



# Towards Exploring the Limitations of Test Selection Techniques on Graph Neural Networks: An Empirical Study

Xueqi Dang<sup>1</sup> · Yinghua Li<sup>1</sup> · Wei Ma<sup>3</sup> · Yuejun Guo<sup>2</sup> · Qiang Hu<sup>4</sup> · Mike Papadakis<sup>1</sup> · Maxime Cordy<sup>1</sup> · Yves Le Traon<sup>1</sup>

Accepted: 17 June 2024 / Published online: 22 July 2024  
© The Author(s) 2024

## Abstract

Graph Neural Networks (GNNs) have gained prominence in various domains, such as social network analysis, recommendation systems, and drug discovery, due to their ability to model complex relationships in graph-structured data. GNNs can exhibit incorrect behavior, resulting in severe consequences. Therefore, testing is necessary and pivotal. However, labeling all test inputs for GNNs can be prohibitively costly and time-consuming, especially when dealing with large and complex graphs. In response to these challenges, test selection has emerged as a strategic approach to alleviate labeling expenses. The objective of test selection is to select a subset of tests from the complete test set. While various test selection techniques have been proposed for traditional deep neural networks (DNNs), their adaptation to GNNs presents unique challenges due to the distinctions between DNN and GNN test data. Specifically, DNN test inputs are independent of each other, whereas GNN test inputs (nodes) exhibit intricate interdependencies. Therefore, it remains unclear whether DNN test selection approaches can perform effectively on GNNs. To fill the gap, we conduct an empirical study that systematically evaluates the effectiveness of various test selection methods in the context of GNNs, focusing on three critical aspects: **1) Misclassification detection:** selecting test inputs that are more likely to be misclassified; **2) Accuracy estimation:** selecting a small set of tests to precisely estimate the accuracy of the whole testing set; **3) Performance enhancement:** selecting retraining inputs to improve the GNN accuracy. Our empirical study encompasses 7 graph datasets and 8 GNN models, evaluating 22 test selection approaches. Our study includes not only node classification datasets but also graph classification datasets. Our findings reveal that: 1) In GNN misclassification detection, confidence-based test selection methods, which perform well in DNNs, do not demonstrate the same level of effectiveness; 2) In terms of GNN accuracy estimation, clustering-based methods, while consistently performing better than random selection, provide only slight improvements; 3) Regarding selecting inputs for GNN performance improvement, test selection methods, such as confidence-based and clustering-based test selection methods, demonstrate only slight effectiveness; 4) Concerning performance enhancement, node importance-based test selection methods are not suitable, and in many cases, they even perform worse than random selection.

**Keywords** Graph neural networks · Deep learning testing · Test input selection · Labeling

---

Communicated by: Hadi Hemmati

---

Extended author information available on the last page of the article

## 1 Introduction

Graph Neural Networks (GNNs) have emerged as powerful tools across a wide range of domains, such as social network analysis (Li et al. 2017; Wu et al. 2018; Yu et al. 2020), recommendation systems (Ying et al. 2018; Wu et al. 2022; Fan et al. 2019), and drug discovery (Shi et al. 2020; Bongini et al. 2021). Their ability to capture intricate relationships within graph-structured data has driven significant advancements in the fields of machine learning and artificial intelligence (Li et al. 2022). As the application of GNNs continues to expand, the need for effective testing and evaluation methods becomes increasingly critical.

Similar to traditional deep neural networks (DNNs), testing GNNs faces challenges due to the lack of automated testing oracles (Dang et al. 2023). As a result, labeling GNN test inputs heavily relies on manual annotation, which can be expensive and time-consuming, especially when dealing with large and intricate graphs. Furthermore, in specific specialized domains such as molecular property prediction (Duvenaud et al. 2015), where nodes represent atoms and edges represent covalent bonds, the labeling process can heavily rely on domain-specific knowledge, further increasing the expenses.

In the literature (Ma et al. 2021; Chen et al. 2020; Wang et al. 2021), a promising approach for mitigating labeling costs is *test selection*. It focuses on the selection and labeling of a subset of data from the entire test set. Within the field of DNN testing, various test selection techniques have emerged. These techniques can be broadly classified into two categories: 1) test selection for rapid detection of potentially misclassified tests (Ma et al. 2021; Feng et al. 2020) and 2) test selection for precise accuracy estimation (Chen et al. 2020; Li et al. 2019). For simplicity, we refer to these approaches as misclassification detection approaches and accuracy estimation approaches, respectively.

**Misclassification detection approaches** are designed to identify test inputs that are most likely to be misclassified by the DNN model. These selected inputs serve two primary purposes: facilitating the debugging of DNN-based software and retraining the original DNN model to enhance its accuracy (Feng et al. 2020). In the literature, there are three main methods for misclassification detection: 1) Coverage-Based Methods (Ma et al. 2018; Pei et al. 2017): These methods assess the coverage of DNN neurons to identify potentially misclassified test inputs; 2) Surprise Adequacy-Based Methods (Kim et al. 2019): These techniques select test inputs using metrics related to surprise adequacy and activation traces within DNNs; 3) Confidence-Based Approaches (Feng et al. 2020; Ma et al. 2021; Weiss and Tonella 2022): These methods select tests based on the model's prediction confidence. Test inputs that the DNN model is more uncertain are selected. Notably, confidence-based metrics have proven to be more effective and efficient than both surprise adequacy and coverage-based approaches, with runtime typically taking less than 1 second in most cases (Feng et al. 2020).

**Accuracy estimation approaches** aim to select a small set of test inputs to precisely estimate the accuracy of the whole testing set. By only labeling the selected representative tests, it becomes feasible to reduce the labeling expenses. However, existing approaches designed for DNNs, like CES (Li et al. 2019) and PACE (Chen et al. 2020), are not suitable for GNNs due to their design not aligning with graph datasets.

GNNs fundamentally belong to the family of DNN algorithms. They inherit several core concepts from DNNs, such as deep architectures, nonlinear activation functions, and back-propagation algorithms. Therefore, several existing DNN test selection approaches can be applied to GNNs. However, there is a significant gap in adapting DNN test selection methods for GNNs. This challenge arises because, unlike DNNs, where each sample in the test set is treated independently, GNNs exhibit interdependencies among their test inputs (nodes) (Wu

et al. 2020). Consequently, it remains unclear whether test selection approaches designed for DNNs can be effectively utilized for GNNs. Therefore, it is crucial to investigate the effectiveness of DNN test selection methods in the context of GNNs. To fill the gap, we conduct an empirical study to evaluate the effectiveness of test selection methods when applied to GNNs. Our research focuses on four key aspects:

- **Test Selection for Misclassification Detection** As previously mentioned, confidence-based metrics have demonstrated higher effectiveness and efficiency compared to other existing test selection approaches (Feng et al. 2020). Therefore, we specifically evaluate the effectiveness of various confidence-based test selection approaches for selecting potentially misclassified GNN test inputs.
- **Test Selection for Accuracy Estimation** We investigate the effectiveness of various clustering methods for GNN test selection. We extend the concept of model confidence to accuracy estimation, making clustering approaches utilize the model's prediction probability vector for tests (which reflects model confidence) to conduct clustering.
- **Test Selection for Performance Enhancement (using confidence-based approaches)** We investigate the effectiveness of various confidence-based test selection methods, encompassing both approaches for misclassification detection and accuracy estimation, in selecting retraining inputs to enhance the accuracy of GNNs.
- **Test Selection for Performance Enhancement (using approaches based on node importance)** We investigate the effectiveness of node importance-based test selection methods in selecting retraining inputs to improve GNN accuracy. This exploration is motivated by three factors: 1) Nodes with high importance typically encapsulate critical information and exert a more pronounced influence over the entire graph. Therefore, these nodes are more likely to capture essential information crucial for enhancing model performance (Park et al. 2019); 2) Unimportant nodes can contain noise or irrelevant data that can introduce interference during retraining, thereby diminishing model performance; 3) Node importance is a unique data feature in GNNs that can be leveraged for selecting crucial tests. To the best of our knowledge, there has been limited or no study investigating whether node importance can be effectively used for selecting retraining inputs, highlighting the necessity of conducting relevant research.

Building upon these four critical aspects, we perform an empirical study that encompasses 7 graph datasets and 8 GNN models, systematically evaluating the performance of 22 test selection approaches. To offer a more comprehensive evaluation, we incorporate not only node classification datasets (Yang et al. 2016) but also graph classification datasets (Riesen and Bunke 2008; Bianchi et al. 2021; Neumann et al. 2016) in our analysis. Our empirical findings reveal that while certain test selection methods demonstrate efficacy in the context of DNNs (Ma et al. 2021), they do not translate to the same level of effectiveness when applied to GNNs. We delve into the underlying reasons for this disparity in the experimental section. To provide a concise summary, we present the following key conclusions.

- **Test Selection for Misclassification Detection** In the context of GNNs, confidence-based test selection methods do not exhibit the same level of effectiveness as observed in DNNs.
- **Test Selection for Accuracy Estimation** In most cases, clustering-based test selection methods that utilize the model's confidence vector perform better than random selection. However, their improvements compared to random selection are slight.
- **Test Selection for Performance Enhancement (using confidence-based approaches)** The effectiveness of both confidence-based and clustering-based test selection methods

shows only slight enhancements when compared to random selection in selecting retraining inputs to improve GNN accuracy, despite some methods having been demonstrated as performing well in DNNs (Hu et al. 2021).

- **Test Selection for Performance Enhancement (using node importance-based approaches)** Node importance-based test selection methods are not suitable for selecting retraining data to improve GNN accuracy, and in many cases, they even perform worse than random selection.

Our empirical study provides valuable insights for engineers seeking to apply test selection metrics in GNN contexts. We emphasize the constraints of current test selection approaches for GNNs, thus providing guidance for future research to develop new approaches tailored for GNNs. Our datasets, results, and tools are accessible to the community on [GitHub](#)<sup>1</sup>.

In summary, we make the following contributions in this paper:

- We conduct an empirical study to assess the effectiveness of confidence-based test selection methods in identifying potentially misclassified test inputs for GNNs. Our study reveals that confidence-based test selection methods, which perform well in DNNs, do not demonstrate the same level of effectiveness.
- We empirically investigate the effectiveness of clustering approaches that utilize model confidence vectors in estimating GNN accuracy. We demonstrate that clustering-based methods, while consistently performing better than random selection, provide only slight improvements.
- We investigate the effectiveness of misclassification detection approaches and accuracy estimation approaches in selecting retraining inputs to improve GNN accuracy. We find that test selection methods, such as confidence-based and clustering-based test selection methods, demonstrate only slight effectiveness.
- We investigate the effectiveness of test selection methods based on node importance in selecting retraining inputs to improve the GNN accuracy. The results show that node importance-based test selection methods are not suitable, and in many cases, they even perform worse than random selection.

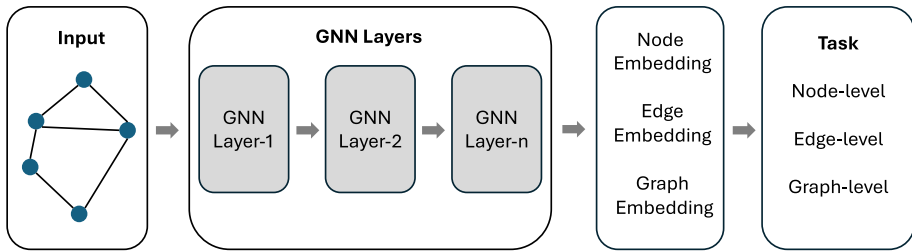
## 2 Background

In this section, we present the fundamental domain concepts central to our research. These encompass Graph Neural Networks, Test Selection in DNN Testing, and Active Learning.

### 2.1 Graph Neural Networks

Graph Neural Networks (GNNs) have demonstrated remarkable effectiveness in addressing machine learning challenges associated with graph-structured data (Zhou et al. 2020; Fan et al. 2019; Sun et al. 2019). These challenges span a variety of domains, including social networks (Li et al. 2017; Wu et al. 2018; Yu et al. 2020), recommendation systems (Ying et al. 2018; Wu et al. 2022; Fan et al. 2019) and bioinformatics (Zhang et al. 2021; Long et al. 2022; Réau et al. 2023). In Fig. 1, we present a general pipeline for GNN models, which includes four main parts: 1) The GNN model receives graph-structured inputs, which can contain nodes and edges (representing the connections between nodes). 2) GNN layers then process this graph-structured data. 3) After multiple layers of processing, the GNN model can generate

<sup>1</sup> [https://github.com/BlueBerry-xueqi/graph\\_testing](https://github.com/BlueBerry-xueqi/graph_testing)



**Fig. 1** The general pipeline for GNN models

node/edge/graph embedding vectors. These are low-dimensional vector representations of node/edge/graph, facilitating efficient processing and analysis by GNN models. 4) Utilizing these embedding vectors, the GNN model can address tasks at the node-level, edge-level, or graph-level correspondingly.

In the following, we introduce some fundamental concepts related to GNNs and graph datasets.

**Graphs** A graph can be formally represented as  $G = (V, E)$ , with  $V$  representing the set of nodes and  $E$  denoting the set of edges that establish connections between these nodes (Dwivedi et al. 2020). Graphs are widely used in various domains (Liu et al. 2020; Jin et al. 2020; Zhou et al. 2020). For instance, in citation networks (Veličković et al. 2017), papers can be represented as nodes linked by citations (edges) and grouped into different categories. In chemistry (Wieder et al. 2020), molecules can be viewed as graphs with atoms as nodes and covalent bonds as edges, simplifying the representation of their 3D structures.

**Graph Analytics Tasks** GNNs can leverage graph structure and node features to perform various analytics tasks. **1) Node-level classification** (Xiao et al. 2022; Zhao et al. 2021), such as categorizing nodes into distinct classes, utilizes individual node predictions. Prevalent datasets for such tasks include Cora (Sen et al. 2008), CiteSeer (Sen et al. 2008), and PubMed (Sen et al. 2008). **2) Graph-level classification** aims to determine entire graph attributes, like predicting molecular properties in a chemical graph. Datasets for these tasks include Mutagenicity (Riesen and Bunke 2008), NCI1 (Shervashidze et al. 2011), and MSRC21 (Neumann et al. 2016). **3) Edge-level classification** focuses on classifying edge types between two given nodes. For example, in biological networks, GNNs can utilize the information of a protein and a small molecule to predict their binding affinity, which is considered as edges within a graph. Datasets for edge classification include: DrugBank (Wishart et al. 2018) and BindingDB (Liu et al. 2007).

**Graph Embeddings** (Cai et al. 2018) offer an approach to diminish the dimensions of nodes, edges, and their related attributes while preserving vital structural information and graph characteristics (Fu et al. 2020). In graph embedding, each node or edge is mapped to a vector, typically in low dimensions. This low-dimensional representation effectively captures the relationships and similarities between nodes or edges, enabling more efficient computation and analysis within the vector space.

