

Null Model-Based Data Augmentation for Graph Classification

Qi Xuan, *Senior Member, IEEE*, Zeyu Wang, Jinhuan Wang, Yalu Shan,
Xiaoke Xu, *Member, IEEE*, Guanrong Chen, *Fellow, IEEE*

Abstract—In network science, the null model is typically used to generate a series of graphs based on randomization as a term of comparison to verify whether a network in question displays some non-trivial features such as community structure. Since such non-trivial features play a significant role in graph classification, the null model could be useful for network data augmentation to enhance classification performance. In this paper, we propose a novel technique that combines the null model with data augmentation for graph classification. Moreover, we propose four standard null model-based augmentation methods and four approximate null model-based augmentation methods to verify and improve the performance of our graph classification technique. Our experiments demonstrate that the proposed augmentation technique has significantly achieved general improvement on the tested datasets. In addition, we find that the standard null model-based augmentation methods always outperform the approximate ones, depending on the design mechanisms of the null models. Our results indicate that the choice of non-trivial features is significant for increasing the performance of augmentation models for different network structures, which also provides a new perspective of data augmentation for studying various graph classification methods.

Index Terms—Null model, data augmentation, graph classification, graph data mining, structural feature.

1 INTRODUCTION

NULL models are pattern-generating models that deliberately exclude a mechanism being tested and primarily rely on its ability to explore non-trivial features of graphs, as a popular analytical tool applied to investigating the dynamics of complex networks. Null models have been applied to analyzing ecological and biogeographic data and quantifying complex network properties such as community structure [1], [2], assortativity [3], degree correlation [4], epidemic spreading rate [5], routing efficiency [6], pattern detection [7], microbial diversification [8], etc. In many applications, null models can reveal important network properties that could not be directly quantified by other models or methods. Inspired by this, we employ null models to improve the accuracy of graph classification.

Various, the graph classification augmentation methods can be roughly divided into two categories: feature augmentation methods and data augmentation methods. Feature augmentation methods focus on the embedding obtained from classical graph classification methods. Specifically, this kind of methods utilize feature selection and feature splicing. In [9], a semi-supervised feature selection approach is taken to search for optimal subgraph features with labeled and unlabeled graphs so as to improve graph classification performance. In [10], the concept of subgraph network

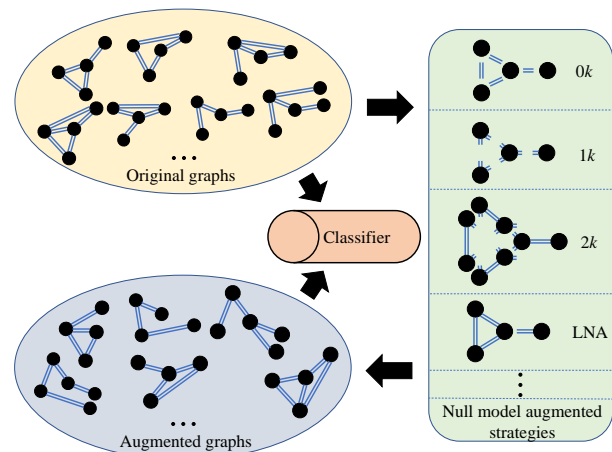


Fig. 1. The process of data augmentation based on null model.

(SGN) is introduced and used to expand the structural feature space of the underlying network, thereby enhancing network classification. In [11], a sampling subgraph network is constructed to address the problem of the SGN model that lacks diversity and has high time complexity. Whereas a feature fusion framework is also established to integrate the structural features of diverse sampling subgraphs to improve the performance of graph classification.

Data augmentation, on the other hand, is a technology that artificially expands the training datasets by allowing limited data to generate more equivalent data, so as to improve the performance of downstream tasks, such as node classification [12], community detection [13], and graph classification [14]. Researches in network science, especially in graph classification, focused on the graph structure and

- Q. Xuan, Z. Wang, J. Wang, Y. Shan are with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou, China (E-mail: xuanqi@zjut.edu.cn; Vencent_Wang@outlook.com; jhwang@zjut.edu.cn; yalushan@foxmail.com).
- X. Xu is with the College of Information and Communication Engineering, Dalian Minzu University, Dalian, China (E-mail: xuxiaoke@foxmail.com).
- G. Chen is with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong SAR, China (E-mail: eegchen@cityu.edu.hk).
- Corresponding author: Jinhuan Wang.

proposed some heuristic methods from the perspective of nodes or communities to modify the graph topology structure [15], [16]. These methods generate augmented data by introducing a tiny disturbance into the original data and then altering the parameters through retraining the model, which can also be thought of as a regularization method but lacks the instruction of data augmentation. Graph classification aims to identify the category labels of graphs in a dataset by using features that are extracted by handcraft, graph kernels [17], graph embedding [18], or graph neural networks [19]. Some non-trivial features of the graph are identified by the classifier. Practically, one should pay more attention to these non-trivial features when proposing data augmentation methods, which not only improve the classifier performance, but also provide some inspiration for the interpretability of the graph classification.

In this paper, we propose a new technique that combines the null model with data augmentation for graph classification. The idea is to find non-trivial features of the graph and develop a null model according to the non-trivial features to produce more virtual data for retraining the graph classifier, as illustrated in Fig. 1. Since the virtual data we generated guarantees that the non-trivial features will not change, their availability remains to be verified. Given this, we adopt the data filter proposed in [14] to filter out fine augmented examples from the generated data. We demonstrate that the method can significantly improve the performance of graph classification. Specifically, we have the following contributions:

- We propose a new technique for graph data augmentation. It is the first methodology to combine the null model with data augmentation and applied to graph classification tasks.
- We develop four standard null model-based data augmentation models and four approximate null model-based data augmentation methods for non-trivial features, which play a significant role in the graph classification.
- We apply our data augmentation methods based on several graph classification methods, and our experimental results on five real-world network datasets demonstrate the effectiveness of the proposed methods.

The rest of the paper is organized as follows. In Sec. 2, we present a brief description of the null model, data augmentation, and graph classification methods. In Sec. 3, we introduce four standard null model-based data augmentation methods and four approximate null model-based data augmentation methods. In Sec. 4, we describe our experimental settings and discuss the results in detail. Finally, in Sec. 5, we conclude the paper and outline some future work.

2 RELATED WORK

In this section, we give a brief review on related work of null models, data augmentation, and graph classification algorithms in graph data mining [20].

2.1 Null Model

A null model is a pattern-generating model based on random sampling from a known or imagined distribution [21].

Null model as an analysis tool was applied in ecological and biogeographic studies in the past few decades, such as the laws of species migration, island rules, and the spatial patterns of trees in temperate forests [22], [23], [24]. Different from other generating models, a null model constructs a model that deliberately excludes a mechanism being tested [25]. Many null models are used to simulate different processes and find the underlying rules, demonstrated that the null models can explain the internal mechanism of the tested model. In network analysis, a null model consists of a network that can be seen as one specific graph with some structural attributes but otherwise is treated as a random network instance. For undirected graphs, several null model generation methods have been proposed, such as 0-order, 1-order, 2-order, 2.5-order null model graphs [4], [26], [27], [28], [29], [30]. Such null models are helpful to explore the nature of modeling and structures of complex networks. However, it is noticed that there is no study to analyze the graph classification task by utilizing null models. In this paper, we adopt three null models with different orders and five novel (approximate) null models generated according to different non-trivial features to enhance the performance of graph classification.

2.2 Data Augmentation

Data augmentation is a common method for solving problems caused by insufficient dataset or model overfitting, and is widely applied to computer vision [31], [32] and natural language processing [33], [34]. Nevertheless, data augmentation in graph data is still in its infancy. In [14], three heuristic methods are developed to generate the virtual data and achieve average improvement in accuracy on graph classification tasks. For node classification tasks, in [12], it was discovered that neural edge predictors can effectively encode class-homophilic structure to promote intra-class edges and demote inter-class edges in given graph structure and they leveraged these results to improve the performance of GNN-based node classification via edge prediction. In [35], the labels of the training set are propagated through the graph structure expanding the training set. Although the above methods can improve the classification performance by expanding the dataset, they all lack interpretability. In this paper, we propose a new technique that combines the null model framework with data augmentation, which can provide an explanatory basis for data augmentation in graph classification.

2.3 Graph Classification

Graph data is ubiquitous in nature and human society, ranging from atomic structures to social networks. Graph classification is an important task in the graph data mining, where its objective is to predict the labels of networks correctly. This task usually is implemented by combining machine learning classifiers and graph representation learning such as graph kernel, graph embedding, and deep learning methods. Among them, the graph kernel is a graph representation method directly oriented to the graph structure, which contains structured information of higher-dimensional Hilbert space, such as WL kernel [36] and Deep

TABLE 1
Notations used in this paper.

