Null Model-Based Data Augmentation for Graph Classification

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

Abstract—Graph classification is an important task widely applied to biochemistry, social networks, and other fields. Since it is a data-dependent problem, insufficient training data will deteriorate the performance of graph classification models. To address this issue, various data augmentation methods have been proposed. However, most existing methods tend to destroy the topological features, leading to a negative impact on information propagation and semantics. While the null model generates new data with the same topological features as the original graph and helps capture the latent information based on topology. Hence, combining the null model with data augmentation for graph classification may be useful in helping models learn graph representations. This paper introduces a novel null model-based augmentation technique for graph classification. Specifically, four standard and four approximate null model-based augmentation methods are developed to verify the effectiveness of the technique. Experimental results on benchmark datasets demonstrate significant performance improvements with the proposed technique. Depending on the design mechanisms of the null models, standard augmentation methods outperform the approximate ones. These findings emphasize the critical role of non-trivial features in enhancing augmentation models for different network structures, providing a new perspective on data augmentation for studying graph classification methods.

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

I. INTRODUCTION

Graph classification is a fundamental task in network science that involves assigning labels to graphs based on features extracted by handcraft, graph kernels [1], graph embedding [2], or graph neural networks [3]. With the growing prevalence of network data in diverse fields such as bioinformatics [4], social science [5], and chemoinformatics [6], graph classification has gained significant importance in data mining. The main objective of graph classification is to extract meaningful information from graphs and apply it to real-world applications [7]. For example, in molecular property prediction tasks, molecular structures are represented as graphs and used for downstream predictions [8]; in the social domain, researchers construct networks based on discussions to predict topic quality or information veracity [9]. However, graph classification has challenges as it requires high-quality labeled training data, which may be scarce and difficult to obtain in real-world scenarios [10]. For instance, in chemistry, the number of known compounds is much smaller than the unknown ones, and labeling compounds is a costly and challenging process. To address these issues, researchers have explored various data augmentation techniques to enhance graph classification models.

Graph classification augmentation methods can be roughly divided into two categories: feature augmentation methods and topological augmentation methods [11]. Feature augmentation methods focus on the embedding obtained from classical graph classification methods. Specifically, these kind of methods utilize feature selection and feature splicing. For instance, in [12], a semi-supervised feature selection approach is taken to search for optimal subgraph features with labeled and unlabeled graphs to improve graph classification performance. In [13], the concept of subgraph network (SGN) is introduced and used to expand the structural feature space of the underlying network, thereby enhancing network classification. In [14], a sampling subgraph network is constructed to address the problem of the SGN model that lacks diversity and has high time complexity. Meanwhile, a feature fusion framework is established to integrate the structural features of diverse sampling subgraphs to improve the performance of graph classification. These methods directly modify the features or representation, however, are abstract and lack interpretability.

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 [15]. Research in network science, especially in graph classification, focused on the graph structure and proposed some heuristic methods from the perspective of nodes or communities to modify the graph topology.
structure [16], [17], [18]. These methods can generate augmented data by introducing a small perturbation to the original data and then adjusting the model parameters through retraining. However, as is shown in Fig. 1(b), augmentation techniques such as DropNode [16] and DropEdge [17] have been observed to remove nodes or edges from the graph, potentially resulting in the loss of essential information and disrupting network connectivity. Consequently, this can negatively impact the propagation of information and alter the semantics of the graph, e.g., the structural properties and the graph representation might be significantly altered with important nodes or edges removed. Also, deleting critical nodes may cause the disconnection of important subgraphs or communities, which could diminish the ability of graph neural networks to capture meaningful patterns and features in the data. Moreover, these approaches lack explicit guidance from data augmentation and can result in the destruction of important topological features. Although it can still be viewed as a type of regularization method.

Actually, topological features and information of a network have long been focal points in graph research [19], [20]. In network science, topology is one of the core characteristics of the network, which can reflect the connections and relationships between nodes in the network. In graph classification tasks, important features can often be extracted from the network topology, and these features often have a great effect on distinguishing different graphs [21]. There is a lot of research that intends to capture the topological feature invariance to achieve effective classification, such as [22] utilizing global topological features (closeness centrality, etc.) to graph classification, [23] applying the concept of persistent homology to extract topology information to help graph classification. Practically, one should pay more attention to these non-trivial features when studying data augmentation methods, which not only improve the classifier performance but also provide some inspiration for the interpretability of the graph classification. Null models refer to randomized networks that have some of the same properties as a real network and have the ability to explore important topology features of the network [24], [25]. As an analysis tool for studying the features and dynamic characteristics of complex networks, its application fields involve ecology [26], social networks [27], biology [28], medical field [29], etc. For instance, [30] constructs null models (degree distribution, connectivity, etc.), compares them with real networks, and explores the influence of rich-club characteristics and the effect of rich nodes in complex networks, which has a crucial meaning in information propagation and marketing; in the medical field, null models can generate brain networks with some features to discriminate the relationship between different features [29]. In many applications, null models generate new data with the same features as the original graph and can reveal important network properties that could not be directly quantified by other models. Inspired by this, this paper combines null models with data augmentation to optimize the graph classification model.

