OR-Gate Mixup Multiscale Spectral Graph Neural Network for Node Anomaly Detection

Zekang Li, Ruonan Liu, *Senior Member, IEEE*, Dongyue Chen, and Qinghua Hu, *Senior Member, IEEE*

***Abstract*—Graph node anomaly detection has important applications in practical scenarios. Although many graph neural networks (GNNs) have been proposed, how to design tailored spectral filters for node anomaly detection to fully mine high-frequency signals in the graph is still a challenge. Most GNNs are equivalent to low-pass filters and mine multiorder signals through a series structure. The computational cost increases as the number of layers increases and further leads to an over-smoothing problem. They mainly focus on low-frequency signals and suppress high-frequency signals, thus smoothing the differences between abnormal and normal nodes, making them indistinguishable. Due to the difficulty in mining high-frequency signals, the poorly distinguishable feature representations learned by low-pass GNNs can even harm the performance of data augmentation. To solve the above challenges, in this article, we propose a OR-gate mixup multiscale spectral GNN (MMGNN) from the spectral domain. Specifically, we design multiorder multiscale bandpass filters through the superposition of polynomial spectral filters and then decompose them into preprocessing parts and training parts to form a double-parallel structure, which can effectively mine high-frequency signals in the graph and reduce computational cost. Finally, we propose OR-gate mixup to perform data augmentation in the spectral space to improve model generalization. Experimental results on four real-world datasets demonstrate the effectiveness of the proposed MMGNN against the state-of-the-art methods.**

***Index Terms*—Graph data augmentation, graph neural networks (GNNs), graph pattern mining, graph representation learning, node anomaly detection.**

I. Introduction

ANOMALIES or outliers refer to rare objects that are different from most normal objects. Due to the scarcity of abnormal patterns, anomalies are difficult to detect and often cause huge economic losses once they occur. Therefore, anomaly detection has attracted much attention due to its important applications in many fields, such as equipment fault operation and maintenance [1], [2], financial risk control [3], and telecom fraud detection [4]. It is used to monitor abnormal signals between sensors, detect suspicious transactions in financial networks, identify fraudsters in telecommunications networks, and mine fake reviews on e-commerce shopping platforms. Graph-structured data is ubiquitous in practical scenarios, representing complex interaction relationships through edges (which can dynamically change) connected between multiple entities without positional relationships. Due to the non-Euclidean structure of the graph and the complex dependency between nodes, graph node anomaly detection is more challenging than that of conventional data structures and needs further exploration [5].

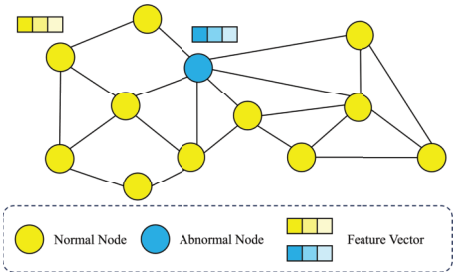

Fig. 1. Connection relationship between normal and abnormal in the graph. In graphs containing anomalies, normal nodes still tend to connect to normal nodes with similar features (low-frequency information), while abnormal nodes tend to connect to normal nodes with dissimilar features (high-frequency information).

With the development of graph neural networks (GNNs), researchers have applied them to node anomaly detection [6], [7], [8], [9], [10]. Most GNNs are proposed based on the homophily assumption (i.e., adjacent nodes have similar features), and they perform well on many graph tasks because most graphs in the real world follow the homophily assumption. However, they perform poorly in node anomaly detection because, in graphs with anomalies, normal nodes still tend to connect to normal nodes with similar features, while abnormal nodes tend to connect to normal nodes with dissimilar features [11], as shown in Fig. 1. The connection of dissimilar nodes results in graphs with anomalies exhibiting heterophily opposite to homophily [12], [13]. Heterophily refers to the phenomenon of edge connections between nodes belonging to different classes (e.g., the abnormal and normal classes in anomaly detection) [14], [15], [16], [17], [18]. Recent studies [19], [20] have shown that abnormal nodes

Received 16 June 2024; revised 20 March 2025; accepted 4 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62206199, in part by the National Key Laboratory of Marine Engine Science and Technology under Grant LAB-2024-04-WD, in part by the Young Elite Scientist Sponsorship Program under Grant YESS20220409, and in part by China Postdoctoral Science Foundation-Tianjin Joint Support Program under Grant 2023T014TJ. *(Corresponding author: Ruonan Liu.)*

Zekang Li, Dongyue Chen, and Qinghua Hu are with the College of Intelligence and Computing and Tianjin Key Laboratory of Machine Learning, Tianjin University, Tianjin 300350, China (e-mail: lizekang6513@tju.edu.cn; dyue_chen@163.com; huqinghua@tju.edu.cn).

Ruonan Liu is with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the National Key Laboratory of Marine Engine Science and Technology, Shanghai 201108, China (e-mail: ruonan.liu@sjtu.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2025.3569413

connecting dissimilar features can cause the spectral energy distribution of the graph to shift to the high frequency, meaning that the spectral energy distribution concentrates less in the low frequency and more in the high frequency. Mining high-frequency signals is beneficial for node anomaly detection. However, the expressive power of GNNs based on homophily is limited to low-pass filters [21], [22], so they can only focus on low-frequency signals and suppress high-frequency signals in the graph, thereby smoothing the difference between abnormal features and normal features, making nodes indistinguishable.

To overcome the shortcomings of low-pass GNNs in node anomaly detection, choosing tailored spectral filters is crucial [19]. Some recent works [19], [20] select multiple sets of filters beyond low pass that focus on both low- and high-frequency signals. However, these filters are usually symmetrical across frequency bands and may treat low- and high-frequency signals equally at the same scale. There is still a risk of remixing low- and high-frequency signals during model training, and only signals of the same order can be mined. To further mine high-order signals, higher-order filters are needed, but as the order increases, the computational cost also increases, which is detrimental to large-scale graph training. Most GNNs mine high-order signals through the series structure, resulting in an over-smoothing phenomenon [21], [23], forcing the representations of adjacent nodes to be similar. The more layers there are, the greater the risk of over-smoothing [24], [25], [26], the more difficult it is to distinguish nodes, and the higher the computational cost. It is a challenge to design beyond low-pass filters that can mine higher-order signals at multiscale for node anomaly detection.

Due to the scarcity of abnormal patterns, most GNNs have poor detection ability for rare abnormal samples. Data augmentation can improve model performance and generalization by adding training data [27]. However, the low-quality representations learned by low-pass GNNs make it difficult to distinguish abnormal nodes from normal nodes and further deteriorate the effectiveness of data augmentation. Meanwhile, current graph data augmentation methods [28], [29], [30] mainly focus on features and structural augmentation, obtaining multiple augmented views through features masking, nodes discarding, edges adding/deleting, and graph random walk. However, features and structural information between augmented views cannot be shared, easily introducing additional computational costs, which is detrimental to large-scale graph training. For node-level and edge-level tasks, nodes are interconnected, and features are not i.i.d. Changes in node features and graph topology may modify the signal distribution on the entire graph, which can also make node features indistinguishable. More efficient graph data augmentation methods need to be designed for detecting difficult anomalies.

To solve the above problems, we propose a OR-gate mixup multiscale spectral GNN (MMGNN) from the spectral domain, which is based on a novel double-parallel structure and differs from the traditional series structure. Specifically, we design multiorder multiscale bandpass filters through the superposition of polynomial spectral filters and then decompose them into the preprocessing part and the training part (i.e., multiplier filters and multiplicand filters), forming the double-parallel structure. We preprocess the graph using parallel multiplier filters to obtain multiorder information and then use parallel multiplicand filters to train the preprocessed graph end-to-end like most GNNs. The double-parallel structure can use more high-order filters to multiscale mine high-frequency signals in the graph within the rated training time, improving the performance. Finally, to overcome the scarcity problem of abnormal patterns, we perform OR-gate mixup data augmentation in the spectral space to improve the generalization of the model. Since our model is based on the double-parallel structure and performs data augmentation in the spectral space, the additional computational cost is less, which is beneficial for large-scale graph training. The main contributions of this article are summarized as follows.