Symbol	Definition
G, G_{aug}	Original/augmented graph
V, E	Set of nodes/edges in graph G
n, m	Number of nodes/edges
\mathcal{D}	Average degree of graph G
C	Average clustering coefficient of graph G
P_L	The proportion of leaf nodes of graph G
d_{max}	The maximum degree of graph G
P_D, J_D	The (joint) degree distribution of graph G
E_C	Average eigenvector centrality of graph G
B_C	Average betweenness centrality of graph G
C_C	Average closeness centrality of graph G
E_{leaf}	The set of edges with leaf nodes
E'	The edges set of augmented graph G'
e_{add}, e_{del}	Edge of addition/deletion
E_{add}, E_{del}	Edges set of addition/deletion
S	The set of node feature value
f_i	The feature($C/B_C/C_C/E_C$) value of node v_i
F	The feature($C/B_C/C_C/E_C$) value of graph G
α	The cost coefficient of augmentation
T	Approximate augmentation iterations
\mathcal{F}	The augmentation function

WL [37]. While graph embedding is a method that maps networks to low micro-dense vectors, such as graph2vec [38]. Deep learning methods have attracted much attention in recent years, including graph neural network (GNN) [39], graph convolutional network (GCN) [40], and Diffpool [41]. Although the above graph representation methods have relatively high expressiveness and learning ability, they don't have good interpretability. Moreover, they rely only on a single network structure, limiting their ability to exploit the latent structural features. In this paper, we combine the interpretability of null models with high expressiveness graph representation methods. Our experimental results demonstrate that the proposed null models indeed enhance the graph representation methods.

3 METHODOLOGY

In this section, we first formulate the problem of data augmentation on graph classification. We then summarize ten graph attributes, propose four standard null model-based augmentation ($0k, 1k, 2k, LNA$) and four approximate null model-based augmentation methods ADA-(C, BC, CC, EC), and demonstrate the construction of the algorithms.

3.1 Notation and Problem Statement

Let $D = \{(G_i, y_i) | i = 1, 2, \dots, N\}$ denote a graph dataset that has N undirected unweighted graphs, where G_i is a graph with label y_i . As usual a graph is denoted as $G = (V, E)$ where $V = \{v_j | j = 1, 2, \dots, n\}$ and $E = \{e_j | j = 1, 2, \dots, m\}$ are the node and edge sets, with n and m denoting the numbers of nodes and edges, respectively. Specifically, for a dataset D , it is split to training set D_{train} , validation set D_{val} and testing set D_{test} . Among them, the

training set D_{train} and validation set D_{val} are used to pre-train the classifier and yield the original graph classifier \mathcal{C}_{ori} .

Graph data augmentation aims to generate novel and realistically rational graphs by designing a transformation model $G_{aug} \sim \mathcal{F}(G_{aug}|G)$, where $\mathcal{F}(\cdot|G)$ is the augmentation distribution conditioned on the original graph, representing the prior information for data distribution. Here, we adopt null models for constructing augmented graphs. Through data augmentation for each graph G in the training set, we get the augmented set $D_{aug} = \{G_{aug}\}$. Then, we combine D_{train} and D_{aug} to get D'_{train}

$$D'_{train} = D_{train} + D_{aug}, \quad (1)$$

which will be used to retrain the classifier with optimization, and so as to obtain the augmented classifier \mathcal{C}_{aug} .

3.2 Graph Attributes

Graph attributes reflect the topology of a graph from different perspectives, which contribute to the graph classification. Here, we summarize ten classical graph attributes.

- **Number of nodes (n):** The number of nodes in the graph G .
- **Number of edges (m):** The number of edges in the graph G .
- **Average degree (\mathcal{D}):** The degree of a node in a graph is the number of connections it has to other nodes. The average degree of the graph G is given by

$$\mathcal{D} = \frac{2m}{n}. \quad (2)$$

- **Degree distribution (P_D):** The degree distribution is the probability distribution of all degrees over the whole network. The degree distribution of the graph G is calculated by

$$P_D(k) = \frac{n_k}{n}, \quad (3)$$

where n_k denotes the number of nodes with degree k in G .

- **The proportion of leaf nodes (P_L):** Leaf node is the node with degree one. The proportion of leaf nodes in the graph G is calculated by

$$P_L = \frac{n_L}{n}, \quad (4)$$

where n_L denotes the number of leaf nodes in G .

- **Joint degree distribution (J_D):** The joint degree distribution [42] refers to the number of degree (probability) of the nodes of each edge. Here, the joint degree distribution of the graph G is defined as

$$J_D(k_1, k_2) = \frac{\mu(k_1, k_2)m(k_1, k_2)}{2m}, \quad (5)$$

where $m(k_1, k_2)$ denotes the number of edges that the nodes with degree k_1 and the nodes with degree k_2 , and $\mu(k_1, k_2)$ is defined as

$$\mu(k_1, k_2) = \begin{cases} 2, & k_1 = k_2 \\ 1, & otherwise \end{cases} \quad (6)$$

- **Average clustering coefficient (C):** The clustering coefficient [43], [44] is a coefficient used to describe the degree of clustering between nodes in the graph. The average clustering coefficient of the graph G can be denoted as

$$C = \frac{1}{n} \sum_{i=1}^n \frac{2L_i}{k_i(k_i - 1)}, \quad (7)$$

where the k_i is the degree of node v_i and L_i is the number of edges among the k_i neighbors of node v_i .

- **Average betweenness centrality (B_C):** The betweenness centrality [45] is a measure of graph centrality based on the shortest paths. The average betweenness centrality of the graph G is computed by

$$B_C = \frac{1}{n} \sum_{i=1}^n \sum_{s \neq i \neq t} \frac{m_{st}^i}{g_{st}}, \quad (8)$$

where g_{st} is the number of shortest paths between node v_s and node v_t , and m_{st}^i is the number of shortest paths passing through node v_i between node v_s and node v_t .

- **Average closeness centrality (C_C):** The closeness centrality [46], [47] is the average length of the shortest paths between a node and other nodes in the graph. And the average closeness centrality of the graph G is computed by

$$C_C = \frac{1}{n} \sum_{i=1}^n \frac{n}{\sum_{j=1}^n d_{ij}}, \quad (9)$$

where d_{ij} is the length of shortest path between nodes v_i and v_j .

- **Average eigenvector centrality (E_C):** The eigenvector centrality [48], [49] is used to measure the importance of nodes on the network. The average eigenvector centrality of the graph G is defined as

$$E_C = \frac{1}{n} \sum_{i=1}^n x_i, \quad (10)$$

where x_i is the importance score of node v_i and given by:

$$x_i = \lambda \sum_{j=1}^n A_{ij} x_j, \quad (11)$$

where λ is an adjustable parameter, which should be less than the reciprocal of the maximum eigenvalue of the adjacency matrix A . Whereas A_{ij} represents the weight of the edge between node v_i and node v_j .

3.3 Graph Data Augmentation

Firstly, we use three classical null models to design the graph augmentation module, specifically the 0-order ($0k$), 1-order ($1k$), and 2-order ($2k$) null models. Then, we construct one standard null model-based heuristic graph augmentation strategy, Leaf Node Augmentation (LNA), and four approximate null model-based heuristic graph augmentation strategies, referred to as Approximate Data Augmentation (ADA), including Betweenness Centrality Augmentation (ADA-BC), Clustering Coefficient Augmentation (ADA-C), Eigenvector Centrality Augmentation (ADA-EC), and Closeness Centrality Augmentation (ADA-CC).

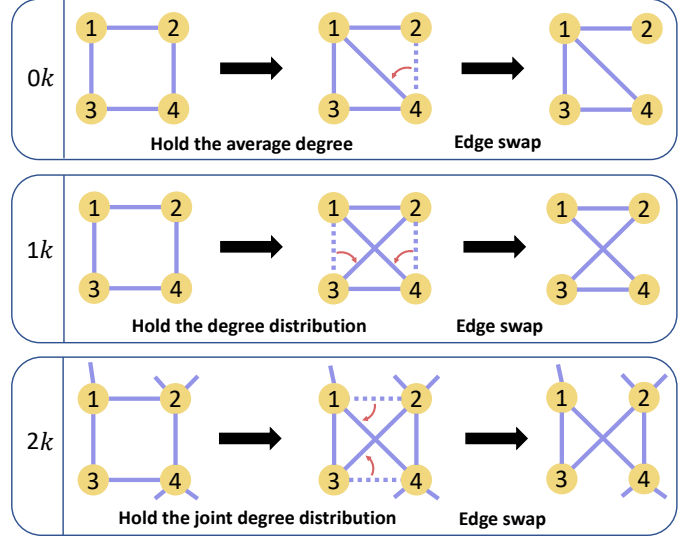


Fig. 2. The augmentation methods of $0k$, $1k$, and $2k$ null models. The legend that the red arrow points from the dotted line to the solid line represents a rewiring operation.