In this paper, a new technique is proposed that combines the null model with data augmentation for graph classification. As illustrated in Fig. 1, the idea is to utilize the null model based on non-trivial features to produce more virtual data for retraining the graph classifier. As Fig. 1(b) null model augmentation strategies part describes, the virtual data guarantees that the non-trivial features will not change, while their availability remains to be verified. Given this, as is shown in Fig. 1(a), we adopt the data filtration mechanism proposed in [31] to filter out fine augmented examples from the generated data. We conducted lots of experiments to demonstrate that the method can significantly improve the performance of graph classification. Specifically, the main contributions are as follows:

- A new technique for graph data augmentation is proposed, which is the first methodology to combine the null model with data augmentation, and is applied to the graph classification task.
- 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 graph classification. Our augmentation models can preserve network features and help the model learn network properties to comprehensively capture graph information.
- We apply our data augmentation methods based on several graph classification methods, and our experimental results on seven real-world network datasets demonstrate the effectiveness of the proposed methods. We find that there is a correlation between the effectiveness of augmentation models and the importance of corresponding features in classification. Moreover, the 2k augmentation method enhances by 6.5% relative to the original classifiers. The performance of 2k is generally better than the six baselines, exhibiting a relative improvement of 12.6% specifically on the OHSU dataset.

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

II. RELATED WORK

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

A. Null Model

A null model is a pattern-generating model based on random sampling from a known or imagined distribution with some same features as real networks [32]. Different from conventional real network analysis methods, the method based on the null model constructs the null model, calculates the properties of the null model and the real network, and combines statistical analysis methods to explore the property description of the network. Therefore, the null model of networks, which can be used to quantify the specific nature of a network, is a powerful tool for analyzing the structural characteristics of complex systems [33] and has been applied in ecological and biological studies [34], [35]. For undirected graphs, several null model generation methods have been proposed, such as 0-order, 1-order, 2-order, and 2.5-order null model graphs [25], [36], [37]. For instance, [28] utilizes the 1-order null model to explore the topological properties and finds that links between highly connected proteins are systematically suppressed, while those between highly-connected and low-connected pairs of proteins are favored; [38] compares the AS-level network with the null model-based on randomized edges, utilizes the network correlation profile quality method explore the degree-related characteristic. 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 with data augmentation. In this paper, we adopt four standard null models with different orders and four approximate null models generated according to different non-trivial features to enhance the performance of graph classification.

B. 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 [39] and natural language processing [40]. Nevertheless, data augmentation in graph data is still in its infancy. [15] introduced the GAUG augmentation method for semi-supervised node classification tasks, utilizing the concept that edge predictors based on neural networks can proficiently encode homologous structures, enhancing relationships between edges within clusters and across clusters. In [31], three heuristic methods are developed to generate the virtual data and achieve average improvement in accuracy on graph classification tasks. [41] proposed MCGL which leverages local homogeneity assumptions and pseudo-label expansion through Monte Carlo sampling to improve semi-supervised node classification tasks on graphs, exhibiting strengths in handling non-linear boundaries and community characteristics. [42] introduced Graph Transplant, enabling the mixing of graphs by replacing sub-graphs while preserving the local structure, and leveraging node saliency for adaptively assigned labels to improve classification performance. In [43], the mixup method paid attention to the graph semantic space, enhancing GNNs by generating new representation samples and reducing over-fitting. Although the above methods can improve classification performance by expanding the dataset, they all lack interpretability. In this paper, we propose a new technique that combines the null model with data augmentation, which can provide an explanatory basis for data augmentation in graph classification.

C. Graph Classification

Graph data is ubiquitous in nature and human society, ranging from atomic structures to social networks, with downstream tasks that node classification [44], link prediction [45], community detection [46], [47], and graph classification [48]. Among them, Graph classification is an important task in graph data mining, where its objective is to correctly predict network labels. 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 [49] and Deep WL [50]. While graph embedding maps networks to low micro-dense vectors, such as graph2vec [51]. Deep learning methods have demonstrated their powerful performance, and attracted much attention in recent years, including GCN [44], GIN [48], and Diffpool [52], etc. Also, there are some frontier works such as KerGNN [53] and GHNN [54] that combine the graph kernel and GNNs. Although the above graph representation methods have relatively high expressiveness and learning ability, due to their heavy reliance on data, sufficient training data can significantly impair their performance and result in overfitting. In this paper, we combine the interpretability of null models with data augmentation to release these phenomena and improve augmented data quality.