**Message Passing** The fundamental concept behind Message Passing in GNNs is to enhance the representation of individual nodes by propagating and aggregating information among neighboring nodes, as described in Wu et al. (2020). For example, when calculating the representation of a node  $N$  at time step  $k$ , the process involves: 1) Gather information from neighbors: Compute the sum of messages from all neighboring nodes of node  $N$  to gather information. 2) Utilize the obtained information: Combine the received messages with the

representation of node  $N$  at time step  $(k - 1)$  to compute the representation of node  $N$  at time step  $k$ .

**Applying GNNs in Software Engineering** GNNs can be applied to various aspects of the field of software engineering. One prevalent application lies in software vulnerability detection (Cheng et al. 2021, 2022). Cheng et al. (2021) proposed DeepWukong, a novel approach for software vulnerability detection, which utilizes GNNs to encode code fragments into a concise low-dimensional representation. Initially, DeepWukong extracts program slices from code fragments, labeling a slice (or an XFG) as vulnerable if it contains a vulnerable statement. Subsequently, a neural network model is trained using both safe and vulnerable program slices. Both the unstructured and structured code information of a program are incorporated when constructing the neural networks, with both types of information fed into the GNNs to generate a compact code representation in the latent feature space. By leveraging recent advancements in GNNs to learn from vulnerable and safe program slices, DeepWukong enables more precise bug prediction. Cheng et al. (2022) proposed ContraFlow, which overcomes limitations of previous GNN-based software vulnerability detection methods by focusing on preserving value flow paths rather than the entire graph. By employing contrastive learning, ContraFlow efficiently selects feasible value-flow paths in the embedding space to represent a code fragment accurately. ContraFlow can identify potential error paths based on path-sensitive representations and interpret crucial value flow paths causing vulnerabilities.

## 2.2 Test Selection in DNN Testing

In the context of DNN testing (Haq et al. 2021; Panichella et al. 2017; Dang et al. 2024; Li et al. 2023), test selection (Ma et al. 2021; Hu et al. 2021) focuses on addressing a practical concern: while collecting unlabelled data is easy and cost-effective, labeling all of it demands substantial effort and specialized domain knowledge. This challenge is typically exacerbated by three key factors: 1) Large-Scale Test Sets: Test sets can be extensive, increasing the workload associated with labeling. 2) Manual Analysis as the Primary Labeling Method: The primary method of labeling involves manual analysis, typically requiring the involvement of multiple individuals to ensure accurate labeling. 3) Dependency on Domain-Specific Knowledge: Labeling frequently necessitates domain-specific expertise, resulting in higher costs associated with employing professionals for the task.

Test selection has emerged as a practical solution for dealing with the labelling cost issue. It involves carefully selecting a subset of unlabeled test data to serve two main objectives: testing DNNs and improving the performance of pre-trained DNNs through retraining. Test selection can be broadly categorized into two main aspects:

- **Misclassification Detection** (Ma et al. 2021; Feng et al. 2020) This aspect focuses on selecting test inputs that are more likely to be misclassified by the DNN model. These tests are more likely to reveal errors in the DNN model and are therefore referred to as “bug-revealing test inputs”. Labeling only these test data can lead to reduced overall labeling costs. Furthermore, in active learning contexts, this test data can then be utilized to enhance the model through retraining (Hu et al. 2021).
- **Accuracy Estimation** (Chen et al. 2020; Li et al. 2019) This aspect involves selecting a small set of representative test inputs capable of precisely estimating the accuracy of the entire testing dataset. By labeling only these representative tests, it becomes possible to estimate the accuracy of the entire test set, thus reducing labeling costs.

## 2.3 Active Learning

Active learning is a well-established concept within both the software engineering (SE) and machine learning (ML) communities (Hu et al. 2021). The fundamental idea behind active learning is to employ machine learning techniques to identify data samples that are relatively challenging to classify (Ren et al. 2021). These samples are then presented for human annotation. The annotated data is subsequently used to further train the target ML models to improve the model's performance. The primary objective of active learning is to determine which samples should be prioritized for manual labelling, enabling the model to actively select the informative data to train the model (Ranganathan et al. 2017). Existing work (Weiss and Tonella 2022) has demonstrated that test selection methods can be employed for active learning. Weiss and Tonella (2022) empirically investigated the effectiveness of various DNN test selection techniques (e.g., DeepGini and Entropy) in identifying inputs potentially useful for active learning. Their study shows that DeepGini, along with several uncertainty-based methods, can effectively select informative inputs in the context of active learning.

## 3 Approach

In our study, we assessed a total of 22 approaches, comprising 7 test selection methods for misclassification detection, 5 test selection approaches for accuracy estimation, 7 node importance metrics, and one baseline method (i.e., random selection). We selected these approaches for the following reasons: 1) These approaches are adaptable for the corresponding GNN test selection task. For example, DeepGini, as highlighted in its original paper (Feng et al. 2020), can be used to identify potentially misclassified test inputs; 2) The selected approaches have demonstrated their effectiveness in the context of DNNs (Feng et al. 2020; Ma et al. 2021); 3) The authors of these approaches have made their implementations publicly available. Below, we will provide a detailed explanation of the basic logic behind each test selection method.

### 3.1 Misclassification Detection Approaches

We employed a total of 10 test selection methods that can be used to detect potentially misclassified GNN tests. One of the classic methods is DeepGini (Feng et al. 2020). Moreover, our empirical study also evaluated several active learning-based test selection strategies (Wang and Shang 2014), including Margin Sampling, Least Confidence, and Entropy. Active learning (Hu et al. 2021) focuses on maximizing model performance gains with minimal sample labeling. Specifically, it aims to select the most valuable samples within an unlabeled dataset and hand them over to the oracle (e.g., human annotator) for labeling, thereby reducing labeling costs while maintaining the model performance. Below, we provide a detailed introduction to the test selection approaches we evaluated.

- **DeepGini** (Feng et al. 2020) DeepGini quantifies the uncertainty in a model's prediction for a given test by calculating the Gini score of this test. This score is derived from the model's prediction probability vector for the test. A higher Gini value indicates that the model is more uncertain on the specific test. Therefore, the test is considered more likely to be misclassified. The computation of the Gini score is illustrated in Formula (1).

$$G(t) = 1 - \sum_{i=1}^N p_{t,i}^2 \quad (1)$$

where  $N$  represents the number of prediction classes, and  $p_{t,i}$  represents the probability that the model will classify the test  $t$  into class  $i$ .

- **Margin Sampling** (Wang and Shang 2014) Margin sampling is an uncertainty-based active learning strategy. Its core idea is to select samples that the model finds most challenging to classify for labeling. Margin Sampling focuses on the difference in the model’s predicted probabilities for the two most confident classes. The smaller this probability gap, the more uncertain the model is about the classification of that sample. The uncertainty score of Margin Sampling is calculated by Formula (2).

$$Margin(t) = p_k(t) - p_j(t) \tag{2}$$

where  $p_k(t)$  refers to the model’s predicted probability for the most confident classification.  $p_j(t)$  refers to model’s predicted probability for the second most confident classification

- **Least Confidence (LC)** (Wang and Shang 2014) Least Confidence is an active learning strategy based on model uncertainty. Specifically, it selects samples for which the model’s prediction is the least confident for labeling. In a classification task, if a model has a low maximum predicted probability value for a specific unlabeled sample, it indicates that the model is highly uncertain about the classification of that sample. The Least Confidence strategy selects such samples for labelling. The score of Least Confidence is computed using Formula (3).

$$L(t) = 1 - \max_{i=1:n} p_i(t) \tag{3}$$

where  $p_i(t)$  represents the probability of test input  $t$  being classified into category  $i$ . Hence,  $\max_{i=1:n} p_i(t)$  represents the model’s predicted probability for the most confident classification.

- **Least Confidence-variant (LC-variant)** (Wang and Shang 2014) In contrast to the Least Confidence metric, which ranks classifications based on the most confident predictions, the Least Confidence-variant model assesses uncertainty by focusing on the model’s least confident prediction category. This variant considers that when the difference between the model’s prediction probability for the least confident classification and 0 is large, it signifies that the model is more uncertain about this test, and this test is more likely to be misclassified. The formula for this variant is provided in Formula (4).

The rationale behind this variant is rooted in the concept of uncertainty, as discussed in previous studies Feng et al. (2020). Specifically, considering a classifier  $M$  capable of classifying test inputs into  $N$  categories, when the prediction probability vector of  $M$  for a test  $t$  is  $(\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N})$ , it signifies that classifier  $M$  is the most certain about this test  $t$ . Since the highest value that the model’s prediction probability can reach for its least confident classification is  $\frac{1}{N}$ , when the model’s prediction probability for the least confident category is higher, it suggests that the model’s confidence for the least confident category approaches  $\frac{1}{N}$ . This suggests that the model exhibits greater uncertainty when predicting this test case. This test is considered more likely to be misclassified.

$$L(t) = \min_{i=1:n} p_i(t) - 0 \tag{4}$$

where  $p_i(t)$  represents the probability of test input  $t$  being classified into category  $i$ . Hence,  $\min_{i=1:n} p_i(t)$  represents the model’s predicted probability for the least confident classification.

- **Entropy** (Weiss and Tonella 2022) Entropy is a commonly used method in active learning. It can measure the uncertainty of a model’s predictions for a given sample. The entropy



method selects samples by calculating the entropy value of the model's predictions for each unlabeled sample. For a given sample, a high entropy value indicates that the model is highly uncertain about the classification of that sample. Therefore, this strategy tends to select samples with high entropy values for labeling, with the aim of improving the model's performance by adding information from these highly uncertain samples.

- **Multiple-Boundary Clustering and Prioritization (MCP)** (Shen et al. 2020) MCP is an extension of the Margin Sampling. It begins by dividing the data into distinct "boundary areas" based on the top-2 predicted classes. Then, MCP selects data points from each area based on the Margin. The selected data points are considered to be tests for which the model exhibits a higher degree of uncertainty. These tests are considered more likely to be misclassified.
- **Variance** (Ma et al. 2021) For a given test case, Variance quantifies the uncertainty in the model's predictions by computing the variance of the model's prediction probabilities for that specific test. A smaller variance suggests that the model exhibits greater uncertainty regarding this test, and this test is considered more likely to be misclassified. The formula for Variance is provided in Formula (5).

$$\text{Var}(x) = \frac{1}{N} \sum_{i=1}^N \text{var}(p_i(t)) \quad (5)$$

where  $N$  represents the number of test inputs in the test set. where  $p_i(t)$  represents the probability of test  $t$  being classified into category  $i$ .

- **ATS** (Gao et al. 2022) ATS is the first adaptive test selection method designed for DNNs, which utilizes differences in model outputs to measure the diversity of behaviors of DNN test inputs. The objective of ATS is to select more diverse tests from the candidate set, as these tests can reveal more different faults in the DNN-driven software.
- **GraphPrior** (Dang et al. 2023) GraphPrior is a test prioritization method specifically designed for GNNs. It utilizes mutation testing to prioritize potentially misclassified test inputs. Specifically, given a test set and a GNN model under testing, GraphPrior generates mutated models based on the original GNN model. GraphPrior assumes that a test input is more likely to be misclassified if it can "kill" many mutated models. Based on this assumption, it identifies and prioritizes possibly misclassified tests.
- **Random selection** (Elbaum et al. 2002) Through the baseline random selection, tests are selected randomly from the test set.

### 3.2 Accuracy Estimation Approaches

The aforementioned confidence-based methods rely on the model's prediction probability vector to assess whether a test is prone to being incorrectly predicted. These methods are efficient and consume minimal time since they only use the model's final prediction probability vector and mathematical approaches for estimating uncertainty. Based on existing research (Chen et al. 2020), clustering is a practical approach for test selection to estimate the accuracy of a test set. Clustering groups similar data points together, allowing for the extraction of representative points from each cluster, which can effectively represent the entire test set. Therefore, we empirically explore the combination of clustering methods with prediction probability vectors for test selection in the context of graph networks to estimate the accuracy of the test set. Below, we introduce all the clustering methods used in our study.

- **K-Means** (Ahmed et al. 2020) K-Means is an unsupervised clustering algorithm. The algorithm initially divides the data into K groups and randomly selects K objects as the initial cluster centers. It then computes distances between each point and all the cluster centers, assigning each point to the closest center. Subsequently, the algorithm recalculates the centroid of each cluster. This process continues to iterate until a specific termination condition is met.
- **K-Means Plus** (Arthur and Vassilvitskii 2007) K-Means Plus is an extension of the K-Means algorithm, primarily enhancing the way initial cluster centers are chosen. In the traditional K-Means algorithm, initial cluster centers are randomly chosen, which can lead to different results in different runs and can affect the algorithm's convergence speed and clustering quality. K-Means++ addresses this issue by intelligently selecting the initial cluster centers, aiming to enhance the algorithm's performance.
- **MiniBatch K-means** (Sculley 2010) MiniBatch K-means is an optimized variant of the K-Means algorithm designed for efficiently handling large-scale data, reducing computational time. It utilizes mini-batches, which are small, random, fixed-size data subsets, to manage data in memory. During each iteration, the algorithm gathers a random sample of the data and employs it to update the clusters.
- **Gaussian Mixture Model (GMM)** (Patel and Kushwaha 2020) The Gaussian Mixture Model is a probabilistic model that posits that all data points are generated by a mixture of finite Gaussian distributions with unknown parameters. It can be thought of as an extension of K-means clustering that incorporates information about the data's covariance structure and potential Gaussian distribution centers.
- **Hierarchical Clustering** (Kaushik and Mathur 2014) Hierarchical Clustering is a versatile clustering algorithm that iteratively combines or divides clusters to create nested structures. The hierarchical organization in Hierarchical Clustering is visualized as a tree, with the root representing the cluster containing all samples and the leaves representing clusters with only one sample each.

### 3.3 Node Importance Metrics

In RQ4, we employed seven approaches to measure node importance in order to perform test selection. These methods were extracted from existing studies (Hu et al. 2015; Qiong and Dongxia 2016; Yang et al. 2019; Ando et al. 2021).

- **Degree** Degree measures the importance of a node based on the number of edges surrounding the node. Nodes with a higher number of edges are considered more important.
- **Eccentricity** Eccentricity quantifies a node's importance by assessing the longest distance from that node to all other nodes. Nodes with small eccentricity values are deemed more crucial, as they play a pivotal role in connecting various components and influencing information dissemination.
- **Center** The Center approach assesses the importance of a node by calculating its distance from the network center. Nodes closer to the center are considered more important. Center posits that nodes closer to the center have a greater influence and significance in terms of network connectivity and information propagation.
- **Betweenness Centrality (BC)** BC assesses the importance of a node by evaluating its role as an intermediary within the network. The node's betweenness centrality depends on the number of times it acts as a transit point along the shortest paths in the network. A higher betweenness centrality indicates that the node plays a more crucial role in

connecting paths between different nodes in the network, and therefore, it is considered more important.