3.3.1 Standard Null Model-based Augmentation

Classical Null Model Augmentation. Typically, there are two methods to generate null models. One is based on the configuration model [50], and the other is based on rewiring edges. In this paper, the method of rewiring edges is used to construct $0k$, $1k$, and $2k$ null models. The null models with different orders hold different properties to remain consistent with the original graph. The corresponding augmentation strategies are briefly described in Fig. 2.

The generation of $0k$ null model is based on random rewiring, that is, randomly selecting an edge (v_2, v_4) to break and randomly selecting a pair of disconnected nodes (v_1, v_4) to connect. In general, the rewiring operation is performed for multiple times according to the experimental setting and the network scale in order to fully randomize the network. One can see that the $0k$ augmentation model holds the same average degree as the original graph.

As for the $1k$ null model, its rewiring constraint is stricter than that of the $0k$ null model. As shown by the $1k$ null model example in Fig. 2, one randomly break (v_1, v_3) , (v_2, v_4) and then connect (v_1, v_4) , (v_2, v_3) to keep the degree of each node unchanged before and after rewiring. That is, the $1k$ null model will select two edges in each rewiring operation while maintaining the consistency of node degree distribution with the original graph on the basis of the $0k$ null model.

The augmentation strategy of higher-order null model is extended from that of the $1k$ null model. Thus, the $2k$ null model is obtained through adding a new restricted condition on the basis of the $1k$ null model. As shown by the $2k$ augmentation example in Fig. 2, rewiring is operated only when the nodes v_2 and v_4 (or v_1 and v_3) have the same degree, i.e., the degree values of the endpoints of the edges remain the same after rewiring. The augmentation distribution based on the $2k$ null model is the joint degree distribution of the original graph.

Leaf Node Augmentation (LNA). A leaf node has degree

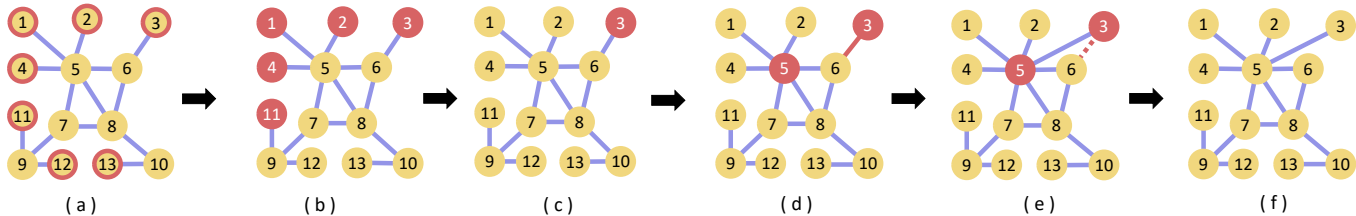


Fig. 3. An example of LNA with augmentation cost coefficient $\alpha = 0.2$. (a) Original graph; (b) The eligible leaf nodes marked by red in the graph; (c) The $\alpha * 5$ augmented nodes (v_3 here) randomly selected from leaf nodes; (d) The nodes with the highest degree (v_5 here) among the neighbors of chosen leaf nodes; (e) The rewiring process, i.e., a new edge is established between v_3 and v_5 , while that between v_3 and v_6 is disconnected; (f) The augmented graph G_{aug} .

1, which is very common and significant in real-world networks, e.g., aldehyde, amino, methyl, and other functional groups on the benzene rings which can determine the chemical properties of the compounds [51]. Also, in the taxonomies of genes [52], the leaf nodes have more important biological meanings than the internal nodes in some situation, for example filial samples are more important than parental samples in the study of genetic diseases with inter-generational genetic attributes. The LNA is an augmentation strategy by fixing the proportion of leaf nodes. Given a graph $G = (V, E)$, denote the set of edges with leaf node by $E_{leaf} = \{(u_i, w_i) \in E | i = 1, \dots, p\}$, where u_i is a leaf node in the graph G , w_i is the neighbor of the leaf node u_i , and p is the number of leaf nodes. LNA obtains augmentation graphs by rewiring the leaf nodes. The construction method is shown in Algorithm 1. To avoid generating more leaf nodes after rewiring, the edges in E_{leaf} should be filtered by a filter \mathcal{L} to get $E'_{leaf} = \{(u_i, w_i) | i = 1, \dots, q\} (q < p)$, where the constraint on w_i is that its degree remains more than 1 after removing the leaf node u_i , with

$$E'_{leaf} = \mathcal{L}(E_{leaf}) = \bigcup_i^p \{\mathbb{I}(u_i, w_i)\}, \quad (12)$$

where the function $\mathbb{I}(u_i, w_i)$ is defined by,

$$\mathbb{I}(u_i, w_i) = \begin{cases} (u_i, w_i), & d(w_i) > 1 \\ \emptyset, & otherwise \end{cases} \quad (13)$$

In order to ensure that no new leaf nodes are generated during the augmentation, each time an edge (u_i, w_i) is chosen from E'_{leaf} , where the degree of w_i must be subtracted by 1. Then, randomly select $E_{del} \subset E'_{leaf}$ as the set of deleted edges, where $|E_{del}| = \alpha * |E'_{leaf}|$ and α is the cost coefficient of augmentation. When deleting the existing edges, the topology of the graph will be damaged to some extent. In order to make new leaf nodes carry as much information of their neighbors as possible, each u_j will be reconnected to the node \bar{w}_j with the highest degree among the neighbors of w_j . Then, the set of adding edges is denoted as $E_{add} = \{(u_j, \bar{w}_j) | j = 1, \dots, \alpha * q\}$. Finally, based on the LNA, the original graph is modified to become a new graph $G_{aug} = (V, E')$, where

$$E' = E \cup E_{add} \setminus E_{del}. \quad (14)$$

Fig. 3 shows an example of LNA with augmentation cost coefficient $\alpha = 0.2$. As is shown in Fig. 3 (a), there is a graph with seven leaf nodes $\{v_1, v_2, v_3, v_4, v_{11}, v_{12}, v_{13}\}$.

Algorithm 1 Leaf Nodes Augmentation

Input: Original graph G

Parameters: Augmentation cost coefficient α

Output: Augmented graph G_{aug}

- 1: Get edges with leaf node E_{leaf} ;
 - 2: Get E'_{leaf} via Eq (12) and Eq (13);
 - 3: $E_{del} \leftarrow \text{RandomSample}(E'_{leaf}, \alpha)$;
 - 4: **for** $(u_j, w_j) \in E'_{del}$ **do**
 - 5: $\bar{w}_j = \arg \max_v ((G.\text{neighbors}(w_j)).\text{degree}());$
 - 6: $E_{add}.\text{append}((u_j, \bar{w}_j))$;
 - 7: **end for**
 - 8: Get G_{aug} via Eq.(14);
 - 9: Return G_{aug} ;
-

Among them, leaf nodes v_{11}, v_{12} have a common neighbor node v_9 . If edges (v_9, v_{11}) and (v_9, v_{12}) are deleted at the same time, node v_9 will become a new leaf node, which will not meet Eq. (13), so one of v_{11}, v_{12} will be randomly selected to put into the eligible nodes set. Also, because v_{10} will become a leaf node after deleting the edge (v_{10}, v_{13}) , node v_{13} does not meet the augmentation constraint. Thus, the five eligible leaf nodes $\{v_1, v_2, v_3, v_4, v_{11}\}$ are marked. Then, randomly select $\alpha * 5$ eligible leaf nodes as augmented nodes, remove their original edges, and connect them to their 2-hop neighbor nodes with the highest degrees. Finally, the leaf node-based augmented graph is obtained.