III. 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 augmentations (0k, 1k, 2k, LNA) and four approximate null model-based augmentation methods ADA-(C, BC, CC, EC), and demonstrate the construction of the algorithms. Moreover, the notations used throughout the paper are listed in Table I.

A. Notation and Problem Statement

Let $D = \{(G_i, y_i) | i = 1, 2, \ldots, N\}$ 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 $V = \{v_j | j = 1, 2, \ldots, n\}$ and $E = \{e_j | j = 1, 2, \ldots, m\}$ are the
The number of nodes in the graph is denoted as \( n \) and the number of edges in the graph is denoted as \( m \). The degree of a node in a graph is the number of edges that the node has. The average degree of the graph \( G \) can be denoted as \( \langle k \rangle = \frac{2m}{n} \).

- **Degree distribution** \( (P_D) \): The degree distribution is the probability distribution of all degrees over the whole network. The degree distribution of the graph \( G \) can be denoted as
  \[
  P_D(k) = \frac{n_k}{n},
  \]
  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 \( G \) can be denoted as
  \[
  P_{L} = \frac{n_{L}}{n},
  \]
  where \( n_{L} \) denotes the number of leaf nodes in \( G \).

- **Joint degree distribution** \( (J_D) \): The joint degree distribution [55] 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
  \[
  J_D(k_1,k_2) = \frac{\mu(k_1,k_2)m(k_1,k_2)}{2m},
  \]
  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
  \[
  \mu(k_1,k_2) = \begin{cases} 
  2, & k_1 = k_2 \\
  1, & \text{otherwise}
  \end{cases}
  \]

- **Average clustering coefficient** \( (C) \): The clustering coefficient [56] 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
  \[
  C = \frac{1}{n} \sum_{i=1}^{n} \frac{2L_i}{k_i(k_i - 1)},
  \]
  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 [57] is a measure of graph centrality based on the shortest paths. The average betweenness centrality of the graph \( G \) can be denoted as
  \[
  B_C = \frac{1}{n} \sum_{i=1}^{n} \sum_{s 
eq i \neq t} \frac{m_{st}}{g_{st}},
  \]
  where \( g_{st} \) is the number of shortest paths between node \( v_s \) and node \( v_t \) and \( m_{st} \) 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 [58] is the average length of the shortest paths between a node and other nodes in the graph. The average closeness centrality of the graph \( G \) can be denoted as
  \[
  C_C = \frac{1}{n} \sum_{i=1}^{n} \frac{n}{\sum_{j=1}^{n} d_{ij}},
  \]
  where \( d_{ij} \) is the length of shortest path between nodes \( v_i \) and \( v_j \).
• **Average eigenvector centrality** \( (E_C) \): The eigenvector centrality [59] is used to measure the importance of nodes on the network. The average eigenvector centrality of the graph \( G \) can be denoted as

\[
E_C = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{10}
\]

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

\[
x_i = \lambda \sum_{j=1}^{n} A_{ij} x_j, \tag{11}
\]

where \( \lambda \) 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 \( v_j \).

### C. Graph Data Augmentation

Firstly, we use three classical null models to design the graph augmentation module, specifically the 0-order (0\( k \)), 1-order (1\( k \)), and 2-order (2\( k \)) 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).

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 [60], and the other is based on rewiring edges. In this paper, the method of rewiring edges is used to construct 0\( k \), 1\( k \), and 2\( k \) 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 0\( k \) 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 multiple times according to the experimental setting and the network scale in order to fully randomize the network. One can see that the 0\( k \) augmentation model holds the same average degree as the original graph.

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

The augmentation strategy of the higher-order null model is extended from that of the 1\( k \) null model. Thus, the 2\( k \) null model is obtained by adding a new restricted condition on the basis of the 1\( k \) null model. As shown by the 2\( k \) 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 2\( k \) null model is the joint degree distribution of the original graph.