1) By analyzing the superposition and decomposition of polynomial spectral filters, we innovatively proposed a double-parallel structure to decompose designed multiorder multiscale bandpass filters into the nonparametric preprocessing part and the learnable training part, which can fully mine high-frequency signals to improve node anomaly detection performance and reduce training computation. The double-parallel structure provides new inspiration for designing computational acceleration algorithms on large-scale graphs.
2) We propose a spectral-based graph data augmentation method called OR-gate mixup. By performing feature mixups in the spectral space after signals filtering, we reduce computational costs and avoid full graph perturbations caused by changing the original features. Performing OR-gate mixup on labels can avoid the risk of vanilla mixup classification boundary blur, improve model generalization, and reduce overfitting risk.
3) Experimental results on four real-world datasets show that the proposed MMGNN is competitive with the state-of-the-art methods. The experimental analysis also indicates that multiorder bandpass filters are beneficial to node anomaly detection, while multiorder low-pass filters are harmful. These provide experimental support to design tailored spectral filters with better performance for node anomaly detection.

The rest of this article is organized as follows. Section II reviews the related work. Section III provides preliminaries and problem definitions. Section IV introduces the design of our multiorder multiscale bandpass filters and the decomposition method for high-order filters. Section V introduces the proposed MMGNN model. Section VI reports the experimental settings and results. Section VII concludes this article and suggests future work.

II. Related Work

A. Graph Node Anomaly Detection

In the real world, various complex application scenarios generate a large amount of graph-structured data. As an effective method for analyzing graphs, GNNs are widely

used to solve the problem of graph node anomaly detection. DOMINANT [6] built a GCN autoencoder for structure and attributes to optimize the reconstruction loss to find anomalies. GraphConsis [7] constructed multiple relations into a single homograph and then used GNNs and attention mechanisms to aggregate neighborhood information for node anomaly detection. CARE-GNN [12] and PC-GNN [13] constructed multiple homographs based on node relations and adaptively pruned edges according to neighbor distribution. PAMFUL [8] proposed an error-bounded distribution-aware margin loss function to mine abnormal nodes through collaborative pattern mining and feature learning. However, the above methods use GNNs equivalent to low-pass filters for node anomaly detection, which cannot effectively mine high-frequency signals in the graph to further improve the distinguishability between abnormal and normal nodes.

B. Spectral GNNs

Spectral GNNs (SGNNs) are generally GNNs equipped with spectral filters defined from the spectral domain. Bruna et al. [31] developed a graph convolution method based on spectral graph theory. Defferrard et al. [32] reduced computational complexity by using k-order truncation of Chebyshev polynomial as the filter. Kipf and Welling [33] introduced a first-order approximate ChebNet as the filter, avoiding the Laplacian matrix eigendecomposition, so that the model can obtain more distant information in graphs by stacking layers. Other GNNs [34], [35], [36] developed from the spatial domain can be approximated as using first-order low-pass filters, which more focus on low-frequency signals rather than high-frequency signals, and improved expressive power by connecting GNN layers in series. However, for graphs with anomalies, stacking multiple layers of low-pass filters can lead to indistinguishable nodes. Balcilar et al. [22] tried to prove that most GNNs are just equivalent low-pass filters and demonstrated the necessity of bandpass and even high-pass filters for more challenging heterophily graph analysis tasks. By setting a group of learnable convolution kernels for heterophily graphs, Yun et al. [37] obtained multihop-rich information in an approximately parallel manner, but fully parameterized convolutional kernels bring more computational overhead. BWGNN [19] and AMNet [20] selected specific bandpass filters of the same order, which can simultaneously capture low-frequency signals and high-frequency signals for node anomaly detection. However, these are only preliminary attempts at bandpass filters, and the potential performance development of bandpass filters still needs further research.

C. Graph Data Augmentation

Data augmentation techniques are widely used in computer vision [38] and natural language processing [39], and a large amount of domain knowledge is used to design appropriate data transformations to improve generalization and performance. However, the graph data augmentation mechanism has not been clarified due to the complex non-Euclidean structure of graph data, and related exploration is still insufficient. Zhao et al. [40] used a modified graph structure for training, Rong et al. [41] and Feng et al. [42] randomly removed edges and nodes from the training graph, You et al. [43] performed random masking operations on node features, Kong et al. [44] augmented node features with gradient-based adversarial perturbations, and Sun et al. [45] combined features augmentation and structure augmentation. However, these graph augmentation methods that mask/add/delete features or edges are sensitive to operations, and slight perturbations to the graph may change the signal distribution of the entire graph, which is detrimental to node-level classification tasks such as node anomaly detection. Han et al. [46] performed linear interpolation on training samples based on mixup [47] to generate new samples. However, for anomaly detection, reliable soft prediction probabilities are required, and vanilla mixups may reduce the distinguishability between abnormal and normal nodes, resulting in blurred classification boundaries.

III. Preliminaries

In this section, we first introduce the notations and problem definition of node anomaly detection (Section III-A) and then introduce the spectral filters and their commonly used polynomial methods to construct SGNNs (Section III-B).

A. Notations and Problem Definition

We focus on the task of node anomaly detection in attributed graphs. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}\}$ be an undirected graph, $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of all N nodes, $\mathcal{E} = \{e_{ij}\}$ is the set of all edges, $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ denotes the node feature matrix. $\boldsymbol{A}$ represents the adjacency matrix, $\boldsymbol{A}_{ij} = 1$ means there is an edge between v_i and v_j, else $\boldsymbol{A}_{ij} = 0$. $\boldsymbol{D} \in \mathbb{R}^{N\times N}$ is the degree matrix with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. y_i is the ground truth of v_i, $y_i = 0$ indicates normal, and $y_i = 1$ indicates abnormal. Due to the scarcity of abnormal patterns, there is a sample imbalance in anomaly detection, that is, $|\mathcal{V}_a| \ll |\mathcal{V}_n|$, where $\mathcal{V}_a$ is the abnormal set and $\mathcal{V}_n$ is the normal set, satisfying $\mathcal{V}_a \cup \mathcal{V}_n = \mathcal{V}$ and $\mathcal{V}_a \cap \mathcal{V}_n = \emptyset$. The node anomaly detection task can be regarded as a class-imbalanced binary classification problem. In this article, we focus on attribute anomalies and leave structural anomalies to future work.

B. Polynomization of Spectral Filters

The Laplacian matrix $\boldsymbol{L}$ is defined as $\boldsymbol{D} - \boldsymbol{A}$ (regular) or $\boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ (normalized), where $\boldsymbol{I}$ is the identity matrix. According to the theory of graph signal processing [48], $\boldsymbol{L}$ can be eigendecomposed into $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ based on the graph Fourier transform, thereby defining the graph filtering operation, where $\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_N]$ is a diagonal matrix composed of eigenvalues, λ represents the eigenvalues, and $\boldsymbol{U}$ is a matrix composed of eigenvectors corresponding to eigenvalues. For normalized $\boldsymbol{L}$, its eigenvalue range is $[0, 2]$. Different eigenvalues represent information of different frequencies in the graph. A signal $\boldsymbol{x} \in \mathbb{R}^N$ is filtered by a filter g as

$$\boldsymbol{z} = g \star \boldsymbol{x} = \boldsymbol{U} g(\boldsymbol{\Lambda}) \boldsymbol{U}^T \boldsymbol{x} \tag{1}$$

where $g(\cdot)$ is a filter defined in the spectral domain at $[\lambda_1, \lambda_N]$ and $\boldsymbol{z} \in \mathbb{R}^N$ is the filtered frequency signal. When performing

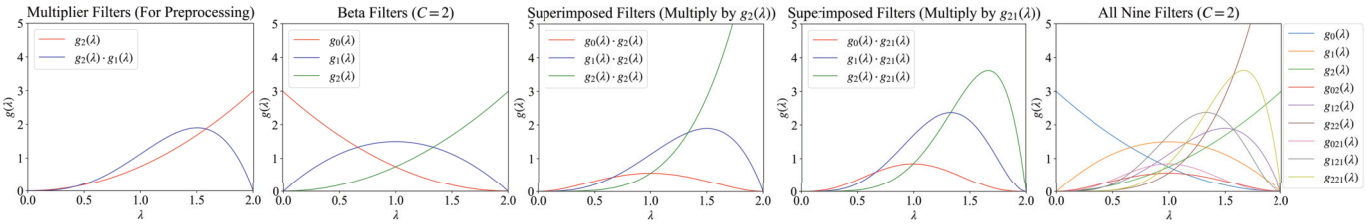

Fig. 2. When $C = 2$. (a) Two multiplier filters, which are second-order and fourth-order, respectively. (b) Vanilla Beta filters are equivalent to second-order. (c) Equivalent fourth-order superimposed filters. (d) Equivalent sixth-order superimposed filters. (e) All nine filters after superposition.