3.3.2 Approximate Null Model-based Augmentation

Approximate Data Augmentation (ADA). Existing null models are based on simple features such as degree distribution, while centrality metrics based on network path structure, such as clustering coefficient, betweenness centrality, closeness centrality, or eigenvector centrality, not only consider the network topology but also summarize the participation or contribution of nodes to the network. It is not difficult to understand that, as the features become more complex, there will be more constraints in the generation of networks based on null models. Also, the more similar the generated network is to the original network, the more difficult the rewiring would become, i.e., fewer edges are appropriate to be selected to rewire under the stricter constraints. In order to address this, we propose a feature approximation method to reduce the impact of certain features in the augmentation process. Here, the concept of feature approximation is applied in the specific

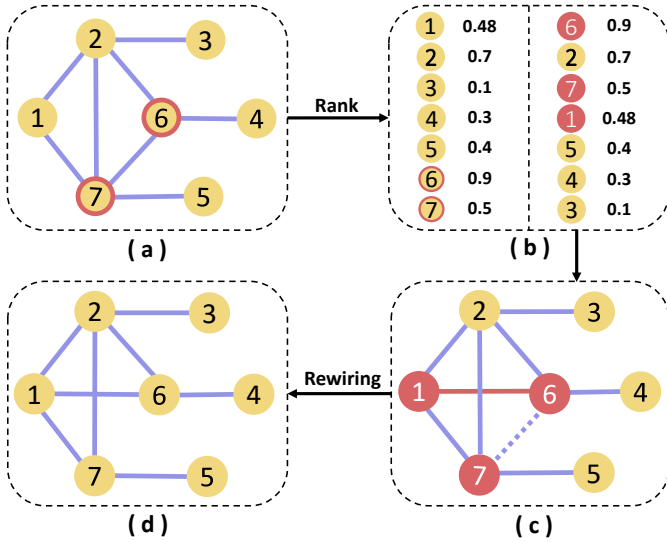


Fig. 4. The process of approximate augmentation. (a) Original graph G , (b) features ranking and augmented node acquisition, (c) rewiring operation (the feature values are fictitious for illustration only), (d) augmented graph G_{aug} .

Algorithm 2 Approximate Augmentation

Input: Original graph G

Parameters: Augmentation cost coefficient α , Iterations T

Output: Augmented graph G_{aug}

- 1: Initialize iteration = 0;
 - 2: Get the feature value of original graph F , the number of edges m , the number of nodes n ;
 - 3: **for** iteration $< T$ **do**
 - 4: Initialize swap = 0;
 - 5: **while** swap $< \alpha * m$ **do**
 - 6: Randomly sample an edge $e_1 = (v_1, v_2)$ without leaf node;
 - 7: Get the set of feature values $\mathcal{S} = \{f_v | v = v_1, \dots, v_n\}$;
 - 8: $u \leftarrow \arg \max_v (f_{v_1}, f_{v_2})$;
 - 9: $w \leftarrow \arg \min_v (f_{v_1}, f_{v_2})$;
 - 10: Get the nodes \bar{w} that $f_{\bar{w}}$ is closest to f_w ;
 - 11: $e_{del} \leftarrow e_1$;
 - 12: $e_{add} \leftarrow e_2 = (u, \bar{w})$;
 - 13: Rewiring to get the G' ;
 - 14: **if** G' is connected **then**
 - 15: swap = swap + 1;
 - 16: **else**
 - 17: Cancel the rewiring;
 - 18: **end if**
 - 19: **end while**
 - 20: **end for**
 - 21: Get G_{aug} via Eq (15);
 - 22: Return G_{aug} ;
-

augmentation algorithms based on average clustering coefficient, average betweenness centrality, average closeness centrality, or average eigenvector centrality. In the following description, the graph feature values for a graph G are denoted by the symbol F , and their feature values for a

node v are replaced with the symbol f .

As shown in Fig. 4, for a given graph $G = (V, E)$, we can first randomly select an edge without leaf node and set it as $e_1 = (v_6, v_7)$. Secondly, we get the list of feature values of each node in the graph $\mathcal{S} = \{f_v | v = v_1, \dots, v_n\}$. Then, as shown in tables of Fig. 4 (b), we sort the list of feature values and get the node whose feature value is closest to that of v_7 . Here, the node with the closest feature value is v_1 . Meanwhile, we also do the operation of approximate augmentation by executing the equation in line 10 of Algorithm 2 to get the node v_1 that has no edge with v_6 but with its feature value closest to the node v_7 . Next, we delete the edge (v_6, v_7) and connect node v_6 with v_1 to get the new edge (v_6, v_1) . After that, if the graph G is connected, we perform rewiring as an effective augmentation operation; otherwise, cancel this rewiring operation and enter the next loop (line 14-19 in Algorithm 2). At this point, the augmentation operation for one edge has been completed, and it will happen $\alpha * m$ times in the process of graph augmentation to generate G' , where α denotes the rewiring cost coefficient and m denotes the number of edges in G . At the same time, we set the iteration parameter T for this augmentation, and choose the best-augmented graph to return:

$$G_{aug} = \arg \min_{G'} |F' - F|. \quad (15)$$

Thus, as long as T is large enough, this model can generate an augmented graph G_{aug} with high similarity.

4 EXPERIMENTS

In this section, we conduct some experiments to evaluate the effectiveness of our graph data augmentation strategies for graph classification on a variety of real-world network datasets. We first introduce the datasets, then outline the graph classification methods adopted as baselines, and finally describe the experimental setup. After that, we show the experimental results with some discussion.

4.1 Datasets

In order to access our augmentation methods, we adopt five commonly used benchmark datasets in experiments, which are BZR, COX2, MUTAG, OHSU, and ENZYMES. Among them, BZR, COX2, MUTAG, and ENZYMES are biological and chemical datasets, and OHSU is a brain dataset. The statistics of these datasets are summarized in Table 2.

- **BZR** [53] is a dataset of 405 ligands for the benzodiazepine receptor, where the nodes and edges represent atoms and chemical bonds, respectively. They are classified to active and inactive compounds according to a predetermined threshold.
- **COX2** [53] is a dataset of 467 cyclooxygenase-2 with in vitro activity against human recombinant enzymes. Their graph topologies represent the position information of atoms and chemical bonds. They can also be classified into active and inactive compounds using a predetermined threshold.
- **MUTAG** [54] is a collection of nitroaromatic compounds, in which nodes and edges correspond to

atoms and chemical bonds, respectively. Their classification is based on their mutagenicity to *Salmonella typhimurium*.

- **OHSU** [55] composes the functional brain networks constructed from the whole brain fMRI map, where each node corresponds to a region of interest (ROI), and the edge represents the correlation between the two ROIs. The dataset can be divided to two classes based on the hyperactivity-impulsive attributes.
- **ENZYMES** [56] represents the protein macromolecules, where nodes indicate the secondary structure elements. If these nodes are neighbors along specific sequence, or they are neighbors in space within the protein structure, they will be connected. Each node is connected to its three nearest spatial neighbors. Further, according to different catalytic reactions, they can be divided into 6 classes.

TABLE 2

Statistics of five datasets used in experiments. N_G , N_C , $\#Nodes$, and $\#Edges$ denote the number of graphs, the number of graph classes, the average number of nodes, and the average number of edges, respectively.

Datasets	N_G	N_C	$\#Nodes$	$\#Edges$
BZR	405	2	43.75	38.44
COX2	467	2	41.22	43.45
MUTAG	188	2	17.93	43.79
OHSU	79	2	82.01	439.66
ENZYMES	600	6	32.63	62.14

4.2 Graph Classification Methods

Here, we utilize some graph classification methods to learn the representations of the original and augmented graphs and then predict the class of the given graph. Under the present framework, we adopt five different methods namely SF, NetLSD, gl2vec, Graph2vec, and Diffpool, where SF and Graph2vec are graph embedding methods, NetLSD and gl2vec are graph kernel models, and Diffpool is an end-to-end graph neural network method.

- **SF** [57] is an embedding method, which relies on spectral features of the graph. It performs graph classification based on the spectral decomposition of the graph Laplacian.
- **Graph2vec** [38] is the first unsupervised embedding approach for an entire network, which can learn data-driven distributed representations of arbitrary sized graphs.
- **NetLSD** [58] is a graph kernel model, which compares graphs and achieves graph classification by extracting a compact signature that inherits the formal properties of the Laplacian spectrum.
- **gl2vec** [59] generates feature representation by static or temporal network graphlet distribution and a null model to compare with random graphs.
- **Diffpool** [41] introduces a way to aggregate nodes to learn a graph representation that contains hierarchical information. It can be combined with GNN architectures in an end-to-end fashion.

4.3 Experimental Setup

In this study, the embedding dimension of all graph kernels and embedding are set as 128. For SF, the random seed value is set to 42. For Graph2vec and gl2vec, the number of cores is set to 4. Given that these methods are based on the rooted subgraphs, some parameters are related to the setting of the WL kernel, where the number of Weisfeiler-Lehman iterations is 2. Also, the parameters are set to commonly-used values: the learning rate is set to 0.025, the epochs is set to 500. For NetLSD, the scheme calculates the heat kernel trace of the normalized Laplacian matrix over a vector of time scales. If the matrix is large, it switches to an approximation of the eigenvalues. Specifically, the number of eigenvalue approximations is set to 200, whereas the minimum and maximum time scale interval are set to -2.0 and 2.0 , respectively. For Diffpool, the parameter epochs are set to 3000 and the other parameters are set to default values [41]. In addition, the first four unsupervised representation methods, SF, Graph2vec, NetLSD, and gl2vec, are paired with four machine learning algorithms to implement the graph classification task, where the four machine learning algorithms are Support Vector Machine classifier based on radial basis kernel (SVM), Logistic regression classifier (Logistic), K-Nearest Neighbors classifier (KNN) and Random Forest classifier (RF). Therefore, there are totally $4 \times 4 + 1 = 17$ kinds of graph classification schemes in validation experiments.