**Leaf Node Augmentation (LNA)**: A leaf node has degree 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 [61]. Also, in the taxonomies of genes [62], the leaf nodes have more important biological meanings than the internal nodes in some situations, for example, filial samples are more important than parental samples in the study of genetic diseases with intergenerational 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 \( E_{leaf} = \{(u_i, w_i) \in E | i = 1, \ldots, p\} \), 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 \( E'_{leaf} = \{(u_i, w_i) | i = 1, \ldots, q \}(q < p) \), where the constraint on \( w_i \) is that its degree remains more than 1 after removing the leaf node \( u_i \), with

\[
E'_{leaf} = \mathcal{L}(E_{leaf}) = \bigcup_i \{u_i, w_i\}, \tag{12}
\]

where the function \( u_i, w_i \) is defined by,

\[
\begin{cases} 
(u_i, w_i), & d(w_i) > 1 \\
\emptyset, & \text{otherwise}
\end{cases} \tag{13}
\]
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 \(E'_{\text{leaf}}\), where the degree of \(w_i\) must be subtracted by 1. Then, randomly select \(E_{\text{del}} \subseteq E'_{\text{leaf}}\) as the set of deleted edges, where \(|E_{\text{del}}| = \alpha \ast |E'_{\text{leaf}}|\) and \(\alpha\) 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 about their neighbors as possible, each \(u_j\) will be reconnected to the node \(\overline{w}_j\) with the highest degree among the neighbors of \(w_j\). Then, the set of adding edges is denoted as \(E_{\text{add}} = \{(u_j, \overline{w}_j) | j = 1, \ldots, \alpha \ast q\}\). Finally, based on the LNA, the original graph is modified to become a new graph \(G_{\text{aug}} = (V, E')\), where
\[
E' = E \cup E_{\text{add}} \setminus E_{\text{del}}.
\]

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 \(\{v_1, v_2, v_3, v_4, v_{11}, v_{12}, v_{13}\}\). 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 (13), so one of \(v_{11}, v_{12}\) will be randomly selected to put into the eligible nodes set. Also, because \(v_{10}\) is a leaf node after deleting the edge \((v_{10}, v_1)\), node \(v_1\) does not meet the augmentation constraint. Thus, the five eligible leaf nodes \(\{v_1, v_{12}, v_3, v_4, v_{11}\}\) are marked. Then, randomly select \(\alpha \ast 5\) 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.

Algorithm 1: Leaf Nodes Augmentation.

**Input:** Original graph \(G\)

**Parameters:** Augmentation cost coefficient \(\alpha\)

**Output:** Augmented graph \(G_{\text{aug}}\)

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

As shown in Fig. 4, for a given graph \(G = (V, E)\), one can first randomly select an edge without leaf nodes and set it as \(e_1 = (v_6, v_7)\). Secondly, one can get the list of feature values of each node in the graph \(\delta = \{f_v | v = v_1, \ldots, v_n\}\). Then, as shown in tables of Fig. 4(b), one can sort the list of feature values and get the node whose feature value is closest to that of \(v_f\). 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 \(v_7\) that has no edge with \(v_6\) but with its feature value closest to the node \(v_7\). Next, the edge \((v_6, v_7)\) will be deleted and node \(v_6\) with \(v_1\) will be connected to get the new edge \((v_6, v_1)\). After that, if the graph \(G\) is connected, the rewiring action can be regarded 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 \(\alpha \ast m\) times in the process of graph augmentation to generate \(G'\), where \(\alpha\) denotes the rewiring cost coefficient and \(m\) denotes the number of edges in \(G\). At the same time, one can set the
Algorithm 2: Approximate Augmentation.

**Input:** Original graph $G$  

**Parameters:** Augmentation cost coefficient $\alpha$, 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 $< \alpha \times m$ do
6: Randomly sample an edge $e_1 = (v_1, v_2)$ without leaf node;
7: Get the set of feature values $S = \{f_v| v = v_1, \ldots, v_n\}$;
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 $w$ that $f_w$ is closest to $f_u$;
11: $e_{del} \leftarrow e_1$;
12: $e_{add} \leftarrow e_2 = (u, w)$;
13: Rewiring to get the $G'$;
14: if $G'$ is connected then
15: swap $= \text{swap} + 1$;
16: else
17: Cancel the rewiring;
18: end if
19: end while
20: end for
21: Get $G_{aug}$ via Eq (15);
22: Return $G_{aug};$

iteration parameter $T$ for this augmentation, and choose the best-augmented graph to return:

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

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

IV. 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 and conduct augmentation testing and analysis. Then we select six comparative methods for comparative experiments to verify the effectiveness of our method. Finally, we analyze the time complexity.