- **Eigenvector Centrality (EC)** Eigenvector Centrality associates a node's importance with the degree to which it is connected to other important nodes. The centrality of a node is determined by the importance of the nodes it is linked to; if a node is connected to others with high Eigenvector Centrality, it will also be considered more important.
- **PageRank** PageRank evaluates the relative significance of nodes in a graph by considering their connectivity and the influence of nodes linked to them. A node's PageRank value depends on both its number of connections and the importance of the nodes that are connected to it. Nodes connected to nodes with higher PageRank values are regarded as more important in this ranking method.
- **Hyperlink-Induced Topic Search (Hits)** Hits determine the importance of nodes through two metrics: Authorities and Hubs. Authorities are assessed based on the quantity and quality of inbound links a node receives, measuring its role as a source of information. Hubs, on the other hand, are evaluated based on the quantity and quality of outbound links, gauging their role as intermediaries in information dissemination. These two metrics interact and are jointly used to assess the relative importance of nodes in the graph network.

## 4 Study Design

### 4.1 Overview

Similar to traditional deep neural networks (DNNs), testing Graph Neural Networks (GNNs) also faces challenges due to the absence of automated testing oracles. This leads to the need for manual labeling of test inputs, a process that can be labor-intensive, especially for large and intricate graphs. Furthermore, in specialized domains like drug discovery, as exemplified by protein interface prediction (Jha et al. 2022), labeling heavily relies on domain-specific knowledge, further escalating costs. In response to the labeling cost issue, existing studies mainly focus on two motivations in the field of DNN testing selection: misclassification detection and accuracy estimation.

- **Misclassification Detection** Misclassification detection aims to select test inputs that are more likely to be misclassified by the DNN model. These selected tests serve two primary purposes: 1) Testers can use them for debugging DNN-based software to enhance the quality of DNNs, and 2) Testers can employ them for DNN model retraining, effectively reducing the cost associated with retraining.
- **Accuracy Estimation** Accuracy Estimation aims to select a small set of representative test inputs capable of providing an accurate estimate of the entire test set's accuracy.

However, a notable gap exists in adapting DNN test selection methods for GNNs. This challenge emerges due to the distinct nature of GNN test data, where test inputs (nodes) are interconnected, unlike DNNs, where each test sample is treated independently. Consequently, it remains uncertain whether test selection approaches originally tailored for DNNs can be suitably applied to GNNs. To fill the gap, we conduct an empirical study to assess the effectiveness of test selection methods when employed within the context of GNNs, including confidence-based approaches, clustering-based approaches, and node-importance-based approaches.

Figure 2 presents an overview of our empirical study. Our study initially focused on three crucial aspects of GNN test selection: GNN accuracy estimation, GNN misclassification detection, and GNN performance enhancement. Specifically, RQ1 focuses on misclassification detection. RQ2 corresponds to accuracy estimation. RQ3 and RQ4 target GNN performance enhancement. In the following, we provide a detailed description of each research question.

- **RQ1: Misclassification Detection.** We evaluate the effectiveness of confidence-based test selection methods for identifying potentially misclassified GNN test inputs, building on their demonstrated efficiency in previous work (Feng et al. 2020).
- **RQ2: Accuracy Estimation.** We extend the concept of model confidence for accuracy estimation, evaluating the effectiveness of various clustering methods that utilize the model’s confidence vector in estimating the accuracy of the GNN test set.
- **RQ3: Performance Enhancement (using confidence-based methods).** We assess the effectiveness of various test selection approaches, encompassing both misclassification detection and accuracy estimation approaches, in selecting retraining inputs for enhancing GNN accuracy.
- **RQ4: Performance Enhancement (using node importance-based methods).** We investigate the effectiveness of node importance-based test selection methods in selecting retraining inputs for improving GNN accuracy. This is motivated by the fact that nodes with high importance tend to capture critical information, while low-importance nodes can introduce noise during retraining. Leveraging node importance in GNNs for test selection is a novel and unexplored area of research.

To provide a more comprehensive assessment, we conducted experiments using a diverse set of 7 graph datasets with 8 GNN models to evaluate the performance of 20 test selection approaches. It is important to emphasize that our dataset includes not only widely adopted node-level datasets but also graph-level datasets in order to ensure a robust evaluation of our methodology. By analyzing the performance of current test selection approaches for GNNs, we aim to investigate the limitations of existing test selection methods in the context of GNNs and provide insights for the future development of novel GNN-oriented test selection methods.

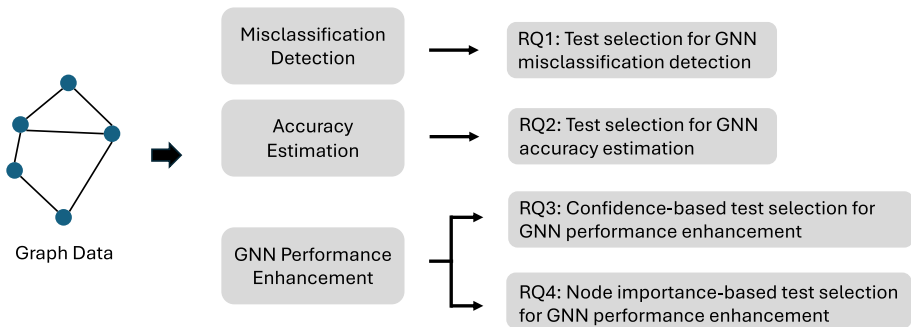


Fig. 2 Overview of our empirical study

## 4.2 Research Questions

Our experimental evaluation answers the research questions below.

- **RQ1: How effective are different test selection metrics in detecting misclassified test inputs for GNNs?** Test selection has emerged as a promising approach for reducing the labelling cost in the testing process. While several test selection techniques have been proposed in the context of DNN testing, their adaptation to GNNs poses distinctive challenges owing to the differences between the test data for DNNs and GNNs. In particular, DNN test inputs are typically independent of one another, whereas GNN test inputs, represented as nodes, exhibit complex interdependencies. Consequently, it remains uncertain whether the DNN test selection methods can perform well on GNNs. In this research question, we assess the effectiveness of multiple test selection approaches in identifying test inputs that are more likely to be misclassified within the context of GNNs.
- **RQ2: How do various accuracy estimation methods perform when applied to GNNs?** Test selection approaches for accuracy estimation are designed to select a subset of test inputs that can effectively estimate the accuracy of the entire testing set. By labeling only the selected tests, the labeling costs can be reduced. In this research question, we empirically assess various test selection approaches in estimating the accuracy of GNNs.
- **RQ3: How do different test selection approaches perform in selecting informative inputs for retraining GNN models?** In this research question, we investigate the effectiveness of diverse confidence-based test selection methods in selecting retraining inputs for GNN accuracy improvement. These methods encompass misclassification detection approaches (RQ1) and accuracy estimation approaches (RQ2).
- **RQ4: To what extent can node importance guide the selection of retraining inputs for GNNs?** In this research question, we explore the effectiveness of node importance-based methods in selecting inputs to enhance GNN accuracy. This exploration is motivated by: 1) Nodes with high importance typically contain crucial information and have a significant impact on the entire graph, making them valuable for improving model performance; 2) Conversely, unimportant nodes can introduce noise or irrelevant data during retraining, potentially degrading model performance; 3) Node importance in GNNs remains unexplored for selecting crucial tests. Our research aims to address this gap.

## 4.3 GNN Models and Datasets

In our experiments, we utilized 7 graph datasets and 8 GNN models to assess the performance of 20 test selection approaches. Detailed information about each dataset and model is elaborated upon in the subsequent sections.

### 4.3.1 Graph Datasets

To provide a more comprehensive evaluation, our dataset encompasses not only widely adopted node-level datasets but also edge-level and graph-level datasets. Node-level tasks are centered on making predictions for individual nodes within a graph. The node-level datasets we utilized consist of Cora (Yang et al. 2016), CiteSeer (Yang et al. 2016), and PubMed (Yang et al. 2016). Edge-level datasets focus on predicting edge types between two given nodes. Our adopted edge-level datasets are DrugBank (Wishart et al. 2018) and BindingDB (Liu et al. 2007). In contrast, graph-level tasks are oriented towards predicting global properties

or characteristics of an entire graph. Our selection of graph-level datasets includes Mutagenicity (Riesen and Bunke 2008), NCI1 (Shervashidze et al. 2011), GraphMNIST (Bianchi et al. 2021), and MSRC21 (Neumann et al. 2016).

### 1) Node Classification Datasets

- **Cora** (Yang et al. 2016) The Cora dataset comprises 2,708 scientific publications (nodes) and 5,429 links (edges) representing citations between them. Nodes represent machine learning papers, and edges indicate citations between pairs of papers. Each paper is categorized into one of seven classes, including topics like reinforcement learning and neural networks.
- **CiteSeer** (Yang et al. 2016) The CiteSeer dataset comprises 3,327 scientific publications (nodes) and 4,732 links (edges). Each paper belongs to one of six categories (e.g., artificial intelligence and machine learning).
- **PubMed** (Yang et al. 2016) The PubMed dataset contains 19,717 diabetes-related scientific publications (nodes) connected by 44,338 links (edges). Publications are classified into three classes (e.g., Cancer and AIDS).

### 2) Graph Classification Datasets

- **Mutagenicity** (Riesen and Bunke 2008) The Mutagenicity dataset presents a diverse collection of 4,337 small molecule graphs, each belonging to one of two distinct classes. It serves as a valuable resource for exploring the mutagenic properties of these molecules, offering insights into their potential health and environmental implications.
- **NCI1** (Shervashidze et al. 2011) NCI1 encompasses 4,110 small molecule graphs, comprising 407 unique molecules classified into two fundamental categories: toxicity and biological relevance. This dataset plays a crucial role in toxicity prediction and drug discovery efforts.
- **GraphMNIST** (Bianchi et al. 2021) GraphMNIST stands as a significant resource in the field of computer vision, consisting of a vast database of handwritten digits. It comprises 412 instances across ten distinct classes, corresponding to integer values from 0 to 9.
- **MSRC21** (Neumann et al. 2016) The MSRC21 dataset is a comprehensive compilation of 563 real-world network graphs from the field of computer vision.

### 3) Edge Classification Datasets

- **DrugBank** (Wishart et al. 2018) The DrugBank dataset is a multi-class classification dataset primarily focused on drug-drug interactions (DDIs). It involves predicting the interaction type between pairs of drugs given their SMILES strings. Compiled manually from FDA/Health Canada drug labels and original literature, the dataset encompasses 86 distinct interaction types, covering a total of 191,808 DDI pairs involving 1,706 unique drugs.
- **BindingDB** (Liu et al. 2007) BindingDB is a public, web-accessible database dedicated to measuring binding affinities. It primarily focuses on the interactions between proteins considered to be drug targets and drug-like small molecules. In our experiment, we classified edges based on the magnitude of their binding affinities for the edge classification task.

## 4.3.2 GNN Models

- **GCN** (Kipf and Welling 2016) GCN is a specialized type of convolutional neural network designed to operate directly on graph structures. It addresses the task of classifying nodes

within graphs, such as documents in citation networks, where only a limited number of nodes have labels. The fundamental concept behind GCN involves leveraging the relationships between edges in a graph to consolidate node information and produce updated node representations. GCN has found application in various research studies, as evidenced by its inclusion in prior works (He et al., 2020; Hong et al., 2020).

- **GAT** (Veličković et al. 2017) The inception of GAT arose from the necessity to enhance traditional Graph Convolutional Networks (GCN). GCN considers all neighboring nodes as equally important. However, in practical scenarios, different neighboring nodes can hold different degrees of significance. As a result, GAT incorporates a self-attention mechanism that assigns individualized attention scores to each neighbor. Consequently, GAT excels in identifying and prioritizing the most crucial neighbors during the information aggregation process.
- **Graph Isomorphism Network (GIN)** (Xu et al. 2018) GIN is designed for processing graph data and solving the graph isomorphism problem. Its working principle involves learning the structural information and connectivity patterns among nodes in a graph, enabling effective identification and comparison of isomorphism between different graphs. The core idea of GIN is to iteratively aggregate feature information from nodes within the graph, capturing and representing essential features of the entire graph.
- **Higher-order Graph Neural Networks (GraphNN)** (Morris et al. 2019) GraphNN is an advanced class of graph-based machine learning models that extend traditional GNNs to capture intricate higher-order relationships within graph-structured data.
- **Message Passing Neural Networks (MPNNs)** (Gilmer et al. 2017) MPNN is a general framework for supervised learning on graphs structured data. It is based on the commonness between several state-of-the-art graph-based neural models.
- **Attention-based Graph Neural Network (AGNN)** (Thekumparampil et al. 2018) AGNN is a neural network architecture designed for graph data analysis. Its distinctive feature is the complete removal of traditional fully connected intermediate layers, replaced with attention mechanisms to better preserve the information within the graph structure.
- **Graclus GNNs** (Mesquita et al. 2020) Graclus GNNs is an approach that integrates the Graclus graph clustering algorithm with GNNs. Graclus is utilized for partitioning a given graph into clusters or communities based on node similarity or relationships. In this model, the graph data undergoes pre-processing with Graclus.
- **GNNs with convolutional ARMA filters (ARMA)** (Bianchi et al. 2021) ARMA refers to an optimized GNN architecture with a new graph convolutional layer inspired by the auto-regressive moving average (ARMA) filter. ARMA brings significant improvements for node classification, graph classification, etc.
- **GSAGE-E** (Hamilton et al. 2017) Graph Sample and Aggregate (GraphSAGE) generates embeddings for nodes by accumulating and integrating characteristics from their adjacent nodes. GraphSAGE samples a predetermined quantity of neighbors for each node. GSAGE-E is a variant model of GraphSAGE aimed at edge classification tasks. In this model, the fused information of two nodes (i.e., concatenating the vectors of two nodes) is utilized to predict the category of the edge between them.
- **TAGCN-E** (Du et al. 2017) The Topology Adaptive GCN (TAGCN) employs a collection of learnable filters, each of a fixed size, to execute convolutional operations on graph structures. These filters adapt to the unique topology of the graph during the convolution process. TAGCN-E is a variant model of TAGCN that focuses on edge classification. In TAGCN-E, the fused information of two nodes is utilized to predict the category of the edge between them.

## 4.4 Measurements

### 4.4.1 Percentage of Fault Detected (PFD)

Following the prior research (Feng et al. 2020), we employ PFD to assess the effectiveness of various test selection methods in detecting misclassified test inputs. The computation of PFD is represented in Formula (6). From a mathematical standpoint, PFD measures the ratio of correctly detected misclassified test inputs to the total number of misclassified tests within the test set. A higher PFD value indicates that the evaluated test selection approach is more effective at identifying misclassified inputs.

$$PFD = \frac{\#T_{detect}}{\#T_{mis}} \quad (6)$$

where  $\#T_{detect}$  represents the number of detected misclassified test inputs, while  $\#T_{mis}$  denotes the total number of misclassified test inputs in the test set. In our study, we assessed the PFD values of different test selection approaches under varying ratios of prioritized tests

### 4.4.2 Root Mean Square Error

The root mean square error (RMSE) measures the average difference between the estimated accuracy and the actual accuracy of a test set. The calculation formula is shown in Formula (7). A lower RMSE value indicates that the selected test inputs can predict the accuracy of the entire test set more accurately, indicating that the utilized test selection method is more effective.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n |a\hat{c}_i - acc|^2} \quad (7)$$

where  $acc$  refers to the actual accuracy, and  $a\hat{c}$  refers to the estimated accuracy.