Considering that lower augmentation cost coefficient will make the features value closer to the original value, we set the modified edge connection ratio (augmentation cost coefficient) of each augmentation model to $\alpha=0.2$ and set the number of the iteration of approximate augmentation as $T=5$. In our experiments, each dataset are divided into the training set, validation set, and test set with the ratio of 7:1:2, where the training set is augmented by the augmentation strategies developed in this paper. Then, we use the data filtering method in [14] to filter the augmented graph set. Finally, we feed the augmented training set into the different graph classification classifiers for training.

In the experiments, the following metrics are adopted to evaluate the graph classification performance of different augmentation strategies:

- **Accuracy.** Accuracy measures the classification performance with the proportion of correctly classified graphs over all graphs in the dataset.
- **Success Rate.** The augmentation success rate refers to the ratio of the cases in which the augmented classification accuracy is higher than the original classification accuracy to the total.
- **Relative Gain Ratio.** The relative gain Ratio is defined as:

$$R_{gain} = \frac{Acc_{aug} - Acc_{ori}}{Acc_{ori}} \times 100\%, \quad (16)$$

where Acc_{aug} and Acc_{ori} denote the augmented and original classification accuracy, respectively.

4.4 Results and Discussion

The experimental results conducted on the five datasets with the configuration in Section 4 are summarized in

TABLE 3
Graph classification results of original and standard null model-based augmentation models. The best results are marked in bold.

Datasets	Aug Model	SF				Graph2vec				NetLSD				gl2vec				Diffpool	R_{Gain}
		SVM	Logistic	KNN	RF	SVM	Logistic	KNN	RF	SVM	Logistic	KNN	RF	SVM	Logistic	KNN	RF		
BZR	original	0.796	0.734	0.805	0.838	0.799	0.799	0.837	0.839	0.807	0.757	0.803	0.819	0.799	0.807	0.843	0.834	0.827	/
	0k	0.799	0.811	0.809	0.841	0.804	0.837	0.852	0.846	0.813	0.81	0.81	0.833	0.801	0.821	0.851	0.839	0.846	2.11%
	1k	0.806	0.802	0.811	0.841	0.802	0.833	0.856	0.839	0.816	0.81	0.815	0.833	0.804	0.842	0.857	0.834	0.853	2.33%
	2k	0.807	0.809	0.807	0.843	0.804	0.836	0.859	0.847	0.821	0.809	0.815	0.834	0.804	0.841	0.853	0.835	0.851	2.49%
	LNA	0.806	0.808	0.816	0.842	0.801	0.833	0.849	0.847	0.815	0.812	0.814	0.832	0.803	0.831	0.852	0.838	0.849	2.29%
COX2	original	0.777	0.735	0.779	0.77	0.778	0.746	0.786	0.783	0.77	0.666	0.774	0.752	0.777	0.728	0.782	0.788	0.804	/
	0k	0.78	0.782	0.783	0.783	0.779	0.789	0.788	0.791	0.776	0.785	0.78	0.768	0.778	0.789	0.787	0.79	0.823	2.91%
	1k	0.782	0.786	0.796	0.79	0.783	0.809	0.791	0.803	0.776	0.786	0.793	0.773	0.779	0.799	0.797	0.794	0.83	3.80%
	2k	0.781	0.781	0.794	0.786	0.781	0.799	0.8	0.8	0.784	0.786	0.795	0.778	0.782	0.797	0.799	0.796	0.822	3.75%
	LNA	0.778	0.785	0.785	0.782	0.78	0.795	0.787	0.787	0.782	0.784	0.785	0.777	0.779	0.794	0.784	0.791	0.828	3.16%
MUTAG	original	0.822	0.824	0.829	0.854	0.744	0.777	0.772	0.818	0.823	0.786	0.827	0.837	0.741	0.795	0.781	0.797	0.759	/
	0k	0.835	0.864	0.837	0.871	0.746	0.831	0.781	0.848	0.843	0.857	0.855	0.864	0.747	0.843	0.809	0.836	0.835	3.82%
	1k	0.834	0.853	0.845	0.862	0.742	0.837	0.817	0.849	0.846	0.864	0.859	0.871	0.749	0.839	0.84	0.817	0.853	4.39%
	2k	0.836	0.858	0.844	0.863	0.746	0.824	0.816	0.835	0.845	0.863	0.855	0.865	0.743	0.82	0.828	0.828	0.853	3.98%
	LNA	0.839	0.858	0.856	0.865	0.747	0.825	0.805	0.863	0.836	0.849	0.854	0.86	0.747	0.84	0.833	0.838	0.851	4.30%
OHSU	original	0.61	0.565	0.61	0.639	0.557	0.58	0.577	0.582	0.547	0.504	0.55	0.558	0.557	0.535	0.542	0.516	0.476	/
	0k	0.678	0.643	0.646	0.735	0.557	0.635	0.603	0.628	0.603	0.552	0.615	0.654	0.557	0.587	0.577	0.62	0.585	10.34%
	1k	0.683	0.645	0.638	0.714	0.559	0.686	0.627	0.678	0.661	0.569	0.604	0.688	0.614	0.608	0.668	0.63	0.588	14.46%
	2k	0.686	0.628	0.64	0.732	0.562	0.635	0.662	0.676	0.663	0.566	0.636	0.66	0.614	0.587	0.639	0.68	0.606	14.63%
	LNA	0.678	0.658	0.64	0.714	0.557	0.648	0.625	0.683	0.656	0.584	0.622	0.68	0.564	0.63	0.614	0.655	0.626	14.30%
ENZYMES	original	0.312	0.241	0.274	0.385	0.365	0.244	0.28	0.322	0.34	0.202	0.311	0.343	0.356	0.245	0.273	0.304	0.353	/
	0k	0.324	0.266	0.292	0.407	0.309	0.275	0.302	0.353	0.357	0.254	0.337	0.371	0.291	0.284	0.278	0.348	0.357	6.10%
	1k	0.343	0.273	0.335	0.429	0.399	0.28	0.334	0.369	0.363	0.26	0.355	0.377	0.374	0.285	0.668	0.63	0.588	14.30%
	2k	0.336	0.269	0.319	0.425	0.398	0.288	0.346	0.38	0.362	0.267	0.348	0.381	0.383	0.29	0.295	0.368	0.437	15.02%
	LNA	0.336	0.263	0.333	0.439	0.377	0.261	0.321	0.359	0.39	0.262	0.369	0.382	0.358	0.27	0.298	0.347	0.422	12.75%

TABLE 4
Graph classification results of original and approximate augmentation models. The best results are marked in bold.