graph filtering operation, the $\boldsymbol{L}$ eigendecomposition of large-scale graphs requires a large amount of computation, so the polynomial fitting filter function method [32] is generally used to reduce the computational complexity

$$g \star \boldsymbol{x} = \boldsymbol{U}\left(\sum_{k=0}^{K}\alpha_k\boldsymbol{\Lambda}^k\right)\boldsymbol{U}^T\boldsymbol{x} = \sum_{k=0}^{K}\alpha_k\boldsymbol{U}\boldsymbol{\Lambda}^k\boldsymbol{U}^T\boldsymbol{x} = \sum_{k=0}^{K}\alpha_k\boldsymbol{L}^k\boldsymbol{x} = g(\boldsymbol{L})\boldsymbol{x} \tag{2}$$

where α_k is a polynomial parameter and $g(\boldsymbol{L})$ is a kth-order polynomial filter function. For the node feature matrix $\boldsymbol{X} \in \mathbb{R}^{N\times d}$, there is a matrix form

$$\boldsymbol{Z} = g(\boldsymbol{L})\boldsymbol{X}. \tag{3}$$

It is worth noting that since abnormal nodes are often cunningly connected to large amounts of normal neighborhoods, signals in the graph are biased toward high-frequency [19]. Most GNNs are equivalent to low-pass filters, focusing on low-frequency signals and suppressing high-frequency signals, thus smoothing the difference between abnormal and normal nodes, resulting in poor performance. Therefore, designing tailored spectral filters that focus more on high-frequency signals is the key to improving node anomaly detection performance.

IV. Multiorder Multiscale Filters Design

In this section, we first introduce the self-superposition of same-order filters to construct our multiorder multiscale bandpass filters that focus more on high-frequency signals (Section IV-A). Then, we decompose the multiorder multiscale bandpass filters into the preprocessing part and the training part to reduce computational cost (Section IV-B). The designed multiorder multiscale bandpass filters are the key to our proposed MMGNN model.

A. Multiorder Multiscale Bandpass Filters

In node anomaly detection, the spectral energy distribution in graphs shifts to the high-frequency part [19] due to the presence of abnormal nodes, thus models focusing on multiscale high-frequency signals can better learn the difference between abnormal and normal features to improve node distinguishability. To overcome the shortcoming of low-pass filters that ruthlessly filter out high-frequency signals, we use graph wavelet transform to design our multiorder multiscale bandpass filters.

Wavelets have proved to be an exceptionally useful tool for signal processing, they can simultaneously localize signal content in both space and frequency. Hammond et al. [49] proposed the graph wavelet transform, using the basis composed of the eigen functions of the Laplacian matrix $\boldsymbol{L}$ to construct a group of wavelets from the "mother" wavelet ψ as $\mathcal{W} = (\mathcal{W}_{\psi_0}, \mathcal{W}_{\psi_1}, \ldots, \mathcal{W}_{\psi_C})$. A single graph wavelet transform can be written as

$$\mathcal{W}_{\psi_i}(\boldsymbol{X}) = \boldsymbol{U}g_i(\boldsymbol{\Lambda})\boldsymbol{U}^T\boldsymbol{X} = g_i(\boldsymbol{L})\boldsymbol{X}. \tag{4}$$

This is not much different from graph convolution used by most GNNs. However, according to the Parseval theorem, the graph wavelet transform needs to satisfy the admissible condition

$$\int_0^\infty \frac{|g_i(w)|^2}{w}dw = C_g < \infty. \tag{5}$$

This guarantees that $g_i(\cdot)$ performs like a bandpass filter rather than a low-pass filter in the spectral domain. Graph wavelet transform can cover different frequency bands through a group of parallel filters $\{g_0, g_1, \ldots, g_C\}$. However, for the same group of filter functions, their orders are the same. Here, we specifically use the discrete polynomial Beta wavelet filter function [19], [50]

$$\mathcal{W}_{p,q} = \boldsymbol{U}\beta_{p,q}(\boldsymbol{\Lambda})\boldsymbol{U}^T = \beta_{p,q}(\boldsymbol{L}) = \frac{\left(\frac{\boldsymbol{L}}{2}\right)^p\left(\boldsymbol{I}-\frac{\boldsymbol{L}}{2}\right)^q}{2B(p+1,q+1)} \tag{6}$$

where $p, q \in \mathbb{N}$ and $B(p+1, q+1) = p!q!/(p+q+1)!$ is a constant. Constraining $p+q = C \in \mathbb{N}^+$, then a group of $C+1$ filter functions of the same c-order is formed. When $C = 2$, all three filter functions are shown in Fig. 2(b). We simply write $\beta_{0,2}(\lambda) = g_0(\lambda)$, $\beta_{1,1}(\lambda) = g_1(\lambda)$, and $\beta_{2,0}(\lambda) = g_2(\lambda)$.

Fig. 2(b) shows that Beta filters include not only low-pass but also bandpass and high-pass filters, which can capture both low-frequency and high-frequency signals. But when C is determined, the same group of filters are of the same order. To further mine multiorder high-frequency signals, it is necessary to define multiple groups of filters using different C. However, when C is larger and the order is higher, the computational cost increases, which is bad for large-scale graph training. At the same time, vanilla Beta filters are symmetrical in the frequency band, which may treat low-frequency and high-frequency signals equally, making it difficult to mine multiscale differential

information. There is still a risk of remixing low- and high-frequency signals during model training, resulting in poor performance.

To overcome the above shortcomings of vanilla Beta filters, we further design our multiorder multiscale bandpass filters. After filters polynomialization, $\beta_{p,q}(\lambda)$ is a polynomial in λ. Intuitively, we can obtain multiorder filters by superimposing (i.e., multiplying) polynomial filters of different orders or the same order, because generally, the multiplication of two polynomials is still a polynomial.

Definition 1 (Polynomial Multiplication): Let $p(x)$ and $q(x)$ be polynomials of order r_p and r_q on the field P, respectively, and $r_p, r_q \in \mathbb{N}^+$. Define $p(x)q(x) = f(x)$ as the polynomial multiplication, and the order of $f(x)$ is r_f, satisfying $r_f = r_p + r_q$.

Since the superposition of polynomial filters of different orders inherently involves the calculation of multiple filters of different orders, particularly, we analyze the self-superposition of same-order filters. Here, we still analyze the group of vanilla Beta filters when $C = 2$ in Fig. 2(b), selecting $g_1(\lambda)$ and $g_2(\lambda)$ to construct two multiplier filters (to distinguish them from multiplicand filters) for subsequent superposition operations. The first multiplier filter is $g_2(\lambda)$, and the second multiplier filter is denoted as $g_{21}(\lambda)$, calculated as follows:

$$g_2(\lambda) \cdot g_1(\lambda) = g_{21}(\lambda). \tag{7}$$

The curves of these two multiplier filters $g_2(\lambda)$ and $g_{21}(\lambda)$ are shown in Fig. 2(a). It can be seen that they are both beyond low-pass filters. Then, we use the three vanilla Beta filters of group$_0$ in Fig. 2(b) as the multiplicand filters and multiply them with the multiplier filters in full arrangement

$$\left.\begin{aligned} g_0(\lambda) \cdot g_2(\lambda) &= g_{02}(\lambda) \\ g_1(\lambda) \cdot g_2(\lambda) &= g_{12}(\lambda) \\ g_2(\lambda) \cdot g_2(\lambda) &= g_{22}(\lambda) \end{aligned}\right\} \text{group}_1 \tag{8}$$

$$\left.\begin{aligned} g_0(\lambda) \cdot g_{21}(\lambda) &= g_{021}(\lambda) \\ g_1(\lambda) \cdot g_{21}(\lambda) &= g_{121}(\lambda) \\ g_2(\lambda) \cdot g_{21}(\lambda) &= g_{221}(\lambda) \end{aligned}\right\} \text{group}_2. \tag{9}$$

Now, we get group$_1$ and group$_2$, two groups of new filters, where group$_1$ is shown in Fig. 2(c) and group$_2$ is shown in Fig. 2(d). It is worth noting that after superposition, the new filters of group$_1$ are equivalent to fourth-order, and the new filters of group$_2$ are equivalent to sixth-order, both of which have bandpass and high-pass properties. The vanilla Beta filters of group$_0$ are equivalent to second-order. The filters of group$_0$, group$_1$, and group$_2$ together form our multiorder multiscale bandpass filters. They focus more on multiscale high-frequency signals in the graph, fully mining the differences between abnormal and normal features, thus improving node distinguishability.

Without loss of generality, the superposition between vanilla filters of different orders also has similar properties. In Section VI-H, we will further analyze the impact of spectral filter properties on node anomaly detection.