## 4.5 Implementation and Configuration

This project is implemented using the PyTorch 1.11.0 and PyTorch Geometric 2.1.0 framework. We integrated the available implementations of the test selection approaches (Feng et al. 2020; Ma et al. 2021; Hu et al. 2021) into our experimental pipeline. To implement the clustering-based test selection methods, we utilized the package scikit-learn 1.0.2. To implement node importance metrics, we employed the package networkx 2.6.3. Our experiments were conducted on a high-performance computer cluster, with each cluster node equipped with a 2.6 GHz Intel Xeon Gold 6132 CPU and an NVIDIA Tesla V100 16G SXM2 GPU. For data processing tasks, we conducted corresponding experiments on a MacBook Pro laptop running Mac OS Big Sur 11.6, equipped with an Intel Core i9 CPU and 64 GB of RAM.

## 5 Results and Analysis

### 5.1 RQ1: Test Selection for GNN Misclassification Detection

**Objectives:** We investigate the effectiveness of 8 confidence-based test selection methods for GNNs in the context of node classification and graph classification tasks, respectively.



**Table 1** Effectiveness of misclassification detection approaches with respect to random selection (baseline) in terms of PFD

Data	Model	Approach DeepGini	ATS	LC	Margin	GraphPrior	Entropy	LC-variant	Variance	MCP
Cora	GCN	4.0588	2.3725	4.1531	<b>4.7450</b>	5.9804	3.9332	3.2034	4.0500	4.3557
	GAT	2.0876	2.8715	2.3436	<b>5.7431</b>	6.5317	1.9738	1.3559	2.0649	4.5054
	AGNN	3.4733	2.4286	3.4006	<b>4.8574</b>	5.6503	3.4509	3.2690	3.4613	3.7715
	ARMA	4.2897	2.7089	4.2859	<b>5.4179</b>	6.6693	4.1769	3.5346	4.2250	4.9692
	GCN	1.9209	1.8116	2.1096	<b>3.6233</b>	4.9170	1.8282	1.3654	1.8942	3.5128
	GAT	1.0895	2.0998	1.2338	<b>4.1997</b>	5.4637	1.0201	0.7090	1.1042	3.5111
CiteSeer	AGNN	2.1020	2.0201	2.0425	<b>4.0402</b>	5.2734	2.0891	1.9020	2.0891	3.9655
	ARMA	2.5656	1.8109	2.6523	<b>3.6219</b>	4.2475	2.4156	2.0682	2.5614	3.3946
	GCN	3.1220	2.2013	3.3303	<b>4.4028</b>	5.4706	2.8711	1.7775	3.1095	4.2942
	GAT	1.8241	1.5241	1.8955	<b>5.2141</b>	5.7141	1.6941	1.2343	1.7783	5.1693
	AGNN	2.5477	2.5891	2.6080	<b>5.1784</b>	5.9597	2.4586	1.8991	2.5610	4.8041
	ARMA	4.2553	3.2553	4.2477	5.0853	5.8553	4.1420	2.9595	4.2304	<b>5.1101</b>
PubMed	GraphNN	5.0694	4.0694	5.0944	6.0750	-	4.9306	3.6417	5.1600	4.9083
	GIN	2.5519	2.2519	2.6519	3.9764	-	2.4736	2.2387	2.5962	1.9009
MSRC21	GSAGE-E	4.0417	1.9535	4.0878	3.9071	-	3.8784	3.0482	4.0321	4.0122
	TAGCN-E	3.7352	1.7373	3.7517	3.4748	-	3.5893	2.8072	3.7024	3.6539
DrugBank	GSAGE-E	3.1728	1.5787	3.1799	3.1574	-	3.1616	2.8503	3.1682	3.1701
	TAGCN-E	2.9532	1.4667	2.9856	2.9335	-	2.9582	2.7912	2.9515	2.9574

**Table 2** Comparative effectiveness of misclassification detection approaches relative to baseline (normalization analysis)

Data	Model	Approach DeepGini	ATS	LC	Margin	GraphPrior	Entropy	LC-variant	Variance	MCP
Cora	GCN	0.6787	0.3967	0.6945	<b>0.7934</b>	1.0000	0.6577	0.5356	0.6772	0.7283
	GAT	0.3196	0.4396	0.3588	<b>0.8793</b>	1.0000	0.3022	0.2076	0.3161	0.6898
	AGNN	0.6147	0.4298	0.6018	<b>0.8597</b>	1.0000	0.6107	0.5786	0.6126	0.6675
	ARMA	0.6432	0.4062	0.6426	<b>0.8124</b>	1.0000	0.6263	0.5300	0.6335	0.7451
	GCN	0.3907	0.3684	0.4290	<b>0.7369</b>	1.0000	0.3718	0.2777	0.3852	0.7144
	GAT	0.1994	0.3843	0.2258	<b>0.7687</b>	1.0000	0.1867	0.1298	0.2021	0.6426
CiteSeer	AGNN	0.3986	0.3831	0.3873	<b>0.7661</b>	1.0000	0.3962	0.3607	0.3962	0.7520
	ARMA	0.6040	0.4263	0.6244	<b>0.8527</b>	1.0000	0.5687	0.4869	0.6030	0.7992
	GCN	0.5707	0.4024	0.6088	<b>0.8048</b>	1.0000	0.5248	0.3249	0.5684	0.7850
	GAT	0.3192	0.2667	0.3317	<b>0.9125</b>	1.0000	0.2965	0.2160	0.3112	0.9047
	AGNN	0.4275	0.4344	0.4376	<b>0.8689</b>	1.0000	0.4125	0.3187	0.4297	0.8061
	ARMA	0.7619	0.5828	0.7605	0.9105	1.0000	0.7416	0.5299	0.7574	<b>0.9149</b>
MSRC21	GraphNN	0.8345	0.6699	0.8386	1.0000	-	0.8116	0.5995	0.8494	0.8080
	GIN	0.6418	0.5663	0.6669	1.0000	-	0.6221	0.5630	0.6529	0.4780
DrugBank	GSAGE-E	0.9887	0.4779	1.0000	0.9558	-	0.9488	0.7457	0.9864	0.9815
	TAGCN-E	0.9956	0.4631	1.0000	0.9262	-	0.9567	0.7482	0.9869	0.9739
	GSAGE-E	0.9978	0.4965	1.0000	0.9929	-	0.9942	0.8963	0.9963	0.9969
BindingDB	TAGCN-E	0.9891	0.4913	1.0000	0.9825	-	0.9908	0.9349	0.9886	0.9906

**Experimental Design:** In the first step, we collected 10 test selection methods from existing studies (Ma et al. 2021; Feng et al. 2020; Weiss and Tonella 2022) that can be adapted for GNN misclassification detection. These approaches have been proven effective in the context of DNNs. Moreover, we also evaluated a test prioritization method specifically designed for GNNs, called GraphPrior (Dang et al. 2023), and compared its effectiveness with these DNN test prioritization methods. To provide a more comprehensive evaluation, we include not only node classification datasets but also edge classification and graph classification datasets in our analysis. Following the methodology of previous research (Feng et al. 2020), we utilized the PFD metric to evaluate the effectiveness of various test selection methods in selecting misclassified test inputs. PFD directly measures the ratio of correctly identified misclassified test inputs to the total number of misclassified tests within the test set. Hence, it provides a straightforward reflection of the effectiveness of test selection methods. A higher PFD value indicates that the evaluated test selection approach is more effective at detecting misclassified inputs. Moreover, in order to more clearly demonstrate the difference in effectiveness between the test selection method and the baseline method (random selection), we performed normalization to the experimental results (using Formula (8) Ali et al. 2014) and reported the results.

$$x_{\text{normalized}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (8)$$

where  $x$  is the original value.  $x_{\min}$  is the minimum value within all the values.  $x_{\max}$  is the maximum value within all the values.  $x_{\text{normalized}}$  is the resulting normalized value.

**Results:** The results of RQ1 are presented in Tables 1, 2, 3, and Fig. 3. Table 1 presents the effectiveness of various test selection approaches on graph datasets across three different classification tasks: node classification, edge classification, and graph classification datasets. We shaded the approach with the highest effectiveness for each case in gray. On the node classification dataset, we highlighted in bold the method that performs best among all approaches not specifically designed for GNNs.

From Table 1, we see that on the node classification datasets (i.e., Cora, CiteSeer, and PubMed), GraphPrior, specifically designed for GNNs, demonstrates the highest effectiveness across all cases. Furthermore, among all approaches not specifically designed for GNNs, Margin performs the best in the majority of cases (90% among all cases). Similarly, on the graph classification datasets, the best-performing test selection method is also Margin. On the edge classification datasets (i.e., DrugBank and BindingDB), the best-performing method is the least confidence, which performs the best across all cases.

Table 2 presents the normalization results for the sum of PFDs for all test selection methods. We utilize random selection as the baseline for normalization. Hence, the normalization results for random selection (baseline) are consistently 0 across all subjects. Detailed normalization calculation methods are provided in the experimental design of RQ1. In this context, if the value for a test selection approach is closer to 1, it indicates that the effectiveness of this test selection method is higher. The experimental results confirm the above conclusions that, on the node classification datasets, GraphPrior, specifically tailored for GNNs, performs as the most effective method in each case. Among the approaches not specifically designed for GNNs, Margin outperforms others in most instances. On graph classification datasets, Margin also performs as the top-performing test selection method. For edge classification datasets, least confidence performs as the most effective approach.

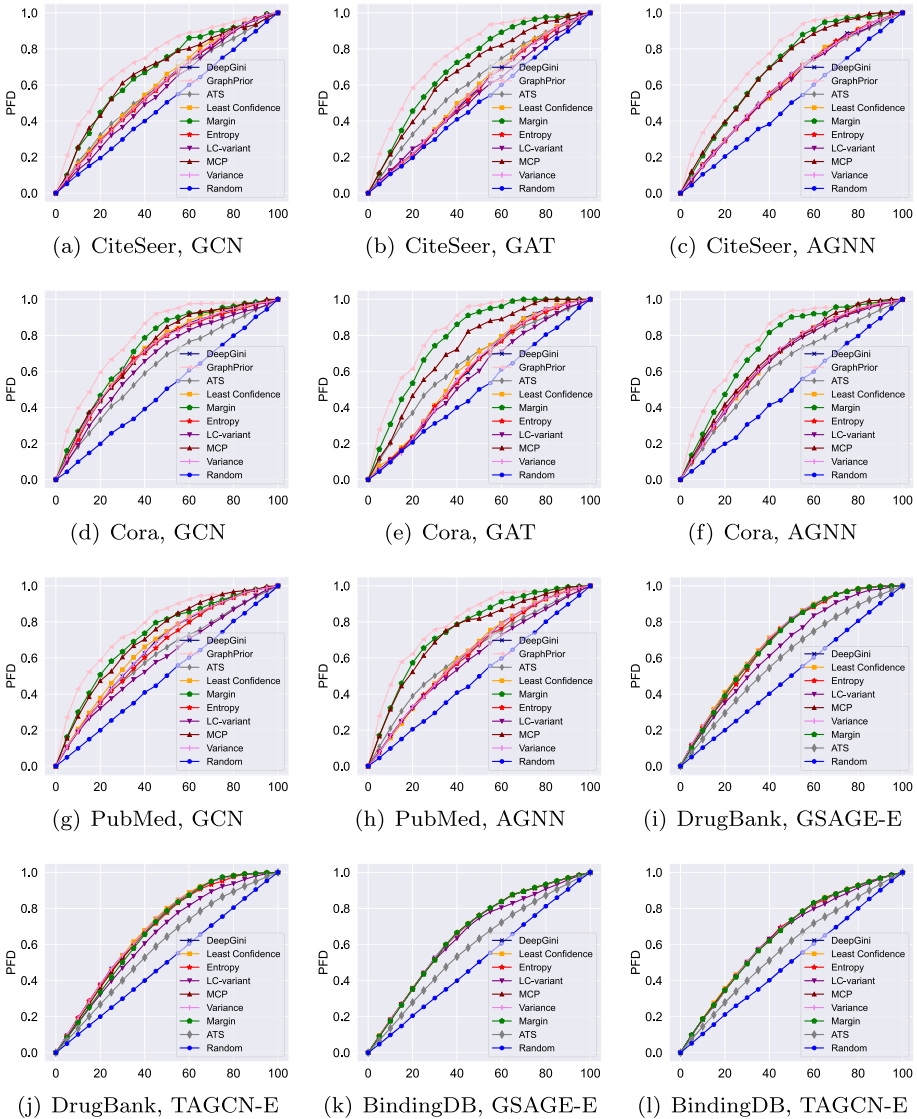
Table 3 provides a more detailed breakdown of the effectiveness of various test selection methods across different classification tasks, including the node-level, edge-level, and graph-level classification tasks. The method that performs the best in each case is still highlighted in gray, and in node classification datasets, the best-performing method among all methods

**Table 3** Effectiveness comparison of misclassification detection approaches on node and graph classification tasks, respectively

Task	Approach	Percentage of test case executed						
		10%	20%	30%	40%	50%	60%	70%
Node-Level	DeepGini	0.1855	0.3372	0.4792	0.6014	0.7080	0.7971	0.8693
	ATS	0.1833	0.3401	0.4683	0.5805	0.6726	0.7531	0.8212
	LC	0.1850	0.3403	0.4859	0.6154	0.7180	0.8029	0.8743
	Margin	<b>0.2895</b>	<b>0.5031</b>	<b>0.6474</b>	<b>0.7596</b>	<b>0.8366</b>	<b>0.8994</b>	<b>0.9367</b>
	GraphPrior	0.3963	0.5812	0.7268	0.8244	0.9015	0.9493	0.9642
	Entropy	0.1827	0.3346	0.4704	0.5928	0.6982	0.7846	0.8640
	LC-variant	0.1750	0.3152	0.4410	0.5543	0.6528	0.7409	0.8203
	MCP	0.2644	0.4669	0.6175	0.7298	0.8119	0.8681	0.9183
	Variance	0.1828	0.3370	0.4784	0.6042	0.7077	0.7966	0.8671
	Random	0.0997	0.2009	0.3014	0.4019	0.4991	0.6033	0.6993
	DeepGini	0.1743	0.3835	0.5691	0.6952	0.7977	0.8614	0.9181
	ATS	0.1723	0.3532	0.4976	0.6401	0.7612	0.8235	0.8991
	LC	0.1878	0.4054	0.5366	0.7131	0.7973	0.8651	0.9257
	Margin	0.2238	0.5498	0.6253	0.7935	0.8684	0.9251	0.9811
	Entropy	0.1781	0.4147	0.5616	0.6934	0.7836	0.8512	0.9013
	LC-variant	0.1653	0.3298	0.4746	0.6033	0.7311	0.8068	0.8906
MCP	0.2238	0.4193	0.5414	0.6625	0.7558	0.8401	0.8684	
Variance	0.1869	0.4123	0.5818	0.6953	0.7986	0.8693	0.9177	
Graph-Level	Random	0.0950	0.2158	0.3005	0.3827	0.4972	0.6088	0.6959
DeepGini	0.1944	0.3667	0.5318	0.6650	0.7762	0.8602	0.9196	
ATS	0.1414	0.2800	0.4089	0.5288	0.6382	0.7314	0.8115	
LC	0.1936	0.3733	0.5299	0.6689	0.7774	0.8636	0.9194	
Margin	0.1813	0.3560	0.5152	0.6572	0.7730	0.8585	0.9193	
Entropy	0.1934	0.3614	0.5201	0.6613	0.7732	0.8536	0.9156	
LC-variant	0.1795	0.3429	0.4940	0.6197	0.7302	0.8130	0.8784	
MCP	0.1898	0.3654	0.5256	0.6637	0.7755	0.8608	0.9194	
Variance	0.1941	0.3661	0.5312	0.6648	0.7759	0.8575	0.9192	
Edge-Level	Random	0.1015	0.2039	0.3026	0.4005	0.5032	0.6043	0.7036

not specifically designed for GNNs is also highlighted in bold. In Table 3, we see that, in the node classification datasets, the best-performing method is GraphPrior, which is specifically designed for GNNs. Among all methods not specifically designed for GNNs, Margin performs the best. In the edge-level datasets, Least Confidence and DeepGini perform the best. In graph-level datasets, Margin performs the best. This further confirms the conclusions obtained above.

