Well for Neural Network Test Prioritization and Active Learning (Replicability Study). In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, Virtual Event, South Korea, 18–22 July 2022; pp. 139–150.
26. Wang, D.; Shang, Y. A New Active Labeling Method for Deep Learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Beijing, China, 6–11 July 2014; pp. 112–119.
27. Yoo, S.; Harman, M. Regression Testing Minimization, Selection and Prioritization: A Survey. *Software Testing, Verification and Reliability*, 2012, 22(2), 67–120.
28. Tian, C.; Song, M.; Fan, X.; et al. A Tree-guided CNN for image super-resolution. *IEEE Transactions on Consumer Electronics*, 2025.
29. Tian, C.; Zheng, M.; Lin, C.W.; et al. Heterogeneous window transformer for image denoising. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.
30. Tsai, Y.Y.; Can, E.; Huang, Y.C.; Lee, K.; Oktay, O.; Schwing, A.G. Generating 3D Adversarial Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 13–19 June 2020; pp. 14596–14605.
31. Ying, R.; Bourgraph, D.; Gan, J.; Zitnik, M.; Leskovec, J. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, Vancouver, BC, Canada, 8–14 December 2019; pp. 9240–9251.
32. Botsch, M.; Kobbelt, L.; Pauly, M.; Alliez, P.; Levy, B. *Polygon Mesh Processing*. A K Peters/CRC Press: Natick, MA, USA, 2010.

33. Rusu, R.B.; Blodow, N.; Beetz, M. Fast Point Feature Histograms (FPFH) for 3D Registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 12–17 May 2009; pp. 3212–3217.
34. Pei, K.; Cao, Y.; Yang, J.; Jana, S. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, Shanghai, China, 28–31 October 2017; pp. 1–18.
35. Ma, L.; Juefei-Xu, F.; Zhang, F.; Sun, J.; Xue, M.; Li, B.; Chen, C.; Su, T.; Li, L.; Liu, Y.; et al. DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE)*, Montpellier, France, 3–7 September 2018; pp. 120–131.
36. Mo, K.; Zhu, S.; Chang, A.X.; Yi, L.; Tripathi, S.; Guibas, L.J.; Su, H. PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 15–20 June 2019; pp. 909–918.
37. Rothermel, G.; Untch, R.H.; Chu, C.; Harrold, M.J. Prioritizing Test Cases for Regression Testing. *IEEE Transactions on Software Engineering*, 2001, 27(10), 929–948.
38. Aichernig, B.K.; Brandl, H.; Jöbstl, E.; Krenn, W.; Schlick, R.; Tiran, S. Killing Strategies for Model-based Mutation Testing. *Software Testing, Verification and Reliability*, 2015, 25(8), 716–748.
39. DeMillo, R.A.; Lipton, R.J.; Sayward, F.G. Hints on Test Data Selection: Help for the Practicing Programmer. *IEEE Computer*, 1978, 11(4), 34–41.
40. Papadakis, M.; Kintis, M.; Zhang, J.; Jia, Y.; Le Traon, Y.; Harman, M. Mutation Testing Advances: An Analysis and Survey. *Advances in Computers*, 2019, 112, 275–378.