B. Decomposition of High-Order Filters

As shown in Fig. 2(e), by superposing Beta filters of the same order, we have constructed our multiorder multiscale bandpass filters. For $C = 2$, a total of nine filters can be obtained after superposition, which is more effective for capturing high-frequency differential signals between nodes. However, additional high-order filters will increase the computational cost, which is detrimental to large-scale graph training. Therefore, we provide a method to reduce the computational cost of high-order filters. For polynomial filters, the multiplication of two polynomials is still a polynomial, and conversely, a high-order reducible polynomial can be decomposed into two lower-order polynomials.

Definition 2 (Reducible Polynomial): Let $f(x)$ be a polynomial of order r_f on the field P. If there are nonconstant polynomials $p(x)$ and $q(x)$ of order less than r_f on P, satisfying $f(x) = p(x)q(x)$ and $r_f = r_p + r_q$, then $f(x)$ is defined as reducible polynomial on the field P, otherwise irreducible polynomial.

Therefore, for a high-order reducible polynomial filter $g_f(\lambda)$, it can be split into two low-order filters, the multiplicand filter $g_p(\lambda)$ and the multiplier filter $g_q(\lambda)$

$$g_f(\lambda) = g_p(\lambda) \cdot g_q(\lambda). \tag{10}$$

The order relationship satisfies $r_f, r_p, r_q \in \mathbb{N}^+$ and $r_f = r_p + r_q$. So, in the graph wavelet transform, the following calculation decomposition can be performed:

$$\begin{aligned} g_f(\boldsymbol{L})\boldsymbol{X} &= \left[g_p(\boldsymbol{L}) \cdot g_q(\boldsymbol{L})\right]\boldsymbol{X} \\ &= g_p(\boldsymbol{L})\left[g_q(\boldsymbol{L})\boldsymbol{X}\right] \\ &= g_p(\boldsymbol{L})\boldsymbol{X}_q \end{aligned} \tag{11}$$

where $\boldsymbol{X}_q = g_q(\boldsymbol{L})\boldsymbol{X}$. This means that for complex high-order filtering, we can calculate $\boldsymbol{X}_q$ first and then $g_f(\boldsymbol{L})\boldsymbol{X} = g_p(\boldsymbol{L})\boldsymbol{X}_q$. Due to the good inductive bias of bandpass filters, we can decompose the designed multiorder multiscale band-bass filters into a nonparametric preprocessing part (using multiplier filters) and a learnable training part (using multiplicand filters), thereby reducing the computational cost of high-order filtering, which is beneficial for large-scale graphs training. The double-parallel structure formed by the decomposition method will be introduced in Section V-A.

V. Proposed Model

In this section, we introduce our proposed OR-gate MMGNN in detail. The overall framework is shown in Fig. 3. First, by decomposing the designed multiorder multiscale bandpass filters into the preprocessing part and the training part, a double-parallel structure is formed to effectively mine high-frequency signals and reduce computational cost (Section V-A). Then, we propose OR-gate mixup to perform data augmentation in the spectral space after parallel filtering in the training part, avoiding additional views and improving model generalization (Section V-B). Finally, we discuss the differences between our proposed MMGNN and previous works and summarize our highlights (Section V-C).

A. Double-Parallel Structure

In Section IV-A, we have constructed our multiorder multiscale bandpass filters. Now, to reduce the computational

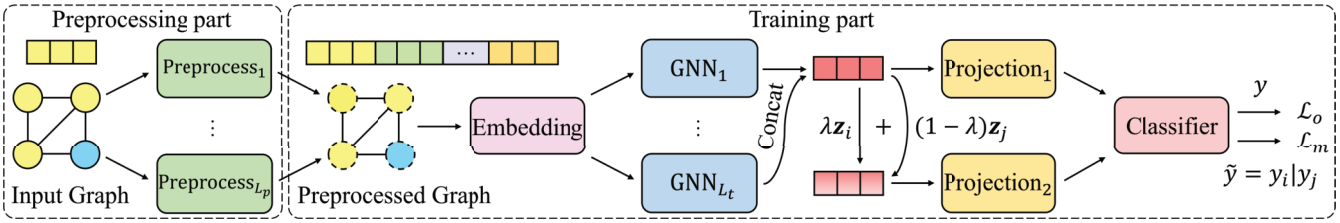

Fig. 3. Framework of the proposed MMGNN. Decompose the multiorder multiscale bandpass filters into a double-parallel structure consisting of a preprocessing part and a training part. First, obtain multiorder information through the multiplier filters in the preprocessing part and then use another learnable multiplicand filter to capture multiorder and multiscale frequency signals from the preprocessed graph and concatenate them. Perform OR-gate mixup in the filtered spectral space. The concatenated signals and mixup signals are finally transmitted to the classifier to obtain the prediction results.

cost of parallel high-order filtering, we decompose them into preprocessing parts and training parts. We first use the multiplier filters $g_2(\lambda)$ and $g_{21}(\lambda)$ constructed in Section IV-A to preprocess the graph without parameterization to extract high-order information

$$\boldsymbol{X}_2 = g_2(\boldsymbol{L})\boldsymbol{X} \tag{12}$$

$$\boldsymbol{X}_{21} = g_{21}(\boldsymbol{L})\boldsymbol{X} \tag{13}$$

where $\boldsymbol{X}_2 \in \mathbb{R}^{N\times d}$ is the second-order feature matrix and $\boldsymbol{X}_{21} \in \mathbb{R}^{N\times d}$ is the fourth-order feature matrix. We keep the input and output dimensions of filters consistent. Then concatenate them along the feature dimension

$$\boldsymbol{X}_m = [\boldsymbol{X}; \boldsymbol{X}_2; \boldsymbol{X}_{21}] \tag{14}$$

where $\boldsymbol{X}_m \in \mathbb{R}^{N\times 3\ d}$ is the multiorder feature matrix. The above preprocessing for extracting multiorder information is the first parallel structure of the double-parallel structure. The preprocessed graph is then fed into the training model for end-to-end training. First, the multiorder features are embedded

$$\boldsymbol{H} = \text{Embedding}(\boldsymbol{X}_m) \tag{15}$$

where Embedding$(\cdot)$ is a simple two-layer MLP with ReLU activation function. Then, the multiplicand filters [i.e., vanilla Beta filters with $C = 2$ in Fig. 2(b)] described in Section IV-A are used for parallel filtering [19] in the training part, and the frequency signal $\boldsymbol{Z}_k$ filtered by the kth multiplicand filter is

$$\boldsymbol{Z}_k = g_k(\boldsymbol{L})\boldsymbol{H}. \tag{16}$$

The parallel structure of the preprocessing part and the training part jointly constitute the double-parallel structure, completing the filtering operations of our designed multiorder multiscale bandpass filters, which can fully mine high-frequency signals in the graph. Because the filtered signals obtained by parallel filtering in the training part have different frequency characteristics, we directly concatenate them along the feature dimension

$$\boldsymbol{Z} = [\boldsymbol{Z}_0; \boldsymbol{Z}_1; \ldots; \boldsymbol{Z}_C] \tag{17}$$

where $\boldsymbol{Z}$ is the multifrequency signal matrix in the spectral space after concatenating.

Overall, decomposing the constructed multiorder multiscale bandpass filters into the double-parallel structure not only avoids the additional computational cost of high-order filtering but also avoids the feature transformation

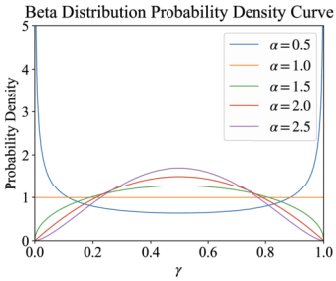

Fig. 4. Beta distribution probability density curve.

of equal status for all filtered signals during training, preserving good inductive bias of bandpass filters, which is a balance between parametric and nonparametric. Due to the extraction of high-order information in the preprocessing part, only second-order computations are needed during training to simultaneously mine second-, fourth-, and sixth-order high-frequency signals.