However, we find that in GNNs, uncertainty-based test selection methods (such as Margin Sampling) perform less effectively compared to their performance in the context of traditional DNNs. Based on the findings from previous work (Feng et al., 2020), DeepGini can achieve a PFD of around 90% when selecting 30% of the data, which means that DeepGini can detect about 90% misclassified tests when selecting 30% of tests from the test set. However, as suggested in Fig. 3, which visually illustrates the effectiveness of different test selection



**Fig. 3** Percentage of Fault Detected (y-axis) with different test selection approaches given the ratio of tests executed (x-axis)

methods, DeepGini can only detect around 50% of misclassified tests when selecting 30% of tests in the context of GNN test selection. Even the best-performing test selection method, Margin Sampling, can only detect approximately 50% to 70% of misclassified tests, significantly lower than its performance on DNNs. Below, we analyze the reasons for the reduced performance of uncertainty-based methods.

There are four potential factors that hinder confidence-based approaches from achieving the same level of effectiveness as in DNNs. In test selection: 1) they do not account for the interdependencies among test inputs (nodes) within the GNN test set, which are crucial

**Table 4** Effectiveness of accuracy estimation approaches with respect to random selection (baseline) in terms of RMSE

Data	Model	Approach				
		GMM	Hierarchical	K-Means	K-Means Plus	MimiBatch K-Means
Cora	GCN	0.4332	0.3596	0.3912	0.3713	0.4172
	GAT	-0.0758	0.2509	0.2008	0.0977	-0.0957
	AGNN	0.2029	-0.0571	-0.0479	-0.0273	0.1537
	ARMA	-0.3282	0.1636	0.1449	-0.2729	-0.2430
	GCN	0.2617	-0.2774	-0.2906	0.2151	0.1487
CiteSeer	GAT	0.2181	0.1796	0.2217	0.1153	0.0731
	AGNN	0.2054	0.0359	-0.0089	0.2052	0.2437
	ARMA	0.0860	0.1633	0.1799	0.0878	0.2096
	GCN	0.2891	0.2037	0.2645	0.3233	0.1979
	GAT	0.2163	0.2303	0.1739	0.2597	0.2321
PubMed	AGNN	0.3540	0.3506	0.3078	0.3010	0.3603
	ARMA	0.2070	0.2177	0.1979	0.2435	0.2080
	GraphNN	0.2364	0.2566	0.2277	0.0892	0.2120
Mutagenicity	GIN	0.2773	0.2023	0.2503	0.2190	0.1766
	GraphNN	0.2582	0.1768	0.2845	0.3240	0.2798
NCII	GIN	0.1849	0.1914	0.1720	0.1341	0.1894
	GSAGE-E	-0.0394	-0.0378	-0.0365	-0.0298	-0.0422
BindingDB	TAGCN-E	0.0076	0.0067	0.0113	0.0191	-0.0013

**Table 5** Average Effectiveness of accuracy estimation approaches with respect to random selection (baseline) in terms of RMSE

Data	Model	Approach			
		GMM	Hierarchical	K-Means	K-Means Plus
Cora	GCN	0.0217	0.0180	0.0196	0.0186
	GAT	-0.0002	0.0006	0.0005	0.0002
	AGNN	0.0101	-0.0029	-0.0024	-0.0014
	ARMA	-0.0164	0.0082	0.0072	-0.0136
	GCN	0.0131	-0.0139	-0.0145	0.0108
	GAT	0.0109	0.0090	0.0111	0.0058
CiteSeer	AGNN	0.0103	0.0018	-0.0004	0.0103
	ARMA	0.0043	0.0082	0.0090	0.0044
	GCN	0.0145	0.0102	0.0132	0.0162
	GAT	0.0108	0.0115	0.0087	0.0130
	AGNN	0.0177	0.0175	0.0154	0.0150
	ARMA	0.0103	0.0109	0.0099	0.0122
PubMed	GraphNN	0.0118	0.0128	0.0114	0.0045
	GIN	0.0139	0.0101	0.0125	0.0109
	GraphNN	0.0129	0.0088	0.0142	0.0162
	GIN	0.0092	0.0096	0.0086	0.0067
	GSAGE-E	-0.0020	-0.0019	-0.0018	-0.0015
	TAGCN-E	0.0004	0.0003	0.0006	0.0010
Mutagenicity	GCN	0.0010			
	GAT	-0.0002			
	AGNN	0.0077			
	ARMA	-0.0121			
	GCN	0.0074			
	GAT	0.0037			
NCII	AGNN	0.0122			
	ARMA	0.0105			
	GCN	0.0099			
	GAT	0.0116			
	AGNN	0.0180			
	ARMA	0.0104			
BindingDB	GraphNN	0.0106			
	GIN	0.0088			
	GraphNN	0.0140			
	GIN	0.0095			
	GSAGE-E	-0.0021			
	TAGCN-E	-0.0001			

for GNN model inference. Confidence-based prioritization approaches typically function on test sets where each test is treated as independent; 2) Irregular Data: Graph data is typically irregular, with varying numbers of connections and neighbor nodes for each node. This irregularity adds complexity to the application of confidence-based approaches to graphs, making it potentially challenging to effectively capture this complexity; 3) Local and Global Dependencies: Graph data typically exhibit both local and global dependencies. Node attributes and connections can introduce complexity to confidence-based methods since it is challenging to capture these multi-scale dependencies; 4) Size and complexity of graphs. Graphs can exhibit different sizes and complexities. Confidence-based methods can be affected when applied to graph datasets of different sizes and complexities.

**Answer to RQ1:** *When applied to GNNs (including tasks such as node classification, edge classification, and graph classification), uncertainty-based test selection methods (such as Margin and DeepGini), as well as ATS, do not demonstrate the same level of effectiveness as they exhibited in DNNs.*

### 5.2 RQ2: Test Selection for GNN Accuracy Estimation

**Objectives:** We evaluate the effectiveness of various clustering methods that utilize the model’s confidence vector in estimating the accuracy of the GNN test set.

**Experimental Design:** In the initial step, we selected five widely recognized clustering algorithms. For each test instance in the test set, we obtain the model’s prediction probability vector, which can reflect the model’s confidence in its predictions. We call this vector the confidence vector. Following this, we utilize each clustering algorithm to group these instances based on their respective confidence vectors. Subsequently, we select  $N$  central points from each cluster. These chosen test instances form a subset of the original test set and can then be employed to predict the overall accuracy of the entire test set.

**Results:** The results pertaining to RQ2 are presented in Tables 4, 5, 6, and Fig. 4. Specifically, Table 4 exhibits the effectiveness of different test selection approaches related to random selection. In Table 4, the values represent the effectiveness of each test selection approach relative to random selection. Specifically, the calculation process is illustrated in Formula (9). It is important to note that when using RMSE values to measure effectiveness, a smaller RMSE implies higher effectiveness for a given test selection method. Therefore, in Formula (9), if the *diff* for a test selection method  $TS$  is positive, it indicates that the sum of RMSE values for  $TS$  is lower than that of random selection, suggesting that the effectiveness of  $TS$  is higher than random selection. In the case where  $TS$ ’ *diff* is positive, if the *diff* is larger, it indicates that the RMSE values of  $TS$  compared to those of random selection are smaller. Since smaller RMSE implies higher effectiveness, it suggests that the effectiveness of  $TS$  relative to random selection is higher.

$$diff = \sum_{r=10}^{100} (RMSE_{Random}^r - RMSE_{TS}^r) \tag{9}$$

where  $r$  represents the number of tests selected. For example, if  $r = 80$ , it indicates that 80 tests are selected from the test set.  $RMSE_{TS}^r$  refers to the effectiveness (measured by RMSE) of the test selection approach  $TS$  when selecting  $r$  test inputs.  $RMSE_{Random}^r$  refers to the effectiveness (measured by RMSE) of random selection when selecting  $r$  test inputs.



**Table 6** Effectiveness comparison among accuracy estimation approaches on node, graph, and edge classification, respectively

Task	Approach	Number of Selected Test Inputs										
		10	20	30	40	50	60	70	80	90	100	
Node-Level	GMM	0.0908	0.0719	0.0621	0.0515	0.0473	0.0444	0.0385	0.0406	0.0418	0.0389	
	Hierarchical	0.1015	0.0711	0.0598	0.0543	0.0501	0.0384	0.0410	0.0392	0.0363	0.0344	
	K-Means	0.1022	0.0739	0.0638	0.0529	0.0456	0.0456	0.0383	0.0370	0.0345	0.0349	
	K-Means Plus	0.1041	0.0715	0.0599	0.0510	0.0462	0.0395	0.0399	0.0405	0.0416	0.0386	
	MiniBatch K-Means	0.0987	0.0727	0.0624	0.0544	0.0438	0.0467	0.0422	0.0404	0.0338	0.0366	
	Random	0.1241	0.0851	0.0724	0.0618	0.0557	0.0475	0.0454	0.0416	0.0439	0.0394	
	GMM	0.0850	0.0685	0.0531	0.0464	0.0376	0.0403	0.0359	0.0363	0.0359	0.0281	
	Hierarchical	0.1152	0.0544	0.0501	0.0521	0.0555	0.0357	0.0372	0.0333	0.0323	0.0364	
	K-Means	0.0942	0.0672	0.0533	0.0556	0.0421	0.0385	0.0335	0.0350	0.0341	0.0358	
	K-Means Plus	0.0998	0.0773	0.0537	0.0483	0.0483	0.0410	0.0366	0.0341	0.0423	0.0254	
Graph-Level	MiniBatch K-Means	0.0967	0.0765	0.0525	0.0452	0.0417	0.0442	0.0382	0.0369	0.0245	0.0299	
	Random	0.1253	0.0897	0.0716	0.0586	0.0550	0.0512	0.0449	0.0407	0.0428	0.0411	
	GMM	0.3225	0.1826	0.1810	0.1593	0.1387	0.1294	0.1458	0.0891	0.0792	0.0611	
	Hierarchical	0.2662	0.1771	0.1939	0.1854	0.1286	0.1340	0.1397	0.0930	0.0825	0.0722	
	K-Means	0.2404	0.1720	0.1885	0.1387	0.1533	0.1550	0.1494	0.1148	0.0744	0.0639	
	K-Means Plus	0.1446	0.1631	0.1434	0.1426	0.1507	0.1607	0.1346	0.1365	0.0791	0.0746	
	MiniBatch K-Means	0.2126	0.2465	0.1625	0.1558	0.1819	0.1595	0.1657	0.1185	0.0893	0.0852	
	Random	0.2326	0.1706	0.1351	0.1164	0.1047	0.0974	0.0903	0.0755	0.0522	0.0517	
	Edge-Level	GMM	0.0908	0.0719	0.0621	0.0515	0.0473	0.0444	0.0385	0.0406	0.0418	0.0389
		Hierarchical	0.1015	0.0711	0.0598	0.0543	0.0501	0.0384	0.0410	0.0392	0.0363	0.0344
K-Means		0.1022	0.0739	0.0638	0.0529	0.0456	0.0456	0.0383	0.0370	0.0345	0.0349	
K-Means Plus		0.1041	0.0715	0.0599	0.0510	0.0462	0.0395	0.0399	0.0405	0.0416	0.0386	
MiniBatch K-Means		0.0987	0.0727	0.0624	0.0544	0.0438	0.0467	0.0422	0.0404	0.0338	0.0366	
Random		0.1241	0.0851	0.0724	0.0618	0.0557	0.0475	0.0454	0.0416	0.0439	0.0394	
GMM		0.0850	0.0685	0.0531	0.0464	0.0376	0.0403	0.0359	0.0363	0.0359	0.0281	
Hierarchical		0.1152	0.0544	0.0501	0.0521	0.0555	0.0357	0.0372	0.0333	0.0323	0.0364	
K-Means		0.0942	0.0672	0.0533	0.0556	0.0421	0.0385	0.0335	0.0350	0.0341	0.0358	
K-Means Plus		0.0998	0.0773	0.0537	0.0483	0.0483	0.0410	0.0366	0.0341	0.0423	0.0254	