Datasets	Aug Model	SF				Graph2vec				NetLSD				gl2vec				Diffpool	R_{Gain}
		SVM	Logistic	KNN	RF	SVM	Logistic	KNN	RF	SVM	Logistic	KNN	RF	SVM	Logistic	KNN	RF		
BZR	original	0.796	0.734	0.805	0.838	0.799	0.799	0.837	0.839	0.807	0.757	0.803	0.819	0.799	0.807	0.843	0.834	0.827	/
	ADA-C	0.802	0.808	0.808	0.84	0.801	0.826	0.846	0.842	0.81	0.808	0.809	0.825	0.799	0.827	0.846	0.844	0.846	1.85%
	ADA-BC	0.803	0.81	0.812	0.843	0.8	0.832	0.847	0.841	0.809	0.804	0.81	0.83	0.803	0.839	0.848	0.837	0.848	2.06%
	ADA-CC	0.804	0.809	0.811	0.844	0.803	0.825	0.847	0.844	0.812	0.806	0.814	0.833	0.801	0.831	0.854	0.84	0.848	2.13%
	ADA-EC	0.804	0.81	0.802	0.843	0.804	0.832	0.854	0.839	0.812	0.816	0.809	0.829	0.798	0.841	0.849	0.841	0.852	2.20%
COX2	original	0.777	0.735	0.779	0.77	0.778	0.746	0.786	0.783	0.77	0.666	0.774	0.752	0.777	0.728	0.782	0.788	0.804	/
	ADA-C	0.78	0.785	0.782	0.78	0.778	0.78	0.783	0.792	0.773	0.779	0.777	0.764	0.778	0.784	0.79	0.793	0.822	2.66%
	ADA-BC	0.783	0.783	0.788	0.783	0.779	0.797	0.788	0.797	0.78	0.785	0.795	0.776	0.779	0.787	0.786	0.793	0.827	3.33%
	ADA-CC	0.78	0.779	0.782	0.779	0.78	0.797	0.787	0.792	0.778	0.784	0.778	0.77	0.779	0.795	0.786	0.792	0.824	3.00%
	ADA-EC	0.78	0.789	0.781	0.78	0.778	0.785	0.792	0.797	0.774	0.779	0.776	0.771	0.778	0.782	0.791	0.79	0.821	2.85%
MUTAG	original	0.822	0.824	0.829	0.854	0.744	0.777	0.772	0.818	0.823	0.786	0.827	0.837	0.741	0.795	0.781	0.797	0.759	/
	ADA-C	0.837	0.845	0.847	0.874	0.745	0.807	0.792	0.823	0.845	0.856	0.856	0.861	0.751	0.815	0.784	0.801	0.837	2.89%
	ADA-BC	0.841	0.863	0.844	0.876	0.746	0.807	0.78	0.835	0.854	0.868	0.853	0.866	0.748	0.843	0.809	0.825	0.84	3.78%
	ADA-CC	0.837	0.866	0.851	0.87	0.745	0.818	0.803	0.841	0.842	0.867	0.854	0.869	0.748	0.839	0.826	0.819	0.844	4.09%
	ADA-EC	0.833	0.839	0.839	0.868	0.746	0.823	0.8	0.835	0.849	0.859	0.857	0.873	0.749	0.825	0.804	0.809	0.836	3.40%
OHSU	original	0.61	0.565	0.61	0.639	0.557	0.58	0.577	0.582	0.547	0.504	0.55	0.558	0.557	0.535	0.542	0.516	0.476	/
	ADA-C	0.671	0.628	0.646	0.696	0.557	0.651	0.604	0.62	0.591	0.562	0.583	0.639	0.557	0.641	0.626	0.62	0.584	10.45%
	ADA-BC	0.684	0.627	0.638	0.699	0.557	0.643	0.635	0.646	0.623	0.567	0.605	0.628	0.557	0.618	0.588	0.633	0.573	10.88%
	ADA-CC	0.686	0.617	0.646	0.709	0.557	0.629	0.647	0.617	0.612	0.562	0.617	0.635	0.557	0.584	0.627	0.622	0.588	10.79%
	ADA-EC	0.656	0.64	0.643	0.702	0.557	0.613	0.6	0.628	0.582	0.57	0.637	0.653	0.557	0.623	0.605	0.661	0.589	10.94%
ENZYMES	original	0.312	0.241	0.274	0.385	0.365	0.244	0.28	0.322	0.34	0.202	0.311	0.343	0.356	0.245	0.273	0.304	0.353	/
	ADA-C	0.329	0.255	0.297	0.41	0.319	0.28	0.301	0.344	0.357	0.252	0.344	0.366	0.299	0.283	0.277	0.343	0.419	7.19%
	ADA-BC	0.328	0.262	0.297	0.407	0.316	0.278	0.301	0.339	0.36	0.243	0.341	0.371	0.291	0.283	0.281	0.33	0.405	6.39%
	ADA-CC	0.329	0.27	0.301	0.412	0.323	0.282	0.298	0.343	0.355	0.238	0.342	0.371	0.294	0.293	0.27	0.339	0.416	7.19%
	ADA-EC	0.326	0.264	0.301	0.405	0.314	0.269	0.31	0.345	0.358	0.238	0.343	0.373	0.295	0.284	0.274	0.349	0.417	6.91%

Table 3 and Table 4. The data in the two tables consist of the classification accuracy of the original model and the augmentation model, and the average of the relative gain ratio of each augmentation method in different classification mechanisms. The data are analyzed from different perspectives such as the performance of augmentation models, the augmentation success rate, and the analysis of time complexity.

4.4.1 Performance of Augmentation Models

Table 3 presents the graph classification results of the original and standard null model-based augmentation models. It is easy to see that, $0k$, $1k$, $2k$, and LNA augmentation models significantly improve the performance of graph classification compared with the original models based on the five graph representation methods, where the $2k$ augmentation model based on $gl2vec$ -RF for the OHSU dataset even leads to an improvement of 16.4%. It is worth noting that

both $1k$ and $2k$ null model-based augmentation strategies generally have the best augmentation effectiveness on all datasets, which achieve the average gain ratio of 14.46% and 14.63% respectively for OHSU, and 14.30% and 15.02% respectively for ENZYMES. This is reasonable because these two augmentation models maintain the basic features such as (joint) degree distribution, which can better describe the network correlation.

Also, due to the approximate nature of the ADA methods, extra bias could be introduced, and thus these methods are typically somewhat weaker in enhancing graph classification models, as shown in Table 4. As can be seen, the best performance is achieved with average gain ratio of 10.94% (obtained by ADA-EC for OHSU), which is about 4% less than the $2k$ null model-based augmentation model. For the multi-class dataset ENZYMES, the classification results obtained by the standard null model-based augmentation

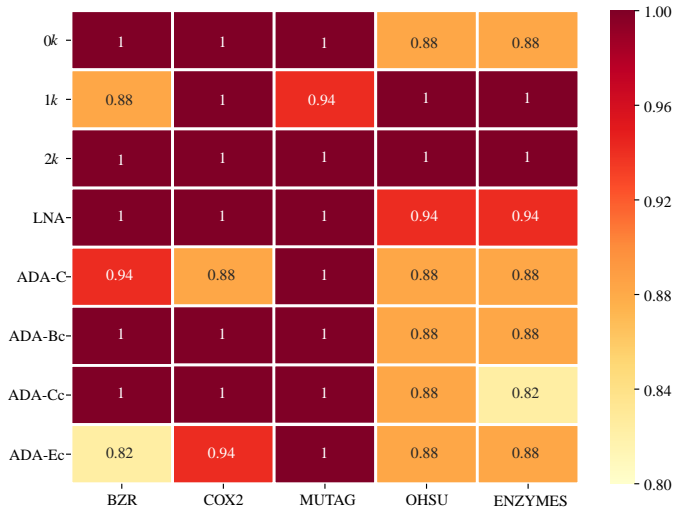


Fig. 5. Statistics of augmentation success rate of various datasets.

TABLE 5
Average augmentation time for the graph on five datasets (unit: second).

Dataset	BZR	COX2	MUTAG	OHSU	ENZYMES
0k	1.087	0.674	0.012	0.777	0.240
1k	0.006	0.004	0.003	0.066	0.002
2k	0.006	0.008	0.002	0.066	0.002
LNA	0.009	0.011	0.004	0.088	0.003
ADA-C	0.072	0.113	0.036	3.673	0.138
ADA-BC	0.969	3.388	0.216	46.831	1.478
ADA-CC	0.457	1.771	0.143	18.363	0.751
ADA-EC	0.763	3.158	0.392	3.801	1.102

models are also significantly better compared with ADA models. Besides, it is found that better classification enhancement performance generally occurs with the *Diffpool* method or unsupervised representation methods with *RF* classifier, which indicates that the effectiveness of these data augmentation methods could be further improved by designing appropriate graph representation and classification methods.

Moreover, we also investigate the augmentation success rate based on different augmentation models using the five datasets. According to Table 3 and Table 4, one can see that, the augmentation models outperform the original model in 168 out of 170 cases. Also, the augmentation success rate of the $2k$ is higher than other augmentation models, even reaching 100% on the five datasets, as shown in Fig. 5. The minimum augmentation success rates of ADA-CC and ADA-EC also reach 82%. Overall, the average augmented success rate of all the augmentation models is higher than 91% on these datasets. This phenomenon supports the conclusion that our augmentation methods can indeed improve the performance of graph classification.

4.4.2 Design of Null Model-Based Augmentation Models

Since our augmentation models show general improvement on the five datasets, we further dissect and illustrate their different performances on some particular graphs. We extract graph features manually and utilize *Gini importance*

computed by the *Random Forest* [60], [61], [62] classifier to evaluate the feature importance. The feature with the high *Gini importance* plays a more significant role than others in graph classification. Fig. 6 shows a compound chart consisting of the *Gini importance* (bar) of features and the average gain ratio (line) of augmentation models, where (a)-(e) are based on the null model-based augmentation, and (f)-(j) are based on the approximate null model-based augmentation. Interestingly, we find the consistency between the augmentation effects of the augmentation models and their corresponding features. This phenomenon is more prominent in the standard null model-based augmentation. Also, we find that the $2k$ augmentation model on MUTAG dataset has high *Gini importance* but relatively low average gain ratio. A possible reason is that the augmentation based on $2k$ null model will break the structure of the benzene ring, which however is important for the classification of MUTAG. The non-positive trends in Fig 6 (f)-(j) are also expected. Although the approximate augmentation methods are designed based on the principle of ensuring the consistency of the features as much as possible, its randomness will inevitably bring bias. Therefore, it is worth exploring the possibility to design particular augmentation strategies for different kinds of network structures.