B. OR-Gate Mixup

Benefiting from the excellent signal mining ability of our designed multiorder multiscale bandpass filters, we argue that data augmentation in the filtered spectral space (rather than the original feature space) is beneficial, without adding additional views to increase the computational cost. So, we propose OR-gate mixup to improve model generalization and alleviate the risk of overfitting during training. Specifically, we perform linear interpolation [47] between multifrequency signals z of training samples after parallel filtering in the training part to construct virtual samples

$$\tilde{\boldsymbol{z}} = \gamma \boldsymbol{z}_i + (1-\gamma)\boldsymbol{z}_j \tag{18}$$

where $\tilde{\boldsymbol{z}}$ is the mixup multifrequency signal and $(\boldsymbol{z}_i, y_i)$ and $(\boldsymbol{z}_j, y_j)$ are two samples randomly drawn from the training data. $\gamma \sim \text{Beta}(\alpha, \alpha)$ is sampled from the Beta distribution, and $\gamma \in [0, 1]$. The parameter α controls the shape of the Beta distribution, indirectly controlling the strength of interpolation between two samples. In practice, we set α to 2.0. The Beta distribution probability density curve is shown in Fig. 4. Vanilla mixup [47] does the same interpolation for labels; however, this can lead to blurred classification boundaries. Differently, we design OR-gate mixup for labels

$$\tilde{y} = y_i | y_j. \tag{19}$$

The label combination truth table of OR-gate mixup is shown in Table I. Intuitively, as long as there is a tiny real

TABLE I
OR -GATE MIXUP TRUTH TABLE

Inputs		Output
y_i	y_j	$\tilde{y}$
0	0	0
0	1	1
1	0	1
1	1	1

abnormal information in the signal, we should consider it 100% as an abnormal category. Such abnormal categories are often difficult to detect, but our OR-gate mixup can construct such potential samples to improve the detection ability for difficult abnormal samples. OR-gate mixup makes fewer changes to the distribution of abnormal and normal quantities while performing data augmentation. At the same time, it directly operates on node-level features, which causes less disturbance to the signal distribution of the entire graph and is beneficial for improving model generalization.

Finally, z and $\tilde{z}$ are sent to their respective projection heads (one-layer MLP with the ReLU activation function) for transformation and fed into the shared classifier (two-layer MLP with the ReLU activation function) to output the prediction probabilities p_i and $\tilde{p}_i$

$$p_i = \text{classifier}\left(\text{projection}_1\left(z_i\right)\right) \tag{20}$$

$$\tilde{p}_i = \text{classifier}\left(\text{projection}_2\left(\tilde{z}_i\right)\right). \tag{21}$$

We use weighted cross-entropy as the loss function

$$\mathcal{L}_o = \sum_i \left(w y_i \log\left(p_i\right) + \left(1 - y_i\right) \log\left(1 - p_i\right)\right) \tag{22}$$

$$\mathcal{L}_m = \sum_i \left(\tilde{w} \tilde{y}_i \log\left(\tilde{p}_i\right) + \left(1 - \tilde{y}_i\right) \log\left(1 - \tilde{p}_i\right)\right) \tag{23}$$

where w and $\tilde{w}$ are the ratio of normal labels to abnormal labels. The overall objective function is $\mathcal{L} = \beta_1 \mathcal{L}_o + \beta_2 \mathcal{L}_m$, and β_1 and β_2 are hyperparameters for balancing $\mathcal{L}_o$ and $\mathcal{L}_m$, and we set β_1 and β_2 to 0.5. When testing the performance of the model, we only use the prediction probability p_i.

Algorithm 1 Preprocessing Part of the MMGNN

Input: Attributed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X} \in \mathbb{R}^{N \times d}\}$; Vanilla Beta filters $g_1(\lambda)$ and $g_2(\lambda)$ when $C = 2$.
Output: Preprocessed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}_m \in \mathbb{R}^{N \times 3\ d}\}$.

$\boldsymbol{X}_2 \leftarrow g_2(\boldsymbol{L})\boldsymbol{X}$;
$\boldsymbol{X}_{21} \leftarrow g_1(\boldsymbol{L})\boldsymbol{X}$;
$\boldsymbol{X}_{21} \leftarrow g_2(\boldsymbol{L})\boldsymbol{X}_{21}$;
$\boldsymbol{X}_m \leftarrow [\boldsymbol{X}; \boldsymbol{X}_2; \boldsymbol{X}_{21}]$;

The pseudocode of our proposed MMGNN model is described in Algorithms 1 and 2.

C. Differences and Highlights

Here, we discuss the differences between our proposed MMGNN and previous works and summarize our highlights. The series structure in Fig. 5(a) captures long-distance information in graphs by stacking multiple identical GNNs. The

Algorithm 2 Training Part of the MMGNN

Input: Preprocessed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}_m \in \mathbb{R}^{N \times 3\ d}\}$; Vanilla Beta filters $g_0(\lambda)$, $g_1(\lambda)$ and $g_2(\lambda)$ when $C = 2$; Beta distribution parameter α.
Output: Category prediction probability p_i for all nodes.

for *each iteration* **do**
  $\boldsymbol{H} \leftarrow \text{Embedding}(\boldsymbol{X}_m)$;
  for $k = 0, \ldots, C$ **do**
    $\boldsymbol{Z}_k \leftarrow g_k(\boldsymbol{L})\boldsymbol{H}$;
  end for
  $\boldsymbol{Z} \leftarrow [\boldsymbol{Z}_0; \boldsymbol{Z}_1; \ldots; \boldsymbol{Z}_C]$;
  $\widetilde{\boldsymbol{Z}}^{train}, \widetilde{\boldsymbol{Y}}^{train} \leftarrow$ shuffle the arrangement of node features and labels in the training set $(\boldsymbol{Z}^{train}, \boldsymbol{Y}^{train})$;
  $\gamma \leftarrow$ sample from Beta distribution $\text{Beta}(\alpha, \alpha)$;
  for $i = 0, \ldots, N^{train}$ **do**
    $\tilde{\boldsymbol{z}}_i^{train} \leftarrow \gamma \boldsymbol{z}_i^{train} + (1 - \gamma)\tilde{\boldsymbol{z}}_i^{train}$;
    $\tilde{y}_i^{train} \leftarrow y_i^{train} \mid \tilde{y}_i^{train}$;
  end for
  $P \leftarrow \text{classifier}(\text{projection}_1(\boldsymbol{Z}))$;
  $\widetilde{P}^{train} \leftarrow \text{classifier}(\text{projection}_2(\widetilde{\boldsymbol{Z}}^{train}))$;
  Use the training set to update model parameters by minimizing $\mathcal{L}$;
end for

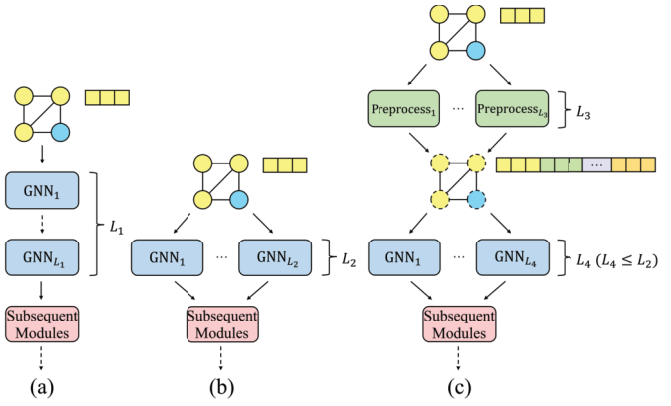

Fig. 5. Comparison of two common GNN structures and our double-parallel structure of the MMGNN. (a) Series structure. (b) Parallel structure. (c) Double-parallel structure.

parallel structure in Fig. 5(b) obtains different types of information in graphs by parallelizing multiple GNNs. Unlike GNNs with series stacked layers [33], [35] and GNNs with multiple filters in parallel [19], [20], our novel double-parallel structure in Fig. 5(c) decomposes the designed multiorder multiscale bandpass filters into the preprocessing part and the training part, so that the number of parallel layers L_4 can be smaller than L_2, which can reduce computational cost and be beneficial for large-scale graph training. Meanwhile, most preprocessing methods [28], [51] such as random walk, adding distance coding, usually extract general explicit information from the spatial domain, while our preprocessing method uses bandpass filters with a good inductive bias to directly extract multiorder implicit information from the spectral domain.

In terms of highlights, our multiorder multiscale bandpass filters can fully mine high-frequency signals of different scales

TABLE II
STATISTICS OF THE REAL-WORLD DATASETS

Dataset	# Nodes	# Edges	# Features	# Heterophily	Anomaly (%)
Amazon	11,944	4,398,392	25	0.0456	6.87
YelpChi	45,954	3,846,979	32	0.2268	14.53
T-Finance	39,357	21,222,543	10	0.0292	4.58
T-Social	5,781,065	73,105,508	10	0.3761	3.01

in the graph to improve the performance of node anomaly detection. The proposed OR-gate mixup performs data augmentation in the spectral space without adding additional views, avoids the risk of classification boundary blur, improves model generalization, and reduces overfitting risk.