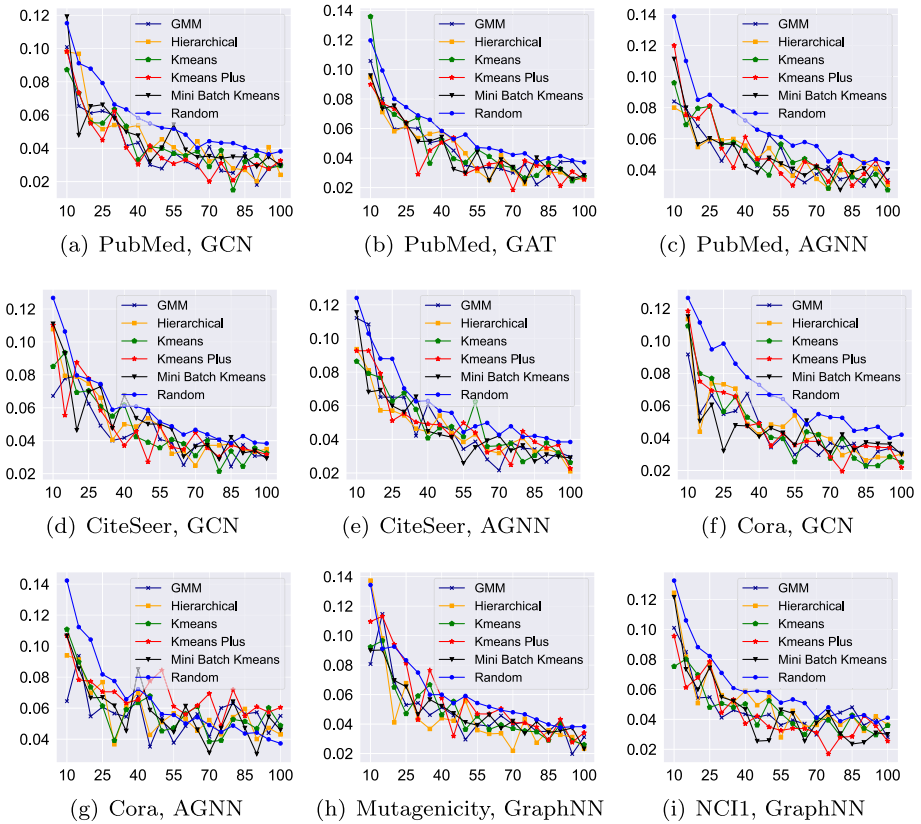


Fig. 4 Root Mean Squared Errors(y-axis) of different test selection approaches given the number of tests selected (x-axis)

In Table 4, we see that, on node classification datasets (i.e., Cora, CiteSeer, and PubMed), the clustering-based test selection methods perform better than random selection in the majority of cases (85%). Similarly, on graph classification datasets (Mutagenicity and NCI1), the clustering methods consistently perform better than random selection. Table 6 further illustrates the effectiveness of different clustering-based test selection approaches, with the best-performing method highlighted in gray for each case. In Table 6, the “Number of Selected Test Inputs” indicates the number of tests selected from the test set. We see that, across both node classification and graph classification tasks, clustering-based test selection methods consistently perform the best.

However, the improvement achieved by clustering-based test selection methods compared to random selection is marginal. For example, when selecting ten tests, in terms of RMSE, the best clustering-based method only exceeds random selection by approximately 0.03, and when selecting 80 tests, the best clustering-based method only surpasses random selection by around 0.01. Similarly, Fig. 4 visually confirms these conclusions, with the blue line representing the baseline (i.e., random selection). We see that while all clustering methods are effective in most cases, the improvements achieved are slight. Moreover, Table 5 presents the average differences between all test selection methods and random selection across all cases. The difference values are all around 0.01. Hence, we conclude that clustering-based

methods are effective in selecting representative test data for node classification and graph classification datasets but achieve limited improvements.

In the above, we analyzed the effectiveness of the clustering-based test selection approach in node classification and graph classification tasks. Next, we focus on the effectiveness of clustering methods in edge classification datasets (BindingDB). In Table 4, we see that, on edge classification datasets, the clustering-based test selection approaches perform better than random selection in 40% of cases. Furthermore, Table 6 further highlights the effectiveness of clustering methods across different classification tasks. We see that, in edge-level tasks, random selection exhibits better performance in most cases. From selecting 30 tests and 40 tests up to selecting 100 tests, random selection consistently shows the best performance. This implies that, in the majority of cases, the clustering-based test selection method does not perform as well as random selection on edge classification datasets. In the following, we analyze the potential reasons:

In graph datasets, since a node can be connected to multiple other nodes, the number of edges can far exceed the number of nodes, making the information on edges more complex and diverse, leading to uneven data distribution. Clustering algorithms typically aim to group data points into collections with higher similarities. However, when the data distribution is uneven, clustering algorithms can have difficulty effectively assigning data to the correct clusters. This leads to poor performance when using clustering-based test selection methods in edge classification tasks.

**Answer to RQ2:** *In node classification and graph classification tasks, all clustering-based test selection methods perform better than random selection in most cases, but their improvements relative to random selection are slight. On edge classification tasks, clustering-based test selection methods do not perform better than random selection in most cases.*

### 5.3 RQ3: Confidence-Based Test Selection for GNN Performance Enhancement

**Objectives:** We evaluate the effectiveness of various test selection methods derived from the two aforementioned research questions in selecting informative retraining inputs to enhance GNN model performance. Specifically, these methods correspond to the test selection approaches for misclassification detection (RQ1) and accuracy estimation (RQ2).

**Experimental Design:** In previous research questions, we assessed multiple test selection methods tailored for misclassification detection and accuracy estimation. In this research question, we apply these methods to select tests for the retraining of the original GNN model, with the objective of improving its prediction accuracy.

The steps and methods we employed for retraining follow the existing study of DNN test selection (Ma et al. 2021). In the initial phase, given a GNN model  $M$  and a graph dataset, we partition the dataset into a training set, a candidate set, and a test set. The test set remains untouched throughout the process. First, we train an initial GNN model using the training set and record the initial accuracy of  $M$  on the test set. Subsequently, we apply various test selection methods to select different subsets of data from the candidate set. We then utilize the selected test data to retrain the original GNN model, recording the model's accuracy after each retraining. By observing the improvement in model accuracy after retraining with data selected using different test selection methods, we can assess and compare the effectiveness of these data selection methods.

**Table 7** Effectiveness of test selection approaches with respect to random selection (baseline) in selecting retraining inputs to improve GNN accuracy

Approach Data	Model	GMM	Hierarchical	K-Means	Spectrum	GraphPrior	ATS	DeepGini	LC	Margin	Entropy	Variance	MCP
Cora	GCN	0.3741	0.2778	0.3704	<b>0.4926</b>	0.2094	0.3759	0.2333	-0.1296	0.2593	0.2259	0.3815	0.4667
	GAT	0.2037	0.2185	0.1926	-0.0333	<b>0.4004</b>	0.1629	0.1185	-0.1741	0.3481	0.0444	0.1333	0.2815
	AGNN	-0.3778	-0.4407	-0.6593	-0.5333	0.0449	-0.4426	0.0815	-0.6667	<b>0.1778</b>	0.1185	-0.2259	-0.0481
	ARMA	0.0889	0.0111	-0.1296	-0.0111	<b>0.2561</b>	-0.0278	0.0815	-0.1407	0.2148	0.0556	0.0741	0.1667
CiteSeer	GCN	0.0482	0.0000	-0.1777	0.0120	0.1191	-0.1009	-0.1777	-0.4518	0.0693	0.0090	-0.0241	<b>0.1235</b>
	GAT	-0.0271	0.0090	0.0030	-0.0542	<b>0.2327</b>	0.0256	0.0904	-0.1596	0.1837	0.0572	0.0813	0.2319
	AGNN	-0.1536	-0.0723	-0.1084	-0.1265	<b>-0.0293</b>	-0.1024	-0.0331	-0.2289	-0.0873	-0.0813	-0.1024	-0.0904
	ARMA	<b>0.0723</b>	-0.0331	-0.0663	-0.0873	-0.0654	-0.0632	-0.0151	-0.1265	-0.1114	-0.0843	-0.0361	-0.0512
PubMed	GCN	-	-	-	-	0.1402	0.0426	0.0396	-0.0573	0.0873	0.0553	0.1167	<b>0.3151</b>
	GAT	-	-	-	-	0.1057	0.0253	0.0223	0.0061	0.0492	0.007	0.1451	<b>0.2055</b>
	AGNN	-	-	-	-	<b>0.0275</b>	-0.0621	-0.0147	-0.1096	-0.1243	-0.0096	-0.3125	-0.2760
MNIST	ARMA	-	-	-	-	<b>0.1255</b>	0.0352	0.0208	-0.0436	0.0680	0.0639	0.0213	0.0685
	NN	0.0612	0.0750	-0.0012	0.0362	<b>0.1022</b>	0.0243	0.0888	-0.0400	0.0562	0.0800	0.0500	0.0175
MNIST	GRACLUS	0.2340	0.1180	0.0760	0.2220	0.2099	0.125	0.1300	0.0280	<b>0.2500</b>	0.0720	0.1760	-0.0040

The bold entries represent the best-performing method in each case

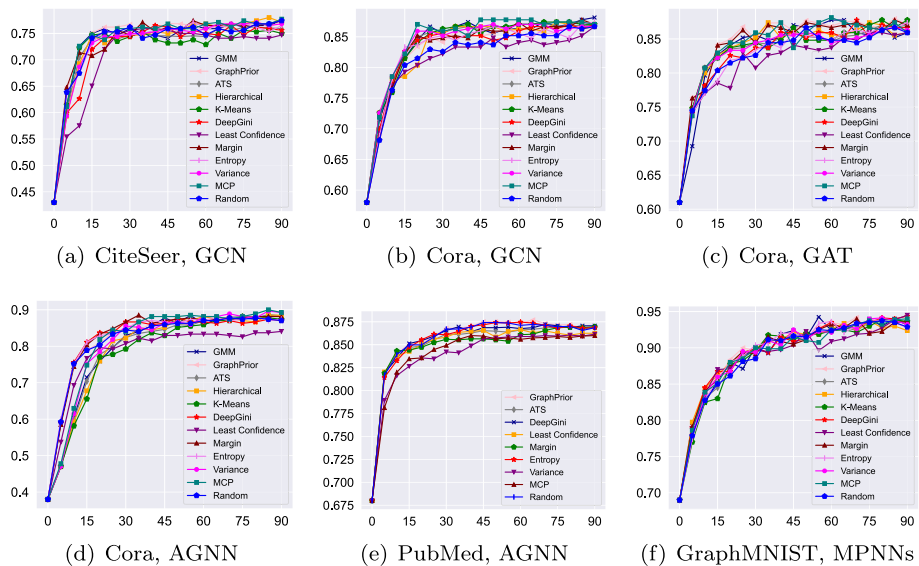
Additionally, in the retraining experiments (RQ3), the initial accuracy of the utilized GNN models are: on the CiteSeer dataset: 40% to 60%, on the PubMed dataset: 65% to 70%, on the Cora dataset: 35% to 65%, and on the GraphMNIST dataset: 65% to 70%. The evaluated models' original accuracy range follows the work of Hu et al. (2021).

**Results:** The experimental results for RQ3 are presented in Table 7 and Fig. 5. Table 7 presents the effectiveness of all test selection methods in terms of their relative improvement or decline compared to the baseline method (i.e., random selection). We calculate the improvement of each test selection method relative to random selection using Formula (10). In Table 7, if a test selection method outperforms random, its value is positive and highlighted in gray; conversely, if it performs worse, its value is negative and highlighted in white.

$$imp = \sum_{i=1}^{steps} (Acc_{TS}^i - Acc_{Random}^i) \tag{10}$$

where *steps* represent the total number of retraining steps.  $Acc_{TS}^i$  refers to the accuracy of the model retrained using the data selected by the test selection metric *TS*.  $Acc_{Random}^i$  refers to the accuracy of the model retrained using the data selected by the random selection approach. *imp* represents the effectiveness improvement of the test selection metric *TS* over random selection.

In Table 7, we see that some test selection methods, such as DeepGini and MCP, along with GraphPrior, perform better than the baseline (random selection) in the majority of cases. Specifically, in approximately 61% of the cases, the evaluated test selection methods perform better than random selection. The top three best-performing methods are GraphPrior, Margin Sampling, and MCP. GraphPrior achieves the best performance in 50% of the cases, MCP in 21.43% of the cases, and Margin Sampling leads in 14.29% of the cases.



**Fig. 5** Test accuracy (y-axis) achieved by different data selection approaches given the percentage of retrain data selected (x-axis)

However, despite improvements, the extent to which test selection methods improve GNN model accuracy compared to the baseline is slight. Figure 5 offers a visual representation of the effectiveness of various test selection methods. In this figure, the blue line denotes the baseline - random selection. We see that the majority of uncertainty-based test selection methods, as well as GraphPrior, only show minor improvements over the baseline (random selection), with some methods even performing worse than random selection. However, a previous study on DNN test selection (Hu et al. 2021) demonstrated that some uncertainty-based test selection metrics, such as Margin and MCP, can consistently exhibit strong performance. However, these metrics do not achieve consistently strong performance in GNN test selection.

Below, we provide some potential reasons why some test selection methods (e.g., margin and MCP) are effective in DNNs but exhibit only small improvements over the baseline approach when applied to GNNs.

- **Inadequate representativeness** The inputs selected through test selection methods aimed at misclassification detection are typically samples that are more likely to be misclassified. These inputs can be specific in the feature space, representing only a small part of the data distribution. They cannot be sufficient to represent the complex structure and diversity of the entire graph, thus affecting the effectiveness of retraining.
- **Differences in data structure** DNNs typically process data where each sample is independent of others, and retraining the model does not require considering the relationships between samples. In contrast, in graph data processed by GNNs, nodes (i.e., samples) are interconnected through edges. Therefore, the information of a node depends not only on its own features but also on its neighboring nodes and the overall structure of the graph. When test selection methods from DNNs are applied to GNNs, these methods cannot adequately capture and utilize the complex interdependence of graph data for retraining.
- **Differences in Learning Mechanisms** GNNs update node representations by aggregating information from neighboring nodes, which differs from the working mechanism of DNNs. Therefore, the reason for the misclassification of a node can be not only due to the features of the node itself but could also involve information from its neighboring nodes. Simply selecting these misclassified inputs for retraining ignores the crucial information from their neighbors.

**Answer to RQ3:** *The effectiveness of both confidence-based and clustering-based test selection methods in improving GNN model accuracy through the selection of retraining data shows only slight enhancements when compared to random selection, despite some methods having been demonstrated to be effective in DNNs.*

## 5.4 RQ4: Node Importance-Based Test Selection for GNN Performance Enhancement

**Objectives:** We assess the effectiveness of node importance-based test selection methods in improving GNN accuracy during retraining. This investigation is driven by several factors: 1) Nodes with high importance typically encapsulate critical information and have a more significant impact on the overall graph (Park et al. 2019). Consequently, these nodes are more likely to capture essential information that is crucial for enhancing model performance; 2) Unimportant nodes may contain noise or irrelevant data that could introduce interference during retraining, potentially leading to a reduction in model performance; 3) Node importance is a distinctive data feature in GNNs that can be leveraged for the selection of critical tests. Currently, there is a gap in research regarding whether node importance can effectively

guide the selection of retraining inputs. Therefore, it is imperative to conduct relevant studies in this area.

**Experimental Design:** In the initial step, we evaluated the initial accuracy of the target GNN model. Subsequently, we ranked all tests in the test set by importance, using each node importance metric. Based on each metric, we selected the top important tests, ranging from 10% to 80%, and then proceeded to retrain the original GNN model. We recorded the model’s accuracy after each round of retraining.