A. Datasets

In order to access our augmentation methods, we adopt seven commonly used benchmark datasets in experiments, which are five small-scale datasets (BZR, COX2, MUTAG, OHSU, ENZYMES), and two large-scale datasets (IMDB-BINARY, IMDB-MULTI). Among them, BZR, COX2, MUTAG, and ENZYMES are biological and chemical datasets, OHSU is a brain dataset, and IMDB-BINARY and IMDB-MULTI are social datasets. Here, we first conduct augmentation experiments and topology feature analysis on five small-scale datasets with the traditional machine learning framework. To further demonstrate the effectiveness of our augmentation model, we add two large-scale social datasets and conduct extensive comparison experiments with the deep learning framework. The statistics of these datasets are summarized in Table II.

| Datasets | $N_G$ | $N_C$ | $\#Nodes$ | $\#Edges$ |
|----------|-------|-------|-----------|-----------|
| BZR | 405 | 437.5 | 38.44 |
| COX2 | 467 | 41.22 | 43.45 |
| MUTAG | 188 | 17.93 | 43.79 |
| OHSU | 79 | 82.01 | 439.66 |
| ENZYMES | 600 | 32.63 | 62.14 |
| IMDB-BINARY | 1000 | 19.77 | 96.53 |
| IMDB-MULTI | 1500 | 13.00 | 65.94 |

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.

- **Relative Gain Ratio:** The relative gain Ratio is defined as:

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

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

Here, we use 16 traditional machine learning combination methods as well as a deep learning-based hierarchical pooling algorithm to validate our approach. We then explore the relationship between the null model augmentation and the contribution of network topological features in the classification process by the Gini importance.

1) Experimental Setup: 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 [63], NetLS [64], gl2vec [65], Graph2vec [51], and Diffpool [52], where SF and Graph2vec are graph embedding methods, NetLS, and gl2vec are graph kernel models, and Diffpool is an end-to-end graph neural network method.

And in this study, the embedding dimension of all graph kernels is 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, and the epochs are 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 intervals 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 [52]. In addition, the first four unsupervised representation methods, SF, Graph2vec, NetLS, 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 a 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 is 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 [31] to filter the augmented graph set. Finally, we feed the augmented training set into the different graph classification classifiers for training.

2) Performance of Augmentation Models: The experimental results conducted on the five datasets with the configuration in Section IV are summarized in Tables III and IV. 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.

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

3) 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 classifier to evaluate the feature importance. Fig. 5 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 consistency between the augmentation effects of the augmentation models and their corresponding features. This phenomenon is more prominent in the standard

![Fig. 5](image-url)
null model-based augmentation. Also, we find that the $2k$ augmentation model on MUTAG dataset has high Gini importance but a relatively low average gain ratio. A possible reason is that the augmentation based on the $2k$ 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. 5(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, their 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. 6. 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 Tables III and IV 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.

### D. Comparison Experiments

Here, to further verify the effectiveness of our methods, we selected the optimal models from the standard null model and the approximate null model augmentation methods and compared them with six data augmentation methods on GCN [44] and GIN [48].

#### 1) Experimental Setup: Specifically, comparison methods are four topology-based methods and two feature-based methods:

- **Node dropping (DropN):** [16]: Node dropping is an augmentation method that randomly drops a certain portion of nodes.
- **Edge perturbation (PermE):** [42]: Modifying the network topology is one of the most common methods in graph generation and augmentation. PermE randomly adds or removes edges with a certain portion.
- **Motif similarity (M-S):** [31]: Motif similarity is a structural-mapping-based data augmentation method for graph classification.
- **Graph Transplant:** [42]: Graph transplant is a node saliency-guided graph mixup method with local structure preservation for improved graph classification performance.

#### Table V

| Encoder | Aug Model | MUTAG | BZR | COX2 | OHSU | ENZYM | I5S | IMDB-BINARY | IMDB-MU_11 |
|---------|-----------|-------|-----|------|------|-------|-----|--------------|-------------|
| original | 0.753 | 0.822 | 0.769 | 0.501 | 0.440 | 0.738 | 0.503 |             |             |
| Graph   | 0.754 | 0.834 | 0.794 | 0.537 | 0.430 | 0.727 | 0.485 |             |             |
| PermE   | 0.777 | 0.836 | 0.773 | 0.429 | 0.438 | 0.721 | 0.465 |             |             |
| ADA-C   | 0.793 | 0.834 | 0.794 | 0.523 | 0.429 | 0.741 | 0.465 |             |             |
| ADA-B   | 0.793 | 0.834 | 0.794 | 0.523 | 0.429 | 0.741 | 0.465 |             |             |
| ADA-C   | 0.793 | 0.834 | 0.794 | 0.523 | 0.429 | 0.741 | 0.465 |             |             |
| ADA-E   | 0.793 | 0.834 | 0.794 | 0.523 | 0.429 | 0.741 | 0.465 |             |             |
| 2k      | 0.803 | 0.845 | 0.807 | 0.559 | 0.492 | 0.745 | 0.510 |             |             |
| ADA-C   | 0.777 | 0.844 | 0.803 | 0.537 | 0.673 | 0.740 | 0.502 |             |             |

*The values underlined in the text represent the suboptimal performance, while the values in bold indicate the optimal performance.*
• **Attribute masking (MaskN): [66]**: MaskN randomly masks node features with a certain portion.