VI. EXPERIMENT

In this section, we comprehensively evaluate the performance of our proposed MMGNN on four public real-world datasets and conduct the following experiments and analysis: 1) the effectiveness of the MMGNN for node anomaly detection compared with other state-of-the-art methods (Section VI-E); 2) the training speed of the MMGNN compared to other methods on large-scale graphs (Section VI-F); 3) ablation study of multiorder multiscale bandpass filters and OR-gate mixup of the MMGNN (Section VI-G); 4) the benefits of multiorder bandpass filters for node anomaly detection and the harmfulness of multiorder low-pass filters (Section VI-H); and 5) parameter sensitivity analysis of the OR-gate mixup (Section VI-I).

A. Datasets

We conduct experiments on four public real-world datasets. The dataset's statistics are summarized in Table II. The heterophily degree of a graph $\mathcal{G}$ can be defined as hetero($\mathcal{G}$) = $\sum_{(i,j)\in\mathcal{E}} \mathbb{I}\{\mathbf{y}_i \neq \mathbf{y}_j\}/|\mathcal{E}|$, where $|\mathcal{E}|$ is the total number of edges and $\mathbb{I}$ is an indicator function.

1) *Amazon:* Amazon [52] is a spam review dataset that includes product reviews under the Musical Instrument category. The goal is to find anomalous users who paid under the Musical Instrument category on Amazon.com but write fake product reviews. The graph contains three types of edges: U-P-U (users who have reviewed at least one same product), U-S-U (users who have reviewed at least one product with the same star rating within a week), and U-V-U (users who rank in the top 5% of the similarity rankings in mutual reviews).
2) *YelpChi:* YelpChi [53] is a spam review dataset that contains hotel and restaurant reviews filtered and recommended by Yelp. The goal is to find anomalous reviews that unfairly promote or demote certain products or businesses on Yelp.com. The graph contains three types of edges: R-U-R (reviews posted by the same user), R-S-R (reviews with the same star rating for the same product), and R-T-R (reviews posted for the same product in the same month).
3) *T-Finance:* T-Finance [19] is a transaction dataset aimed at discovering anomalous accounts in the trading network. The nodes are unique anonymized accounts with 10-D features related to registration date, logging activities, and interaction frequency. The edges in the graph represent two accounts with transaction records. Nodes belonging to categories such as fraud, money laundering, and online gambling will be labeled as anomalous by human experts.
4) *T-Social:* T-Social [19] is a transaction dataset aimed at discovering anomalous accounts in the trading network. It has the same node annotations and features as T-Finance. The edge connections between nodes indicate that they have maintained friendship for more than 3 months. The scale of T-Social is 100 times larger than that of Amazon and YelpChi.

B. Baselines

We compare our proposed MMGNN with two classes of baselines: 1) general GNN methods, including GCN [33], ChebyNet [32], GAT [35], GIN [54], GraphSAGE [36], and GWNN [55] and 2) state-of-the-art graph-based anomaly detection methods, including GraphConsis [7], CAREGNN [12], PC-GNN [13], AMNet [20], GDN [14], GHRN [15], and BWGNN [19].

1) *GCN:* It is a spectral-based graph convolutional network that uses a first-order approximation of localized spectral filters to learn node embedding graphs.
2) *ChebyNet:* It is a spectral-based graph convolutional network that restricts the convolutional kernel to a Chebyshev polynomial.
3) *GAT:* It is a GNN based on an attention mechanism, which adaptively aggregates neighborhood features by learning edge weights to learn node embeddings.
4) *GIN:* Inspired by the Weisfeiler–Lehman (WL) graph isomorphism test, GIN models injective function by designing aggregation patterns with sufficient expressive power. The expressive power of GIN is comparable to the WL test.
5) *GraphSAGE:* GraphSAGE first uses the connection information between nodes to sample a fixed number of neighbors and then aggregates neighborhood information through multilayer aggregation functions.
6) *GWNN:* It is a spectral GNN that uses heat kernels to perform graph wavelet transform.
7) *GraphConsis:* It is a heterogeneous GNN used to tackle context, feature, and relation inconsistency problems in graph anomaly detection.
8) *CAREGNN:* It is a camouflage-resistant GNN that enhances the aggregation process with three unique modules against camouflages and reinforcement learning.
9) *PC-GNN:* It is an imbalanced learning method based on the GNN, which solves the class imbalance problem in graph-based fraud detection via resampling.
10) *AMNet:* It is a spectral GNN designed to adaptively capture low-frequency and high-frequency signals through stacking multiple BernNets.
11) *GDN:* It is a graph decomposition network that disentangles node features into class and surrounding features,

TABLE III
F1-MACRO, AUC, AUPRC, AND REC@K OF ALL COMPARED METHODS. THE BEST PERFORMANCE IN EACH DATASET IS HIGHLIGHTED IN BOLD

Dataset	Amazon				YelpChi				T-Finance				T-Social			
Metric	F1-macro	AUC	AUPRC	Rec@K	F1-macro	AUC	AUPRC	Rec@K	F1-macro	AUC	AUPRC	Rec@K	F1-macro	AUC	AUPRC	Rec@K
GCN	74.53±1.38	87.12±0.91	51.79±4.23	53.91±1.95	53.33±0.21	54.41±0.43	17.40±0.15	20.29±0.33	82.35±2.91	90.26±1.24	65.61±6.95	63.45±5.02	63.82±2.35	80.28±4.94	22.24±6.68	27.49±7.02
ChebyNet	91.93±0.64	94.86±2.39	85.89±2.16	83.15±0.84	66.04±0.50	77.95±0.32	41.53±1.24	42.00±0.85	82.46±1.48	91.39±0.69	65.13±4.65	66.07±2.66	62.25±3.16	76.63±2.81	19.57±6.63	26.36±6.27
GAT	83.52±10.28	89.89±5.69	68.98±19.51	67.70±16.85	54.10±0.89	56.13±1.46	19.32±1.31	21.65±1.35	74.70±1.26	88.14±1.85	44.11±4.30	50.73±2.70	63.93±0.95	86.27±1.80	21.05±3.01	28.49±2.44
GIN	66.77±1.97	77.72±4.05	29.19±5.46	39.48±6.00	52.76±0.91	54.71±1.62	17.57±1.25	19.34±1.50	77.67±2.72	77.32±2.51	38.28±7.27	54.18±5.72	65.65±6.13	70.36±8.68	24.31±10.19	32.93±13.50
GraphSAGE	92.14±0.39	93.25±2.09	84.35±2.10	82.03±1.21	66.40±0.70	78.85±0.94	42.63±1.81	42.51±1.36	82.25±2.84	91.23±1.52	61.58±10.64	64.66±5.02	58.45±2.57	76.30±2.72	11.21±2.89	17.65±6.53
GWNN	86.28±1.94	93.65±1.01	80.35±2.79	74.76±3.45	53.70±0.31	57.12±2.45	18.37±1.20	20.95±0.91	77.26±1.96	85.85±1.95	51.38±5.59	53.74±3.51	62.01±1.81	74.58±1.62	14.36±2.59	25.84±4.50
GraphConsis	71.93±1.43	86.00±0.58	45.37±3.02	49.09±3.19	54.07±0.78	56.29±1.61	18.99±1.26	21.41±1.42	73.15±2.13	84.00±2.30	42.65±4.10	48.36±3.01	59.61±1.33	79.76±1.45	13.44±2.11	21.05±2.56
CAREGNN	90.52±1.09	94.22±1.76	85.02±2.26	80.73±2.31	60.11±2.54	72.47±2.89	30.10±3.32	33.26±3.08	77.27±3.19	86.78±4.71	53.51±7.94	57.85±3.72	49.61±0.38	54.14±5.35	3.35±0.49	1.92±0.60
PC-GNN	91.94±0.49	90.58±1.43	82.23±0.94	81.64±0.65	66.24±0.36	76.88±0.49	41.76±1.02	42.35±0.66	85.78±0.56	90.15±0.58	70.56±1.73	70.32±0.83	51.71±1.28	63.94±2.51	4.78±0.60	4.86±1.59
AMNet	91.50±0.38	95.15±0.78	86.08±1.29	82.21±1.40	65.74±0.47	78.05±0.84	40.34±1.45	41.54±1.03	84.37±1.83	92.05±1.42	67.72±5.19	67.70±2.50	59.02±0.74	77.21±1.28	12.53±1.29	19.90±1.56
GDN	82.67±6.79	90.77±1.38	80.89±2.46	76.52±2.85	59.09±1.63	72.40±1.43	32.12±3.58	34.71±3.02	86.82±2.59	93.55±0.83	77.50±4.57	72.80±3.85	-	-	-	-
GHRN	91.95±0.48	97.31±0.30	88.72±0.81	84.42±0.98	69.68±0.74	82.57±0.74	48.74±1.70	48.30±1.25	88.42±0.63	95.13±0.73	82.73±1.02	76.77±1.19	79.63±6.45	93.48±1.10	55.56±16.68	58.02±17.21
BWGNN	92.10±0.27	97.15±1.13	88.73±1.49	84.55±1.28	70.23±0.81	83.20±0.80	50.87±2.01	49.25±1.49	88.75±0.75	95.28±0.22	83.02±1.03	77.77±1.45	83.54±5.27	94.52±2.10	66.02±13.79	67.70±9.97
MMGNN	**92.43±0.46**	**97.81±0.23**	**89.66±0.57**	**85.18±0.42**	**73.10±0.54**	**84.05±0.71**	**56.74±1.28**	**54.14±0.98**	**89.80±0.45**	**96.17±0.32**	**85.81±1.04**	**80.08±0.45**	**94.73±0.16**	**98.14±0.45**	**91.72±0.49**	**87.99±0.49**