As an example, we visualize the different structures generated by the eight null model augmentation models on the seventh graph from the MUTAG dataset, as shown in Fig 7. MUTAG is a dataset of nitroaromatic compounds. After augmentation, one can see that the augmented graphs generated by different models have quite different structures. Compared with null-model augmented graphs, the approximate augmented graphs have less structural similarity to the original network. In particular, by adopting $2k$, ADA-C, ADA-EC, we find that not only the nitroaromatic structure is destroyed, but also the reconstructed six-membered ring may not be benzene rings composed of carbon. This also partly explains the phenomenon that in many cases the approximate augmentation methods are less effective than the standard null model-based augmentation methods.

Indeed, the main purpose of using the null model is to maintain the non-trivial features of a graph and gradually approximate the original graph. The results in Table 3 and Table 4 suggest that the augmentation methods are effective on both two-class and multi-class datasets, the reason being that the key feature of two-class datasets may be single, making two-class datasets easy to classify. However, the classification standard for the multi-class datasets could be different intervals of a key feature. This also gives us some inspiration to build diverse null models for different tasks so as to preserve more significant information.

4.4.3 Analysis of Time Complexity

Now, the computational complexity of the null model-based augmentation is analyzed. Set n , m , α , and T as the number of nodes, the number of edges, the cost coefficient of augmentation, and the approximate augmentation iteration times, respectively, in the original graph. It is easy to verify that the time complexities of $0k$, $1k$, $2k$, LNA, and ADA-C, BC, CC, EC) are $\mathcal{O}(n^2)$, $\mathcal{O}(\alpha * m)$, $\mathcal{O}(\alpha * n)$, $\mathcal{O}(m)$, and $\mathcal{O}(T * \alpha * m)$.

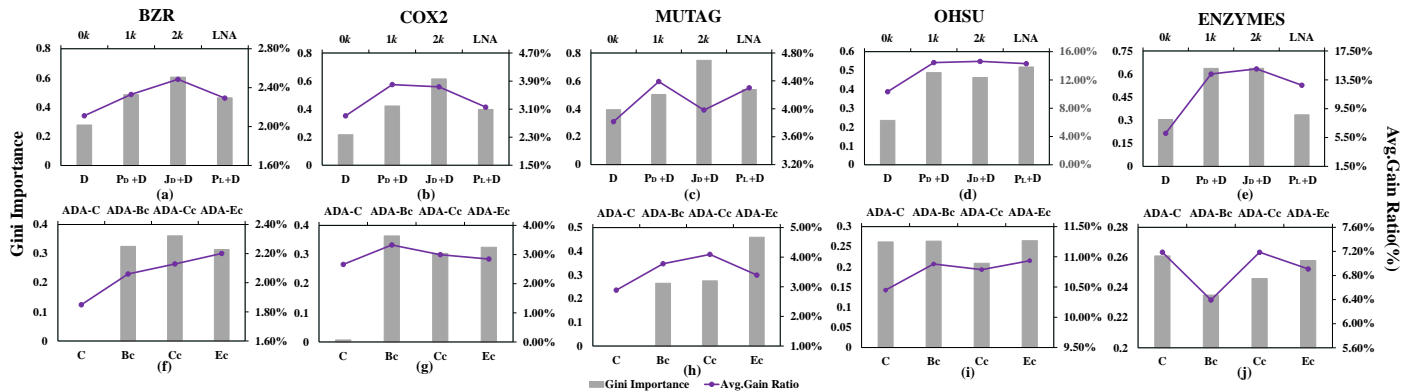


Fig. 6. The compound chart of the *Gini Importance* of features and *Avg. Gain Ratio*, where the features are the average degree of graph (D), the degree distribution of graph (P_D), the joint degree distribution of graph (J_D), the proportion of leaf nodes of graph (P_L), the clustering coefficient of graph (C), the betweenness centrality of graph (B_C), the closeness centrality of graph (C_C), and the eigenvector centrality of graph (E_C).

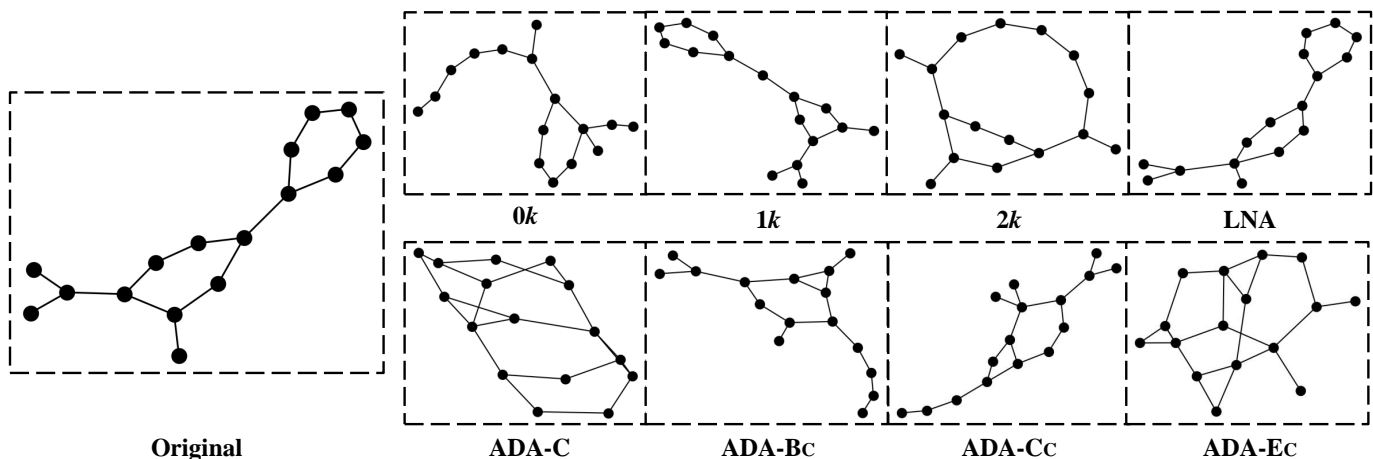


Fig. 7. The seventh graph in MUTAG and its augmentation graphs. Obviously, compared with other augmentation graphs, the 2k, ADA-C, ADA-Bc, ADA-Cc, and ADA-Ec augmented graphs severely damaged the benzene ring of the original graph.

Furthermore, we report the average augmentation running time for each graph based on different augmentation strategies on different datasets. As shown in Table 5, compared to the training time of the classifier, the 1k, 2k, and LNA augmentation methods all take less than 0.1 seconds but achieve relatively high improvements (see the classification results in Table 3), while the time consumptions of ADA-C, ADA-Bc, ADA-Cc, and ADA-Ec are comparatively high. The most critical operation is that the corresponding feature value must be recalculated in the augmentation of each edge. While for networks with high dimensionality and lots of edges, the time consumption on feature value calculation could also be large.

5 CONCLUSION

In this paper, we combine the null model with data augmentation to propose several data augmentation methods, which effectively improve the accuracy of graph classification. We conduct experiments to verify the effectiveness of our methods and analyze the experimental results to demonstrate the new findings. We compare five benchmark networks and our results show that the application

of data augmentation based on the null model can indeed significantly improve the accuracy of graph classification. We conclude that the null model can be applied to complex networks analysis, and it has great potential in the field of graph mining algorithms design. Furthermore, based on the experiments results, we find that the null models have the ability to maintain features consistently with better performance than other methods. These findings also indicate that network features are very important in graph tasks and can provide inspiration for graph data mining research.

In the future, we will study more important features of graph data in graph classification and explore the augmentation methods with such important features to achieve more efficient augmentation. Moreover, we will combine more excellent graph data mining methods with null models in other application scenarios.

REFERENCES

- [1] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, no. 3, p. 036104, 2006.