• **Mixup: [43]**: a feature-based augmentation method that mixup graph-level representation.

And we evaluate our augmentation method with GCN and GIN. The following are common settings for GNNS. Among them, the activation function is ReLU, the dropout ratio is 0.5, and the dimension of hidden layers is 128. We set the number of layers of GCN and GIN to 3 and 4 with a global mean pooling layer to get the graph readout. The Adam optimizer with a learning rate of 0.0005 is used to train all models for 1000 epochs. Considering the scale of the OHSU, IMDB-BINARY and IMDB-MULTI, we set the batch size of OHSU, IMDB-BINARY and IMDB-MULTI to 16, 128, 128, while setting the batch size for other datasets to 64. In addition, during the augmentation process, we set the modified ratio to $\alpha = 0.2$ for each augmented model.

2) **Performance of Comparison**: Here, each dataset is split using the 10-fold cross-validation, where 8 folds will be used for training, 1 fold will be used for validation, and the remaining will be used for testing. Table V reports the results of our models and six baselines on GCN and GIN. In most cases, $2k$ can achieve optimal performance, and ADA-Ec closely follows in its footsteps with good performance. The $2k$ augmentation method enhances by 6.5% relative to the original classifiers. And the classification performance of $2k$ is 2.6% better than the optimal baseline on average relative improvement, especially 12.6% on OHSU. It is reasonable since our method introduces the null model with important topological features to ensure label invariance, and reduces noise while enhancing data diversity. No matter the joint degree distribution or the eigenvector centrality is the crucial feature in networks. Such as in biological networks, joint degree distribution can be used to study the interaction relationships between proteins, so as to reveal the internal mechanism of protein networks [67]. While the eigenvector centrality can be used to predict the function and interaction of nodes and can help researchers understand the mechanism and regulatory relationship of proteins and genes in biological networks. Therefore, compared with other methods, the data obtained based on the augmentation of joint degree distribution and eigenvector centrality has higher quality. Although the $2k$ did not perform well in traditional machine learning methods on MUTAG, it achieved good performance in deep learning methods. The main reasons are as follows: 1) Deep learning-based methods have stronger information extraction capabilities than traditional machine learning methods; 2) Compared to methods such as Graph2Vec, GCN and GIN can capture adjacency relationships and topology between nodes, while possessing both local and global awareness capabilities. As a result, they are better suited to capturing topology information, rendering them more expressive and flexible in learning graph representations and performing graph classification tasks.

E. **Analysis of Time Complexity**

Here, we conduct the time complexity analysis of the null model-based augmentation models. Set $n$, $m$, $\alpha$, 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, B, C, C, Ec) are $\Theta(n^2)$, $\Theta(\alpha + m)$, and $\Theta(T \times \alpha + m)$. For baselines, the time cost of motif-similarity mainly arises from calculating vertex similarity and selecting the most similar structures for augmentation, resulting in a complexity of $\Theta(n^2 m)$; Graph Transplant computes node saliency to select subgraphs and performs mixing to generate the new graph, with a complexity of $\Theta(n^2)$. MixupGraph performs a weighted mixup of the two graphs to obtain the new graph. This mixup method does not introduce additional computation and shares the same time complexity as the original GNN model. However, it is worth noting that the generation time of the feature-based baseline is more influenced by the training epochs and batch size. Achieving improved model performance might require longer training epochs, leading to a proportionally larger time cost.

V. **Conclusion**

This paper is the first attempt to combine the null model with data augmentation. To generate augmented data with label invariance and high quality, we introduce the null model to develop several data augmentation models that preserve important topological features during augmentation. To demonstrate the effectiveness of our method, we conduct experiments with lots of classifiers and baseline methods on several benchmark datasets. Through extensive experiments, the 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 network analysis, and it has great potential in the field of graph mining algorithms design. Furthermore, based on the experiment results, we find that: 1) the null models can maintain features consistently with better performance than other methods; 2) the dataset domain features have a significant impact on graph learning, and domain knowledge should be considered during the augmentation process. 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 explore more important features of graph data in graph classification and features that are meaningful for the dataset domain, and develop 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.

ACKNOWLEDGMENT

The authors appreciate all the members of the IVSN Research Group, Zhejiang University of Technology for their valuable discussions.

REFERENCES

[1] G. Siglidi, 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. 1–5, 2020.
[2] P. Goyal and E. Ferrara, “Graph embedding techniques, applications, and performance: A survey,” Knowl.-Based Syst., vol. 151, pp. 78–94, 2018.

[3] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 61–80, Jan. 2009.