effectively identifying anomaly features invariant to heterophily shift and capturing the local homophily of normals.

12) *GHRN:* It is an approach that addresses the heterophily issue in the spectral domain of graph anomaly detection by pruning interclass edges to emphasize and delineate the graph's high-frequency components.
13) *BWGNN:* It is a spectral GNN based on graph wavelet transform, which uses tailored spectral filters for node anomaly detection to address the "right-shift" phenomenon.

C. Metrics

According to existing anomaly detection benchmarks [56], [57], [58], we select F1-macro, area under the receiver operating characteristic curve (AUC), area under the precision–recall curve (AUPRC), and the recall score within top-k predictions (Rec@K) as performance metrics. We set k as the number of anomalies within the test set. F1-macro is the unweighted average of the F1-score of the two classes, ignoring the imbalanced ratio between normal and abnormal labels. AUC primarily focuses on overall performance and is not sensitive to top-k predictions, Rec@K only cares about top-k performance, and AUPRC strikes a balance between the two.

D. Implementation Details

For GWNN, GraphConsis, CAREGNN, and PC-GNN, we refer to [58] and implement fast algorithms using DGL (Deep Graph Library). For the rest of the baselines, we use the DGL official library or the code provided by the original article. For equal comparisons, we use the homo version of all baselines, which merges all edge types into a homograph. The heterogeneous graph version of our proposed MMGNN can be implemented relatively and is left for further research in the future. All baselines use the Adam optimizer to train 100 epochs with a learning rate of 0.01, and the remaining hyperparameters are set according to the values suggested by the corresponding papers. Our proposed MMGNN is implemented using PyTorch and DGL. All datasets are trained using the Adam optimizer with a learning rate of 0.01, the dimension h for hidden states is set to 64, and the multiplicand filter order C is set to 2. The epochs of Amazon are set to 90, the epochs of YelpChi and T-Social are set to 150, and the epochs of T-Finance are set to 200. We save the model with the best F1-macro in the validation set and finally calculate the values of all metrics for the best model in the test set. The training ratio is 40%, while the remaining data is split by 1:2 for validation and testing. For all methods, we report the mean and standard deviation of ten runs for Amazon and YelpChi, and for T-Finance and T-Social, we report 5 runs. We implement Amazon, YelpChi, and T-Finance on one NVIDIA RTX 2080Ti and implement T-Social on an Intel Xeon Silver 4214 CPU (T-Social is too large to fit into GPU).

E. Performance Comparison

Table III reports the performance of all compared methods. Overall, our proposed MMGNN consistently outperforms other baselines in all datasets. Specifically, compared with the best methods, on Amazon, YelpChi, T-Finance, T-Social, and F1-macro increased by 0.33%, 2.87%, 1.05%, 11.19%, AUC increased by 0.50%, 0.85%, 0.89%, 3.62%, AUPRC increased by 0.93%, 5.87%, 2.79%, 25.70%, and Rec@K increased by 0.63%, 4.89%, 2.31%, 20.29%, respectively. It can be roughly seen that as the scale of the graph increases, the performance improvement of the MMGNN is greater. Specifically, the GCN performs very poorly on Amazon and YelpChi because it is equivalent to a low-pass filter and tends to enhance low-frequency signals while suppressing high-frequency signals. The GCN has a better performance on T-Finance, mainly because the heterophily degree of T-Finance is smaller. The heterophily degrees of all datasets are shown in Table II. The signals in the graph with high heterophily degrees are biased toward the high frequency, while those with low heterophily degrees are biased toward the low frequency. Therefore, the low-pass GCN performs not badly on T-Finance. The better performance of the GCN on T-Social with a high heterophily degree may be due to the smaller average edge connections of each node. When aggregating neighborhood information, abnormal node features are not easily masked by neighborhood features. Compared to the GCN, ChebyNet performs better on almost all datasets because its learnable Chebyshev filter can serve as a bandpass filter. GAT, GIN, and GraphSAGE perform poorly on some datasets because they are designed based on dynamic learning of connection weights between

TABLE IV
COMPARISON OF TRAINING TIME BETWEEN STATE-OF-THE-ART METHODS AND THE MMGNN ON T-SOCIAL

Model	C	Epochs	h	F1-macro	AUC	Time (s)
AMNet	2	100	32	59.02±0.74	77.21±1.28	5,278
	2	100	64	59.02±1.02	77.02±1.67	9,190
GDN	-	100	10	-	-	>12days
GHRN	5	100	10	79.63±6.45	93.48±1.10	2,755
	5	100	64	78.74±12.97	93.36±3.94	10,220
BWGNN	5	100	10	83.54±5.27	94.52±2.10	2,871
	5	100	64	80.78±7.17	93.05±4.23	10,512
MMGNN	2	100	10	86.79±6.04	95.82±1.28	**1,532**
	2	150	10	90.85±3.44	96.25±1.02	2,386
	2	100	64	92.70±1.35	96.82±1.21	3,906
	2	150	64	**94.73±0.16**	**98.14±0.45**	5769

nodes or based on graph isomorphism tests and can only learn general feature representations. Although GWNN is designed based on graph wavelet transform, the heat kernel filters it uses are still essentially low-pass, resulting in poor performance.

GraphConsis, CAREGNN, and PC-GNN are methods designed for node anomaly detection, but they are designed from the perspectives of edge relationship learning. Their performance on some datasets is worse than that of general GNN methods. ChebyNet, which can be equivalent to a bandpass filter, performs even better than them. AMNet, GDN, and GHRN are state-of-the-art node anomaly detection methods that generally demonstrate better performance than general GNNs, but require more training time. The GDN needs to be trained on T-Social for more than 12 days (so the results are not shown), which is unacceptable.

The BWGNN is a state-of-the-art method based on bandpass filters. It uses the same-order bandpass filters and performs well on all datasets, which further demonstrates the superiority of low-pass filters in node anomaly detection. Our MMGNN achieves the best results on all datasets. For T-Social, the BWGNN has achieved good performance by using filters of $C = 5$ for training. However, our MMGNN decomposes the designed multiorder multiscale bandpass filters into the preprocessing part and the training part through the double-parallel structure, only needs to train second-order filters with $C = 2$, and the model training computational cost is lower (comparison experiments of training time on large-scale graphs are shown in Section VI-F). The MMGNN achieved over 10% performance improvement on F1-macro, which proves that it is not enough for the BWGNN to use same-order bandpass filters that treat low- and high-frequency signals equally. The signals in graphs have multiorder multiscale characteristics and are more biased toward the high frequency. Our proposed MMGNN equipped with multiorder multiscale bandpass filters can better learn the difference between abnormal and normal features, thereby significantly improving node anomaly detection performance.

F. Training Time on Large-Scale Graphs

For our proposed MMGNN, the complexity is $O(C|\mathcal{E}|)$, since filters are polynomial functions that can be computed recursively [32], where C is the training filter order and $|\mathcal{E}|$ is the number of edges. The complexity of the BWGNN and MMGNN is the same. To analyze the training speed of the MMGNN on large-scale graphs, we compare the training time of the AMNet, GDN, GHRN, BGWNN, and MMGNN on T-Social. The results are shown in Table IV. The C of the training part in our MMGNN is set to 2, and the learning rate of all compared methods is set to 0.01. They are implemented on an Xeon Silver 4214 CPU, and the mean and standard deviation of five runs are reported.