- [2] R. Cazabet, P. Borgnat, and P. Jensen, "Enhancing space-aware community detection using degree constrained spatial null model," in *International Workshop on Complex Networks*. Springer, 2017, pp. 47–55.
- [3] R. Pastor-Satorras, A. Vázquez, and A. Vespignani, "Dynamical and correlation properties of the internet," *Physical Review Letters*, vol. 87, no. 25, p. 258701, 2001.
- [4] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat, "Systematic topology analysis and generation using degree correlations," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 135–146, 2006.
- [5] E. Estrada, S. Meloni, M. Sheerin, and Y. Moreno, "Epidemic spreading in random rectangular networks," *Physical Review E*, vol. 94, no. 5, p. 052316, 2016.
- [6] X. Nian and H. Fu, "Efficient routing on two layer degree-coupled networks," *Physica A: Statistical Mechanics and its Applications*, vol. 410, pp. 421–427, 2014.
- [7] W. Ulrich and N. J. Gotelli, "Pattern detection in null model analysis," *Oikos*, vol. 122, no. 1, pp. 2–18, 2013.
- [8] X. Zhai, W. Zhou, G. Fei, W. Liu, Z. Xu, C. Jiao, C. Lu, and G. Hu, "Null model and community structure in multiplex networks," *Scientific Reports*, vol. 8, no. 1, pp. 1–13, 2018.
- [9] X. Kong and P. S. Yu, "Semi-supervised feature selection for graph classification," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 793–802. [Online]. Available: <https://doi.org/10.1145/1835804.1835905>
- [10] Q. Xuan, J. Wang, M. Zhao, J. Yuan, C. Fu, Z. Ruan, and G. Chen, "Subgraph networks with application to structural feature space expansion," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 6, pp. 2776–2789, 2021.
- [11] J. Wang, P. Chen, B. Ma, J. Zhou, Z. Ruan, G. Chen, and Q. Xuan, "Sampling subgraph network with application to graph classification," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 3478–3490, 2021.
- [12] T. Zhao, Y. Liu, L. Neves, O. J. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *Association for the Advancement of Artificial Intelligence*, 2021.
- [13] J. Zhou, Z. Chen, M. Du, L. Chen, S. Yu, G. Chen, and Q. Xuan, "RobustECD: Enhancement of network structure for robust community detection," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.
- [14] J. Zhou, J. Shen, S. Yu, G. Chen, and Q. Xuan, "M-evolve: Structural-mapping-based data augmentation for graph classification," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 190–200, 2020.
- [15] J. Zhou, J. Shen, and Q. Xuan, "Data augmentation for graph classification," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2341–2344.
- [16] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, and T. Goldstein, "{FLAG}: Adversarial data augmentation for graph neural networks," 2021. [Online]. Available: <https://openreview.net/forum?id=mj7WsaHYxj>
- [17] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and M. Vazirgiannis, "Grakel: A graph kernel library in python." *J. Mach. Learn. Res.*, vol. 21, pp. 54–1, 2020.
- [18] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.
- [19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.
- [20] Q. Xuan, Z. Ruan, and Y. Min, Eds., *Graph Data Mining: Algorithm, Security and Application*, ser. Big Data Management. Springer Singapore, 2021.
- [21] N. J. Gotelli, G. R. Graves, N. J. Gotelli, and G. R. Graves, *Null models in Ecology*, ser. Null models in Ecology, 1996.
- [22] M. Biddick and K. C. Burns, "A simple null model predicts the island rule," *Ecology Letters*, vol. 24, no. 8, pp. 1646–1654, Aug. 2021.
- [23] M. Carrer, D. Castagneri, I. Popa, M. Pividori, and E. Lingua, "Tree spatial patterns and stand attributes in temperate forests: The importance of plot size, sampling design, and null model," *Forest Ecology and Management*, vol. 407, pp. 125–134, 2018.
- [24] M. W. Reimann, M. Gevaert, Y. Shi, H. Lu, H. Markram, and E. Muller, "A null model of the mouse whole-neocortex micro-connectome," *Nature Communications*, vol. 10, no. 1, pp. 1–16, 2019.
- [25] N. J. Gotelli, "Research frontiers in null model analysis," *Global Ecology and Biogeography*, vol. 10, no. 4, pp. 337–343, 2001.
- [26] M. Gjoka, M. Kurant, and A. Markopoulou, "2.5 k-graphs: from sampling to generation," in *2013 Proceedings IEEE International Conference on Computer Communications*, 2013, pp. 1968–1976.
- [27] P. Mahadevan, C. Hubble, D. Krioukov, B. Huffaker, and A. Vahdat, "Orbis: rescaling degree correlations to generate annotated internet topologies," in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2007, pp. 325–336.
- [28] M. E. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64, no. 2, p. 026118, 2001.
- [29] F. Chung and L. Lu, "Connected components in random graphs with given expected degree sequences," *Annals of Combinatorics*, vol. 6, no. 2, pp. 125–145, 2002.
- [30] —, "The average distances in random graphs with given expected degrees," *Proceedings of the National Academy of Sciences*, vol. 99, no. 25, pp. 15 879–15 882, 2002.
- [31] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 113–123.
- [32] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.
- [33] M. Fadaee, A. Bisazza, and C. Monz, "Data augmentation for low-resource neural machine translation," R. Barzilay and M. Kan, Eds., 2017, pp. 567–573.
- [34] G. G. Ahin and M. Steedman, "Data augmentation via dependency tree morphing for low-resource languages," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.
- [35] H. Dong, Z. Ding, X. He, F. Feng, and S. Bi, "Data augmentation view on graph convolutional network and the proposal of monte carlo graph learning," *CoRR*, vol. abs/2006.13090, 2020.
- [36] N. Shervashidze, P. Schweitzer, E. Jan, V. Leeuwen, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *The Journal of Machine Learning Research*, vol. 12, no. 3, pp. 2539–2561, 2011.
- [37] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *the 21th ACM SIGKDD International Conference*, 2015.
- [38] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," in *Proceedings of the 13th International Workshop on Mining and Learning with Graphs*, 2017.
- [39] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.
- [40] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.
- [41] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, p. 4805–4815.
- [42] I. Stanton and A. Pinar, "Constructing and sampling graphs with a prescribed joint degree distribution," *ACM J. Exp. Algorithmics*, vol. 17, sep 2012. [Online]. Available: <https://doi.org/10.1145/2133803.2330086>
- [43] S. P. Borgatti and M. G. Everett, "Network analysis of 2-mode data," *Social Networks*, vol. 19, no. 3, pp. 243–269, 1997.
- [44] P. G. Lind, M. C. Gonzalez, and H. J. Herrmann, "Cycles and clustering in bipartite networks," *Physical Review E*, vol. 72, no. 5, p. 056127, 2005.
- [45] S. Wasserman, K. Faust *et al.*, "Social network analysis: Methods and applications," 1994.
- [46] A. Bavelas, "Communication patterns in task-oriented groups," *The Journal of the Acoustical Society of America*, vol. 22, no. 6, pp. 725–730, 1950.
- [47] M. A. Beauchamp, "An improved index of centrality," *Behavioral Science*, vol. 10, no. 2, pp. 161–163, 1965.
- [48] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *Journal of Mathematical Sociology*, vol. 2, no. 1, pp. 113–120, 1972.

- [49] —, “Some unique properties of eigenvector centrality,” *Social Networks*, vol. 29, no. 4, pp. 555–564, 2007.
- [50] Z. Ren, M. S. Mariani, Y. Zhang, and M. Medo, “A time-respecting null model to explore the structure of growing networks,” *ArXiv*, vol. abs/1703.07656, 2017. [Online]. Available: <https://doi.org/10.48550/arXiv.1703.07656>
- [51] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, “Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity,” *Journal of Medicinal Chemistry*, vol. 34, no. 2, pp. 786–797, 1991.
- [52] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, “Hierarchical multi-label prediction of gene function,” *Bioinformatics*, vol. 22, no. 7, p. 830–836, apr 2006. [Online]. Available: <https://doi.org/10.1093/bioinformatics/btk048>
- [53] Jeffrey, J., Sutherland, Lee, A., O’Brien, Donald, F., and Weaver, “Spline-fitting with a genetic algorithm: A method for developing classification structure—activity relationships.” *ChemInform*, vol. 35, no. 5, 2004.
- [54] N. Kriege and P. Mutzel, “Subgraph matching kernels for attributed graphs,” in *Proceedings of the 29th International Conference on Machine Learning*, 2012, p. 291–298.
- [55] N. Qiang, Q. Dong, F. Ge, H. Liang, and T. Liu, “Deep variational autoencoder for mapping functional brain networks,” *IEEE Transactions on Cognitive and Developmental Systems*, vol. 12, no. 4, pp. 841–852, 2020.
- [56] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg, “Brenda, the enzyme database: updates and major new developments,” *Nucleic Acids Research*, vol. 32, no. suppl_1, pp. D431–D433, 2004.
- [57] N. de Lara and E. Pineau, “A simple baseline algorithm for graph classification,” *arXiv preprint arXiv:1810.09155*, 2018.
- [58] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller, “Netlsd: hearing the shape of a graph,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2347–2356.
- [59] K. Tu, J. Li, D. Towsley, D. Braines, and L. D. Turner, “gl2vec: Learning feature representation using graphlets for directed networks,” in *Proceedings of the 2019 IEEE/ACM international Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 216–221.
- [60] L. Rokach and O. Maimon, “Top-down induction of decision trees classifiers - a survey,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 4, pp. 476–487, 2005.
- [61] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [62] B. H. Menze, B. M. Kelm, R. Masuch, U. Himmelreich, P. Bachert, W. Petrich, and F. A. Hamprecht, “A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data,” *BMC Bioinformatics*, vol. 10, no. 1, pp. 1–16, 2009.