**Results:** The experimental results for RQ4 are presented in Table 8. Here, we have shaded in gray the approach with the highest effectiveness for each case. We see that, in the majority of cases, test selection methods based on node importance exhibit limitations when selecting inputs for retraining GNN models to improve accuracy. These methods tend to perform less effectively than random selection. Specifically, random selection outperforms node importance-based methods in 75% of the cases. Conversely, node importance-based test selection methods excel in only the remaining 25% of cases. Some potential factors that can lead to the low performance of node importance-based methods include:

- **Lack of Diversity** Node importance methods can select a group of similar or closely related nodes, potentially resulting in a lack of diversity in the selected data. In contrast, randomly selecting nodes can introduce greater diversity, thereby enhancing the model’s ability to generalize.
- **Overfitting** If the nodes selected by node-importance methods are overly specific or concentrated in a particular area, the model can be prone to overfitting to these selected nodes. Randomly selected nodes, on the other hand, can provide a more varied set of information, contributing to mitigate overfitting
- **Noise Tolerance** Occasionally, incorporating some noisy or less significant nodes can potentially enhance the model’s robustness. Randomly selected nodes can introduce such beneficial noise.

**Answer to RQ4:** *Node importance-based test selection methods are not suitable for selecting retraining data to improve GNN accuracy, and in many cases, they even perform worse than random selection.*

**Table 8** Effectiveness of node importance-based test selection approaches with respect to random selection (baseline) in selecting retraining inputs to improve GNN accuracy

Approach	Percentage of test case executed							
	10%	20%	30%	40%	50%	60%	70%	80%
BC	0.8032	0.8060	0.8087	0.8100	0.8171	0.8225	0.8250	0.8265
Center	0.8002	0.8035	0.8042	0.8066	0.8090	0.8135	0.8173	0.8209
Degree	0.7999	0.8002	0.8041	0.8108	0.8136	0.8201	0.8209	0.8237
EC	0.7994	0.7995	0.8020	0.8061	0.8117	0.8136	0.8144	0.8201
Eccentricity	<b>0.8034</b>	0.8051	0.8040	0.8069	0.8094	0.8131	0.8171	0.8216
Hits	0.7999	0.7995	0.8011	0.8080	0.8118	0.8132	0.8120	0.8175
PageRank	0.8028	<b>0.8074</b>	0.8091	0.8157	0.8190	0.8203	0.8262	0.8297
Random	0.8029	0.8061	<b>0.8164</b>	<b>0.8189</b>	<b>0.8259</b>	<b>0.8328</b>	<b>0.8399</b>	<b>0.8426</b>

## 6 Threats to Validity

*Threats to Internal Validity.* The internal threats to validity primarily stem from the implementation of the evaluated test selection approaches. To mitigate this threat, we implemented these approaches using the widely adopted PyTorch library and utilized the implementations of the compared approaches as provided by their respective authors. Another internal threat arises from the selection of clustering algorithms. The effectiveness of test selection can be influenced by the performance of the selected clustering algorithm. To mitigate this threat, we utilized established frameworks in our study. We opted for the widely adopted scikit-learn framework (Pedregosa et al. 2011) to implement the clustering algorithm. Scikit-learn is renowned for its robust performance and extensive user community.

*Threats to External Validity.* The primary external threats to the validity of our study are closely linked to two key aspects: the GNN models under evaluation and the test datasets used in our research. These factors can significantly impact the generalizability and applicability of our findings. To mitigate these potential threats, we made a conscious effort to include a large and diverse set of subjects (pairs of datasets and models) in our study. These subjects represent different combinations of GNN models and test datasets, ensuring that our analysis covers a wide spectrum of scenarios. Firstly, we recognized the critical role of dataset diversity and comprehensiveness in evaluating the efficacy of test selection approaches. We utilized seven prevalent graph datasets, encompassing not only node classification datasets but also graph classification datasets. This deliberate selection allows us to account for various problem domains, thereby enhancing the robustness and adaptability of our study to a multitude of GNN applications. Beyond dataset diversity, the choice of GNN models is pivotal in gaining insights into how test selection methods interact with different model architectures. To this end, we utilized a set of eight distinct GNN models, each possessing its unique characteristics and capabilities. These models span a spectrum of complexity and sophistication, ranging from simpler models to more advanced ones.

## 7 Related Work

We present the related works from three perspectives: DNN test selection, DNN Testing, and Empirical study on active learning.

### 7.1 DNN Test Selection

To tackle the challenge of labeling costs, test selection (Aghababaeyan et al. 2023b) has emerged as a practical solution.

In terms of misclassification detection, Ma et al. (2021) conducted an evaluation of various test selection methods tailored for misclassification detection, including coverage-based, surprise adequacy-based, and confidence-based approaches. Experimental results demonstrated that confidence-based metrics exhibit a robust ability to identify misclassified inputs, surpassing both the surprise adequacy-based and coverage-based test selection approaches. Hu et al. (2021) conducted an empirical evaluation of 15 active learning metrics to determine their effectiveness in selecting inputs for retraining DNNs. Their research demonstrated that the choice of data selection metrics can significantly influence the quality of the resulting model when using active learning for training.

Kim et al. (2019) proposed the Surprise Adequacy Criteria (SADL) for DNN test selection. SADL operates by extracting intermediate outputs from both the test and training data



of DNNs, treating them as features, and then evaluating the surprise adequacy based on the dissimilarity between these features. In this process, two measurements are utilized: Likelihood-based Surprise Adequacy (LSA) and Distance-based Surprise Adequacy (DSA). LSA employs kernel density estimation to compute the dissimilarity, while DSA directly utilizes Euclidean distance. Despite the effectiveness of SADL in the context of DNNs, SADL cannot be directly applied to GNNs. This is because implementing SADL requires measuring the distance between the targeted test inputs and training inputs. However, their method for measuring distance is specifically designed for image/text data, which cannot be directly applied to graph-structured data.

Wang et al. (2021) proposed PRIMA for DNN test prioritization, which identified and prioritizes potentially misclassified test inputs based on intelligent mutation analysis. Despite its effectiveness in the context of DNNs, PRIMA is not suitable for GNNs. This is because PRIMA's mutation operators are not adapted to graph-structured data and GNN models.

In terms of accuracy estimation, Li et al. (2019) introduced the CES (Cross Entropy-based Sampling) method to tackle this challenge. CES accomplishes test selection by minimizing the cross-entropy between the selected subset and the original test set, ensuring that the distribution of the selected test inputs closely matches that of the original test set. Chen et al. (2020) proposed the PACE, which employs a range of techniques to perform test selection, including clustering, prototype selection, and adaptive random testing. The process begins by categorizing all test inputs into different groups based on their testing characteristics. Subsequently, PACE utilizes the MMD-critic algorithm (Kim et al. 2016) to identify prototype test inputs from each group. For test inputs that do not fit into any specific group, PACE employs adaptive random testing to select representative tests.

Our empirical study focuses on evaluating test selection approaches across four areas: 1) Misclassification Detection, 2) Accuracy Estimation, 3) Performance Enhancement guided by confidence-based approaches, and 4) Performance Enhancement guided by node importance-based approaches.

Regarding misclassification detection and performance enhancement, our emphasis has been on evaluating confidence-based approaches due to the following reasons: 1) prior studies (Ma et al. 2021) have demonstrated that confidence-based methods outperform coverage-based and surprise-based approaches in terms of effectiveness; 2) Confidence-based test selection methods are widely recognized as the most efficient and straightforward to implement (Weiss and Tonella 2022), with runtime of less than 1 second in most cases.

## 7.2 Deep Neural Network Testing

In addition to test selection, the field of DNN testing (Jahangirova and Tonella 2020; Zolfagharian et al. 2023; Aghababaeian et al. 2023a) encompasses various noteworthy research directions, with one notable focus being the assessment of DNN adequacy. Pei et al. (2017) introduced the concept of "neuron coverage" as a metric for gauging the comprehensiveness of a test set in terms of its coverage of a DNN model's logic. They employed this metric to propose a white-box testing framework tailored for DNNs. In a subsequent study, Ma et al. (2018) introduced DeepGauge, a set of coverage criteria designed to evaluate the adequacy of tests applied to DNNs. DeepGauge also placed significant emphasis on neuron coverage as a valuable indicator of test input effectiveness. Additionally, they introduced novel metrics with varying levels of granularity to distinguish between adversarial attacks and legitimate test data. Kim et al. (2019) contributed to this area by introducing "surprise adequacy" as a measure for testing DL models. This approach evaluates the effectiveness of a test input by quantifying the surprise it generates concerning the training set. Specifically,

the surprise of a test input is determined by measuring the difference in the activation values of neurons when exposed to this new test input.

### 7.3 Empirical Study on Active Learning

Active learning has been a subject of extensive research in recent years, with empirical studies spanning various domains. Yu et al. (2018) conducted empirical research that focused on active learning techniques for literature reviews. In their work, they cataloged and refined three state-of-the-art active learning methods derived from evidence-based medicine and legal electronic discovery. This effort led to the development of a novel active learning approach designed for the analysis of large document corpora, incorporating and fine-tuning the most effective active learning algorithms. Chen et al. (2006) delved into the effectiveness of active learning in the context of word sense disambiguation. They examined the behavior of active learning by considering two fundamental data selection metrics: entropy and margin. Sassano (2002) explored the practical application of active learning with Support Vector Machines in a challenging natural language processing task, providing insights into its performance in complex scenarios. Furthermore, Weiss and Tonella (2022) conducted a comprehensive investigation into various active learning techniques, revealing that confidence-based methods delivered surprisingly strong results when applied to DNNs.

## 8 Conclusion

In this paper, we conducted a comprehensive empirical study to explore the limitations of test selection approaches in the context of GNNs. We totally evaluated 22 test selection approaches based on 7 graph datasets and 8 GNN models. The results reveal that test selection approaches do not exhibit the same level of effectiveness when applied to GNNs in comparison to DNNs. More specifically, we draw the following conclusions: 1) Confidence-based test selection methods, which perform well in DNNs, do not yield the same level of effectiveness in detecting potentially misclassified tests for GNNs; 2) In the majority of cases, clustering-based test selection methods that utilize the model's confidence vector perform better than random selection. However, their improvements compared to random selection are slight; 3) In terms of performance enhancement, both confidence-based and clustering-based test selection methods show only slight effectiveness; 4) Node importance-based test selection methods are unsuitable for selecting retraining data to enhance GNN accuracy.

**Acknowledgements** This work is supported by the Luxembourg National Research Fund AFR PHD 17036341.

**Data availability** The datasets and code used in the present study are available in our repository: [https://github.com/BlueBerry-xueqi/graph\\_testing](https://github.com/BlueBerry-xueqi/graph_testing)

## Declarations

**Conflict of Interest Statement** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence,

and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Aghababaeian Z, Abdellatif M, Briand L, Ramesh S, Bagherzadeh M (2023a) Black-box testing of deep neural networks through test case diversity. *IEEE Trans Softw Eng*, IEEE
- Aghababaeian Z, Abdellatif M, Dadkhah M, Briand L (2023b) Deepgd: A multi-objective black-box test selection approach for deep neural networks. [arXiv:2303.04878](https://arxiv.org/abs/2303.04878)
- Ahmed M, Seraj R, Islam SMS (2020) The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, MDPI 9(8):1295
- Ali PJM, Faraj RH, Koya E, Ali PJM, Faraj RH (2014) Data normalization and standardization: a technical report. *Machine Learning Technical Reports* 1(1):1–6
- Ando H, Bell M, Kurauchi F, Wong KI, Cheung KF (2021) Connectivity evaluation of large road network by capacity-weighted eigenvector centrality analysis. *Transportmetrica A: Transport Science*, Taylor & Francis 17(4):648–674
- Arthur D, Vassilvitskii S (2007) K-means++ the advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, ACM New York, NY, USA, pp 1027–1035
- Bianchi FM, Grattarola D, Livi L, Alippi C (2021) Graph neural networks with convolutional arma filters. *IEEE Trans Pattern Anal Mach Intell*, IEEE 44(7):3496–3507
- Bongini P, Bianchini M, Scarselli F (2021) Molecular generative graph neural networks for drug discovery. *Neurocomputing*, Elsevier 450:242–252
- Cai H, Zheng VW, Chang KCC (2018) A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans Knowl Data Eng*, IEEE 30(9):1616–1637
- Chen J, Schein A, Ungar L, Palmer M (2006) An empirical study of the behavior of active learning for word sense disambiguation. In: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, ACM New York, NY, pp 120–127
- Chen J, Wu Z, Wang Z, You H, Zhang L, Yan M (2020) Practical accuracy estimation for efficient deep neural network testing. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, ACM New York, NY, USA 29(4):1–35
- Cheng X, Wang H, Hua J, Xu G, Sui Y (2021) Deepwukong: Statically detecting software vulnerabilities using deep graph neural network. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, ACM New York, NY, USA 30(3):1–33
- Cheng X, Zhang G, Wang H, Sui Y (2022) Path-sensitive code embedding via contrastive learning for software vulnerability detection. In: *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, ACM New York, NY, USA, pp 519–531
- Dang X, Li Y, Papadakis M, Klein J, Bissyandé TF, Le Traon Y (2023) Graphprior: mutation-based test input prioritization for graph neural networks. *ACM Trans Softw Eng Methodol*, ACM New York, NY, USA 33(1):1–40
- Dang X, Li Y, Papadakis M, Klein J, Bissyandé TF, Le Traon Y (2024) Test input prioritization for machine learning classifiers. *IEEE Transactions on Software Engineering*, IEEE
- Du J, Zhang S, Wu G, Moura JM, Kar S (2017) Topology adaptive graph convolutional networks. [arXiv:1710.10370](https://arxiv.org/abs/1710.10370)
- Duvenaud DK, Maclaurin D, Iparraguirre J, Bombarell R, Hirzel T, Aspuru-Guzik A, Adams RP (2015) Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, ACM New York, NY, p 28
- Dwivedi VP, Joshi CK, Luu AT, Laurent T, Bengio Y, Bresson X (2020) Benchmarking graph neural networks. [arXiv:2003.00982](https://arxiv.org/abs/2003.00982)
- Elbaum S, Malishevsky AG, Rothermel G (2002) Test case prioritization: A family of empirical studies. *IEEE Trans Softw Eng*, IEEE 28(2):159–182
- Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, Yin D (2019) Graph neural networks for social recommendation. *The world wide web conference*. ACM New York, NY, pp 417–426
- Feng Y, Shi Q, Gao X, Wan J, Fang C, Chen Z (2020) 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*, ACM New York, NY, pp 177–188