[4] J. Gao, J. Gao, X. Ying, M. Lu, and J. Wang, “Higher-order interaction goes neural: A structure assembling graph attention network for graph classification,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 2, pp. 1594–1608, Feb. 2023.

[5] J. Li, Y. Hong, H. Cheng, H. Meng, W. Huang, and J. Huang, “Semisupervised graph classification: A hierarchical graph perspective,” in Proc. World Wide Web Conf., 2019, pp. 972–982.

[6] S. Li, J. Zhou, T. Xu, D. Dou, and H. Xiong, “GeomGCL: Geometric graph contrastive learning for molecular property prediction,” in Proc. AAAI Conf. Artif. Intell., 2022, vol. 36, no. 4, pp. 4541–4549.

[7] A. Goyal and G. Ferrara, “Graph embedding techniques, applications, and performance: A survey,” Knowl.-Based Syst., vol. 151, pp. 78–94, 2018.

[8] Y. Rong, W. Huang, T. Xu, and J. Huang, “M-evolve: Structural-mapping-based data augmentation for graph classification,” IEEE Trans. Neural Netw. Sci. Eng., vol. 8, no. 1, pp. 190–200, Jan.–Mar. 2021.

[9] B. Rieck, C. Bock, and K. Borgwardt, “A persistent Weisfeiler-Lehman graph contrastive learning for molecular property prediction,” in Proc. AAAI Conf. Artif. Intell., 2022, vol. 36, no. 4, pp. 135–146, 2020.

[10] X. Zhuo, J. Shen, S. Yu, C. Chen, and Q. Xuan, “M-evolve: Structural-mapping-based data augmentation for graph classification,” IEEE Trans. Neural Netw. Sci. Eng., vol. 8, no. 1, pp. 190–200, Jan.–Mar. 2021.

[11] A. Gotelli, G. R. Graves, N. J. Gotelli, and C. R. Grase, “Null models in Ecol. (Null Models in Ecology Series). Washington, DC, USA: Smithsonian, 1996.

[12] S. Maslov, K. Sneppen, and A. Zaliznyak, “Detection of topological patterns in complex networks: Correlation profile of the internet,” Physica A: Stat. Mech. Appl., vol. 333, pp. 529–540, 2004.

[13] J. Zhou, Y. Shen, and M. Shah, “Data augmentation for graph neural networks,” in Proc. Adv. Deep Learn. Graph Neural Networks, 2020, pp. 1–13, 2018.

[14] M. Bieddick and K. C. Burns, “A simple null model predicts the island rule,” Ecol. Lett., vol. 24, no. 8, pp. 1646–1654, Aug. 2021.

[15] M. W. Reimann, M. Gevaert, Y. Shi, H. Lu, H. Markram, and E. Muller, “A null model of the mouse white-neocortex micro-connectome,” Nature Commun., vol. 10, no. 1, pp. 1–16, 2019.

[16] P. Mahadevan, D. Kravtsov, F. Rutcheva, and A. Vahdat, “Systematic topology analysis and generation using degree correlations,” ACM SIGCOMM Comput. Commun. Rev., vol. 36, no. 4, pp. 135–146, 2006.

[17] F. Rutcheva, F. Yanardag, and S. V. N. Vishwanathan, “Deep graph kernels,” in Proc. 28th Int. Conf. Neural Inform. Process. Syst., vol. 2, pp. 2224–2323, 2015.

[18] P. Reiser et al., “Graph neural networks for materials science and chemistry,” Commun. Mater., vol. 3, no. 1, 2022, Art. no. 93.

[19] Y. Dou, K. Shu, C. Xia, P. S. Yu, and L. Sun, “User preference-aware fake news detection,” in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2021, pp. 2051–2055.

[20] J. Zhou, J. Shen, and Q. Xuan, “Data augmentation for graph classification,” in Proc. 29th ACM Int. Conf. Inf. Knowl. Manage., 2020, pp. 2341–2344.

[21] K. Hassani and A. H. Khashahmadi, “Contrastive multi-view representation learning on graphs,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 4116–4126.

[22] J. Kong and P. S. Yu, “Semi-supervised feature selection for graph classification,” in Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2010, pp. 793–802.

[23] J. Zhou, J. Shen, and Q. Xuan, “Data augmentation for graph classification,” in Proc. 29th ACM Int. Conf. Inf. Knowl. Manage., 2020, pp. 2341–2344.

[24] X. Kong and P. S. Yu, “Semi-supervised feature selection for graph classification,” in Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2010, pp. 793–802.

[25] Q. Xuan et al., “Subgraph networks with application to structural feature space expansion,” IEEE Trans. Knowl. Data Eng., vol. 33, no. 6, pp. 2776–2789, Jun. 2021.