- Fu X, Zhang J, Meng Z, King I (2020) Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In: Proceedings of The Web Conference 2020, ACM New York, NY, pp 2331–2341
- Gao X, Feng Y, Yin Y, Liu Z, Chen Z, Xu B (2022) Adaptive test selection for deep neural networks. In: Proceedings of the 44th International Conference on Software Engineering, IEEE, pp 73–85
- Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: International conference on machine learning, PMLR, pp 1263–1272
- Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. *Advances in neural information processing systems*, Curran Associates, p 30
- Haq FU, Shin D, Nejati S, Briand L (2021) Can offline testing of deep neural networks replace their online testing? a case study of automated driving systems. *Empirical Software Engineering*, Springer, 26(5):90
- He X, Deng K, Wang X, Li Y, Zhang Y, Wang M (2020) Lightgcn: Simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, ACM New York, NY, pp 639–648
- Hong D, Gao L, Yao J, Zhang B, Plaza A, Chanussot J (2020) Graph convolutional networks for hyperspectral image classification. *IEEE Trans Geosci Remote Sens*, IEEE 59(7):5966–5978
- Hu P, Fan W, Mei S (2015) Identifying node importance in complex networks. *Physica A: Statistical Mechanics and its Applications*, Elsevier 429:169–176
- Hu Q, Guo Y, Cordy M, Xie X, Ma W, Papadakis M, Le Traon Y (2021) Towards exploring the limitations of active learning: An empirical study. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, pp 917–929
- Jahangirova G, Tonella P (2020) An empirical evaluation of mutation operators for deep learning systems. 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), IEEE, pp 74–84
- Jha K, Saha S, Singh H (2022) Prediction of protein-protein interaction using graph neural networks. *Scientific Reports*, Nature Publishing Group UK London 12(1):8360
- Jin W, Ma Y, Liu X, Tang X, Wang S, Tang J (2020) Graph structure learning for robust graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, ACM New York, NY, pp 66–74
- Kaushik M, Mathur B (2014) Comparative study of k-means and hierarchical clustering techniques. *International Journal of Software & Hardware Research in Engineering*, iJournals 2(6):93–98
- Kim B, Khanna R, Koyejo OO (2016) Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, ACM New York, NY, p 29
- Kim J, Feldt R, Yoo S (2019) Guiding deep learning system testing using surprise adequacy. In: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE, pp 1039–1049
- Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. [arXiv:1609.02907](https://arxiv.org/abs/1609.02907)
- Li C, Ma J, Guo X, Mei Q (2017) Deepcas: An end-to-end predictor of information cascades. In: Proceedings of the 26th international conference on World Wide Web, ACM New York, NY, pp 577–586
- Li Y, Dang X, Tian H, Sun T, Wang Z, Ma L, Klein J, Bissyandé TF (2022) Ai-driven mobile apps: an explorative study. [arXiv:2212.01635](https://arxiv.org/abs/2212.01635)
- Li Y, Dang X, Ma L, Klein J, Traon YL, Bissyandé TF (2023) Test input prioritization for 3d point clouds. *ACM Transactions on Software Engineering and Methodology*, ACM New York, NY
- Li Z, Ma X, Xu C, Cao C, Xu J, Lü J (2019) Boosting operational dnn testing efficiency through conditioning. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM New York, NY, pp 499–509
- Liu M, Gao H, Ji S (2020) Towards deeper graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, ACM New York, NY, pp 338–348
- Liu T, Lin Y, Wen X, Jorissen RN, Gilson MK (2007) Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic acids research*, Oxford University Press 35(suppl\_1):D198–D201
- Long Y, Wu M, Liu Y, Fang Y, Kwok CK, Chen J, Luo J, Li X (2022) Pre-training graph neural networks for link prediction in biomedical networks. *Bioinformatics*, Oxford University Press 38(8):2254–2262
- Ma L, Juefei-Xu F, Zhang F, Sun J, Xue M, Li B, Chen C, Su T, Li L, Liu Y, et al. (2018) Deepgauge: Multi-granularity testing criteria for deep learning systems. In: Proceedings of the 33rd ACM/IEEE Int Autom Softw Eng Conf, ACM New York, NY, pp 120–131
- Ma W, Papadakis M, Tsakmalis A, Cordy M, Traon YL (2021) Test selection for deep learning systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, ACM New York, NY, USA, 30(2):1–22
- Mesquita D, Souza A, Kaski S (2020) Rethinking pooling in graph neural networks. *Advances in Neural Information Processing Systems*, ACM New York, NY 33:2220–2231

- Morris C, Ritzert M, Fey M, Hamilton WL, Lenssen JE, Rattan G, Grohe M (2019) Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI conference on artificial intelligence*. ACM New York, NY 33:4602–4609
- Neumann M, Garnett R, Bauckhage C, Kersting K (2016) Propagation kernels: efficient graph kernels from propagated information. *Machine learning*, Springer 102:209–245
- Panichella A, Kifetew FM, Tonella P (2017) Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. *IEEE Trans Softw Eng*, IEEE 44(2):122–158
- Park N, Kan A, Dong XL, Zhao T, Faloutsos C (2019) Estimating node importance in knowledge graphs using graph neural networks. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, ACM New York, NY, pp 596–606
- Patel E, Kushwaha DS (2020) Clustering cloud workloads: K-means vs gaussian mixture model. *Procedia computer science*, Elsevier 171:158–167
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: Machine learning in python. *The Journal of machine Learning research*. JMLR org 12:2825–2830
- Pei K, Cao Y, Yang J, Jana S (2017) Deepxplore: Automated whitebox testing of deep learning systems. In: *proceedings of the 26th Symposium on Operating Systems Principles*, ACM New York, NY, pp 1–18
- Qiong Q, Dongxia W (2016) Evaluation method for node importance in complex networks based on eccentricity of node. In: *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, IEEE, pp 2499–2502
- Ranganathan H, Venkateswara H, Chakraborty S, Panchanathan S (2017) Deep active learning for image classification. In: *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, pp 3934–3938
- Réau M, Renaud N, Xue LC, Bonvin AM (2023) Deeprank-gnn: a graph neural network framework to learn patterns in protein–protein interfaces. *Bioinformatics*, Oxford University Press, 39(1):btac759
- Ren P, Xiao Y, Chang X, Huang PY, Li Z, Gupta BB, Chen X, Wang X (2021) A survey of deep active learning. *ACM computing surveys (CSUR)*, ACM New York, NY, 54(9):1–40
- Riesen K, Bunke H (2008) Iam graph database repository for graph based pattern recognition and machine learning. In: *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4–6, 2008*. *Proceedings*, Springer, pp 287–297
- Sassano M (2002) An empirical study of active learning with support vector machines forjapanese word segmentation. In: *Proceedings of the 40th annual meeting of the association for computational linguistics*, Association for Computational Linguistics, pp 505–512
- Sculley D (2010) Web-scale k-means clustering. In: *Proceedings of the 19th international conference on World wide web*, ACM New York, NY, pp 1177–1178
- Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. *AI magazine*, AAAI 29(3):93–93
- Shen W, Li Y, Chen L, Han Y, Zhou Y, Xu B (2020) Multiple-boundary clustering and prioritization to promote neural network retraining. In: *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, IEEE, pp 410–422
- Shervashidze N, Schweitzer P, Van Leeuwen EJ, Mehlhorn K, Borgwardt KM (2011) Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, JMLR, 12(9)
- Shi C, Xu M, Zhu Z, Zhang W, Zhang M, Tang J (2020) Graphphaf: a flow-based autoregressive model for molecular graph generation. [arXiv:2001.09382](https://arxiv.org/abs/2001.09382)
- Sun C, Shrivastava A, Vondrick C, Sukthankar R, Murphy K, Schmid C (2019) Relational action forecasting. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, pp 273–283
- Thekumparampil KK, Wang C, Oh S, Li LJ (2018) Attention-based graph neural network for semi-supervised learning. [arXiv:1803.03735](https://arxiv.org/abs/1803.03735)
- Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. [arXiv:1710.10903](https://arxiv.org/abs/1710.10903)
- Wang D, Shang Y (2014) A new active labeling method for deep learning. In: *2014 International joint conference on neural networks (IJCNN)*, IEEE, pp 112–119
- Wang Z, You H, Chen J, Zhang Y, Dong X, Zhang W (2021) Prioritizing test inputs for deep neural networks via mutation analysis. In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, IEEE, pp 397–409
- Weiss M, Tonella P (2022) 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*, ACM New York, NY, pp 139–150

- Wieder O, Kohlbacher S, Kuenemann M, Garon A, Ducrot P, Seidel T, Langer T (2020) A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, Elsevier 37:1–12
- Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, Sajed T, Johnson D, Li C, Sayeeda Z (2018) Drugbank 5.0: a major update to the drugbank database for, et al (2018) *Nucleic acids research*. Oxford University Press 46(D1):D1074–D1082
- Wu L, Sun P, Hong R, Fu Y, Wang X, Wang M (2018) Socialgcn: An efficient graph convolutional network based model for social recommendation. [arXiv:1811.02815](https://arxiv.org/abs/1811.02815)
- Wu S, Sun F, Zhang W, Xie X, Cui B (2022) Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, ACM New York, NY 55(5):1–37
- Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. *Trans Neural Netw Learn Syst*, IEEE 32(1):4–24
- Xiao S, Wang S, Dai Y, Guo W (2022) Graph neural networks in node classification: survey and evaluation. *Machine Vision and Applications*, Springer 33:1–19
- Xu K, Hu W, Leskovec J, Jegelka S (2018) How powerful are graph neural networks? [arXiv:1810.00826](https://arxiv.org/abs/1810.00826)
- Yang Y, Yu L, Wang X, Zhou Z, Chen Y, Kou T (2019) A novel method to evaluate node importance in complex networks. *Phys A: Stat Mech Appl*, Elsevier 526:121118
- Yang Z, Cohen W, Salakhudinov R (2016) Revisiting semi-supervised learning with graph embeddings. *International conference on machine learning*. PMLR, ACM New York, NY, pp 40–48
- Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J (2018) Graph convolutional neural networks for web-scale recommender systems. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, ACM New York, NY, pp 974–983
- Yu J, Yin H, Li J, Gao M, Huang Z, Cui L (2020) Enhance social recommendation with adversarial graph convolutional networks. *IEEE Trans Knowl Data Eng*, IEEE
- Yu Z, Kraft NA, Menzies T (2018) Finding better active learners for faster literature reviews. *Empir Softw Eng*, Springer 23:3161–3186
- Zhang XM, Liang L, Liu L, Tang MJ (2021) Graph neural networks and their current applications in bioinformatics. *Frontiers in genetics*, Frontiers Media SA 12:690049
- Zhao T, Zhang X, Wang S (2021) Graphsmote: Imbalanced node classification on graphs with graph neural networks. In: *Proceedings of the 14th ACM international conference on web search and data mining*, ACM New York, NY, pp 833–841
- Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2020) Graph neural networks: A review of methods and applications. *AI Open*, Elsevier 1:57–81
- Zolfagharian A, Abdellatif M, Briand LC, Bagherzadeh M, Ramesh S (2023) A search-based testing approach for deep reinforcement learning agents. *IEEE Transactions on Software Engineering*, IEEE

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Xueqi Dang** received her master's degree from King's College London. She is currently working toward the Ph.D. degree with the University of Luxembourg. Her work mainly involves machine learning testing, especially graph neural network testing and traditional machine learning testing.



**Yinghua Li** received his master's degree from Chongqing University. He was mainly engaged in the research of machine learning. He is currently working toward the Ph.D. degree with the University of Luxembourg. His work mainly involves AI for SE and machine learning testing.



**Wei Ma** received his PhD degree from the University of Luxembourg. He primarily engaged in research on Code Foundation Models and Software Engineering. He is currently working as a postdoctoral researcher at NTU, Singapore. His work mainly involves AI in Software Engineering, Software Engineering in AI, and Software Testing.



**Yuejun Guo** received her PhD degree from the Universitat de Girona. She worked as a Postdoctoral Researcher at the University of Luxembourg from 2020 to 2022, focusing on Deep Learning Testing. She is currently working as an R&T Associate at the Luxembourg Institute of Science and Technology. Her research explores AI/ML for cybersecurity, such as vulnerability detection, repair, and anti-fuzzing.



**Qiang Hu** received his PhD degree from the University of Luxembourg. He primarily engaged in research on Deep Learning Testing. He is currently working as a research associate at the University of Tokyo. His research interests span the areas of software engineering and deep learning, including deep learning testing, AIOps, and AI4SE.



**Mike Papadakis** received the PhD diploma degree in computer science from the Athens University of Economics and Business. He is an associate professor with the Interdisciplinary Center for Security, Reliability and Trust (SnT) with the University of Luxembourg. He is recognised for his work on software testing and in particular in the area of mutation testing. His research interests also include static analysis, prediction modelling and search-based software engineering.




**Maxime Cordy** works as a Research Scientist at SnT - the Interdisciplinary Research Centre on Security, Reliability and Trust (SnT) of the University of Luxembourg. His research activities currently focus on artificial intelligence (in particular, machine learning security, quality assurance, and applications), and he also has a strong background in software engineering. He is one of the three permanent scientists of SnT's SeRVal group (SEcurity, Reasoning, and VALidation), leading a team of 10-15 researchers working on AI.





**Yves Le Traon** is a professor with the University of Luxembourg where he leads the SnT ( Interdisciplinary Centre for Security, Reliability and Trust). His research interests within the group include (1) innovative testing and debugging techniques, (2) Android apps security and reliability using static code analysis, machine learning techniques and, (3) model-driven engineering with a focus on IoT and CPS. His reputation in the domain of software testing is acknowledged by the community. He has been General Chair of major conferences in the domain, such as the 2013 IEEE International Conference on Software Testing, Verification and Validation (ICST), and Program Chair of the 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS). He serves at the editorial boards of several, internationally-known journals (STVR, SoSym, IEEE Transactions on Reliability) and is author of more than 150 publications in international peer-reviewed conferences and journals.

## Authors and Affiliations

Xueqi Dang<sup>1</sup> · Yinghua Li<sup>1</sup>  · Wei Ma<sup>3</sup> · Yuejun Guo<sup>2</sup> · Qiang Hu<sup>4</sup> · Mike Papadakis<sup>1</sup> · Maxime Cordy<sup>1</sup> · Yves Le Traon<sup>1</sup>

✉ Yinghua Li  
yinghua.li@uni.lu

Xueqi Dang  
xueqi.dang@uni.lu

Wei Ma  
ma\_wei@ntu.edu.sg

Yuejun Guo  
yuejun.guo@list.lu

Qiang Hu  
qianghu0515@gmail.com

Mike Papadakis  
michail.papadakis@uni.lu

Maxime Cordy  
maxime.cordy@uni.lu

Yves Le Traon  
yves.letraon@uni.lu

<sup>1</sup> SnT Centre, University of Luxembourg, Luxembourg, Luxembourg

<sup>2</sup> LIST, Luxembourg Institute of Science and Technology, Luxembourg, Luxembourg

<sup>3</sup> Nanyang Technological University, Singapore, Singapore

<sup>4</sup> The University of Tokyo, Bunkyo, Japan