[26] J. Wang et al., “Sampling subgraph network with application to graph classification,” IEEE Trans. Neural Netw. Sci. Eng., vol. 8, no. 4, pp. 3478–3490, Oct.–Dec. 2021.

[27] 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,” 2020, arXiv:2006.13090.

[28] J. Park, H. Shim, and E. Yang, “Graph transplant: Node saliency-guided graph mixup with local structure preservation,” in Proc. AAAI Conf. Intell. Intell. 2022, vol. 36, no. 7, pp. 7966–7974.

[29] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, “Mixup for node and graph classification,” in Proc. Web Conf. 2021, pp. 3663–3674.

[30] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2017.

[31] C. Gao, J. Zhu, F. Zhang, Z. Wang, and X. Li, “A novel representation learning for dynamic graphs based on graph convolutional networks,” IEEE Trans. Cybern., vol. 53, no. 6, pp. 3599–3612, Jun. 2023.

[32] Z. Wang, C. Wang, X. Li, C. Gao, X. Li, and J. Zhu, “Evolutionary Markov dynamics for network community detection,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 3, pp. 1206–1220, Mar. 2020.

[33] C. Gao, Z. Yin, Z. Wang, X. Li, and X. Li, “Multilayer network community detection: A novel multi-objective evolutionary algorithm based on consensus prior information [feature],” IEEE Comput. Intell. Mag., vol. 18, no. 2, pp. 46–59, May 2022.

[34] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” in Proc. Int. Conf. Learn. Representations, 2019.

[35] N. Shervashidze, P. Schweitzer, E. Jan, V. Leeuwen, and K. M. Borgwardt, “Weisfeiler-Lehman graph kernels,” J. Mach. Learn. Res., vol. 12, no. 3, pp. 2539–2561, 2011.

[36] P. Yanardag and S. V. N. Vishwanathan, “Deep graph kernels,” in Proc. 21th ACM SIGKDD Int. Conf., 2015 pp. 1365–1374.

[37] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, “graph2vec: Learning distributed representations of graphs,” in Proc. 13th Int. Workshop Mining Learn. Graphs, Halifax, Nova Scotia, Canada, 2017.

[38] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” in Proc. 32nd Int. Conf. Neural Inf. Process. Syst., 2018, pp. 4805–4815.

[39] A. Cheng, L. You, S. Wang, and L. Tassiulas, “KerGNN: Interpretable graph neural networks with graph kernels,” in Proc. AAAI Conf. Intell. Intell. 2022, pp. 6614–6622.
Yalu Shan received the B.S. degree in 2019 from the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China, where she is currently working toward the M.S. degree with the College of Information Engineering. Her current research interests include graph data mining and network security.

Shanqing Yu received the M.S. degree from the School of Computer Engineering and Science, Shanghai University, Shanghai, China, in 2008, and the M.S. and Ph.D. degrees from the Graduate School of Information, Production and Systems, Waseda University, Tokyo, Japan, in 2008 and 2011, respectively. She is currently an Associate Professor with the Institute of Cyberspace Security and the College of Information Engineering, Zhejiang University of Technology, Hangzhou, China. Her research interests include intelligent computation and data mining.

Xiaoke Xu (Member, IEEE) received the Ph.D. degree from the College of Information and Communication Engineering, Dalian Maritime University, Dalian, China, in 2008. He was the Postdoctoral Fellow with Hong Kong Polytechnic University, Hong Kong, and a Visiting Scholar with the City University of Hong Kong, Hong Kong. He is currently a Professor with the School of Journalism and Communication, Beijing Normal University, Beijing, China. His current research interests include network community detection and data mining in social networks.

Qi Xuan (Senior Member, IEEE) received the B.S. and Ph.D. degrees in control theory and engineering from Zhejiang University, Hangzhou, China, in 2003 and 2008, respectively. He was a Postdoctoral Researcher with the Department of Information Science and Electronic Engineering, Zhejiang University of Technology, Hangzhou, China. His current research interests include network science, graph data mining, and deep learning.

Guanrong Chen (Fellow, IEEE) received the M.Sc. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 1981, and the Ph.D. degree in applied mathematics from Texas A&M University, College Station, TX, USA, in 1987. Since 2000, he has been a Chair Professor and the Founding Director of the Centre for Chaos and Complex Networks, City University of Hong Kong, prior to that he was a tenured Full Professor with the University of Houston, Texas, USA. He was the recipient of the 2011 Euler Gold Medal from Russia, and conferred Honorary Doctorate by the Saint Petersburg State University, Russia, in 2011 and by the University of Le Havre, Normandy, France, in 2014. He is a Member of the Academy of Europe and a Fellow of The World Academy of Sciences, and is a Highly Cited Researcher in Engineering according to Clarivate Web of Science.