The results show that when epochs are 100 and h is 10, the MMGNN has the fastest training speed among all methods and is 87.40% faster than the BWGNN, achieving better performance. The performance improvement is due to the multiorder multiscale bandpass filters we designed, and the improvement in training speed is due to our construction of the double-parallel structure to decompose high-order filters into the preprocessing part and the training part, effectively reducing the computational cost during model training. When epochs are 100 and h is 64, the performance of the MMGNN continues to improve, while the compared methods have no obvious performance improvement. Even under the settings of epochs are 100 and h is 10, the estimated training time of the GDN exceeds 12 days, such training time is unacceptable for efficient training and real-time inference on large-scale graphs.

Overall, we believe that our proposed MMGNN can provide new inspiration for designing computational acceleration algorithms on large-scale graphs. In practical application scenarios, efficient training and real-time inference are crucial, especially for large-scale graphs. Previous methods such as GraphSAGE [36] often cut edge connections through fixed sampling to reduce computational cost. However, for a complete graph, the original edge connections cannot be arbitrarily changed, as slight perturbations to the graph may change the signal distribution of the entire graph, which is detrimental to node-level classification tasks such as node anomaly detection. The difference is that our proposed MMGNN performs a lightweight design of multiorder multiscale bandpass filters by constructing the double-parallel structure, without changing the topology of the original graph. Performing OR-gate mixup in the spectral space does not add additional views, further avoiding additional computational cost.

G. Ablation Study

To further analyze the effectiveness of the MMGNN, we perform an ablation study, and the results are shown in Table V, where MMF represents multiorder multiscale bandpass filters and OM represents OR-gate mixup. The results show that whether it is removing multiorder multiscale bandpass filters [i.e., removing the multiplier filters in the preprocessing part and retaining the multiplicand filters in the training part, which is equivalent to using the same-order bandpass filters in Fig. 2(b)] or removing OR-gate mixup, the performance of the MMGNN will decrease while using both multiorder multiscale bandpass filters and OR-gate mixup shows the best performance on all datasets. Multiorder multiscale bandpass filters and OR-gate mixup play an important role in the MMGNN. Multiorder multiscale bandpass filters are

TABLE V
ABLATION STUDY OF THE MMGNN ON ALL DATASETS

Dataset / Metric	MMF	OM	Amazon F1-macro	Amazon AUC	YelpChi F1-macro	YelpChi AUC	T-Finance F1-macro	T-Finance AUC	T-Social F1-macro	T-Social AUC
MMGNN	×	×	92.02±0.45	97.03±1.58	70.24±0.63	83.16±0.63	88.52±1.01	95.21±0.88	71.42±6.99	88.92±4.95
	✓	×	92.08±0.62	97.54±0.27	71.91±0.71	83.43±0.83	89.50±0.35	95.33±0.42	94.07±1.11	97.75±1.12
	×	✓	92.40±0.27	96.62±1.80	71.02±0.58	83.96±0.60	89.48±2.21	95.66±1.05	84.51±0.56	94.91±0.58
	✓	✓	**92.43±0.46**	**97.81±0.23**	**73.10±0.54**	**84.05±0.71**	**89.80±0.45**	**96.17±0.32**	**94.73±0.16**	**98.14±0.45**

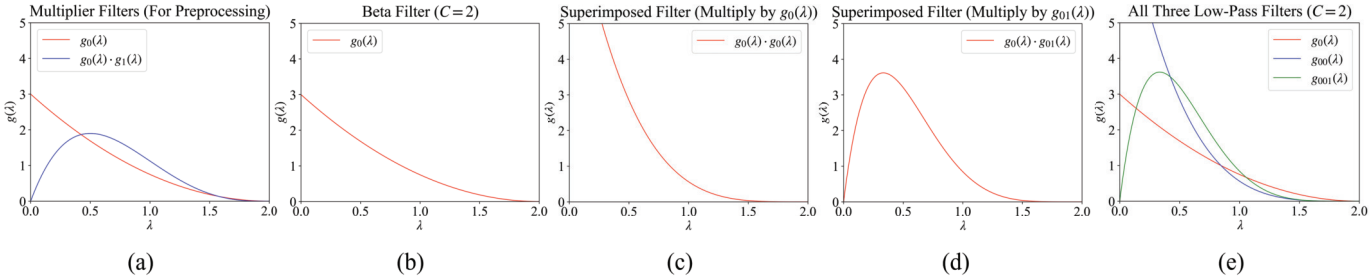

Fig. 6. When $C = 2$. (a) Two multiplier filters, which are second-order and fourth-order, respectively. (b) Vanilla Beta filter equivalent to second-order. (c) Equivalent fourth-order superimposed filter. (d) Equivalent sixth-order superimposed filter. (e) Multiorder low-pass filters.

TABLE VI
IMPACT OF FILTERS PROPERTIES ON NODE ANOMALY DETECTION

Dataset / Metric	Amazon F1-macro	Amazon AUC	YelpChi F1-macro	YelpChi AUC	T-Finance F1-macro	T-Finance AUC	T-Social F1-macro	T-Social AUC
MMGNN-MLow	90.49±0.44	95.91±0.77	67.13±1.02	79.47±1.48	87.66±2.14	92.94±6.22	73.84±8.72	90.18±9.45
MMGNN-1Low	91.56±0.31	95.79±0.55	66.18±0.39	78.61±0.61	88.43±1.34	95.18±0.99	70.15±1.26	90.48±1.09
MMGNN-1Mid	92.34±0.29	95.86±1.64	71.67±0.74	83.53±0.72	87.77±1.26	95.56±0.32	65.01±2.64	86.25±1.56
MMGNN-1High	92.17±0.13	96.55±1.32	71.19±0.51	83.58±0.57	87.26±1.13	95.30±0.30	73.74±0.82	87.11±0.92
BWGNN	92.10±0.27	97.15±1.13	70.23±0.81	83.20±0.80	88.75±0.75	95.28±0.22	83.54±5.27	94.52±2.10
MMGNN	**92.43±0.46**	**97.81±0.23**	**73.10±0.54**	**84.05±0.71**	**89.80±0.45**	**96.17±0.32**	**94.73±0.16**	**98.14±0.45**

specifically designed to mine multiscale high-frequency signals in the graph, thus improving the ability of the MMGNN to distinguish between abnormal and normal nodes. OR-gate mixup improves the ability of the MMGNN to detect difficult abnormal samples by performing feature mixup and label OR-gate mixup in the spectral space after parallel filtering. When the MMGNN removes multiorder multiscale bandpass filters, only the same-order bandpass filters are used for training. Although they can also mine high-frequency signals in the graph, the same-order bandpass filters treat low- and high-frequency signals at the same scale without differential mining. There is still a risk of remixing low- and high-frequency signals during model training, resulting in poor performance. The multiorder multiscale bandpass filters we designed not only overcome the shortcoming of low-pass filters that suppress high-frequency signals but also the shortcoming of same-order bandpass filters that treat low- and high-frequency signals equally, resulting in better performance.

Overall, although reasonable graph data augmentation methods can improve the performance of the model, the improvement is relatively small due to the quality of the node feature representations learned by the model. Designing specific spectral filters with good properties for node anomaly detection is crucial, as it directly affects the distinguishability of node feature representations learned by the model and can indirectly improve the effectiveness of other modules such as graph data augmentation and graph contrastive learning, thereby achieving the effect of mutual cooperation between model modules to improve performance.

H. Harmfulness of Multiorder Low-Pass Filters

In this part, we verify the benefits of multiorder bandpass filters for node anomaly detection, as well as the harmfulness of multiorder low-pass filters and the shortcomings of one single filter. The impact of filter properties on node anomaly detection is shown in Table VI. To construct multiorder low-pass filters, we remove the multiorder multiscale bandpass filters in the proposed MMGNN and then use the same multiplication method (in Section IV-A) to use the vanilla Beta filters ($C = 2$) $g_0(\lambda)$ and $g_1(\lambda)$ to construct two low-pass multiplier filters $g_0(\lambda)$ and $g_{01}(\lambda)$, where $g_{01}(\lambda) = g_0(\lambda) \cdot g_1(\lambda)$, as shown in Fig. 6(a). They are placed in the preprocessing part. For filters in the training part, we only use $g_0(\lambda)$ of the vanilla Beta filters ($C = 2$) as a multiplicand filter, as shown in Fig. 6(b). Two high-order low-pass filters

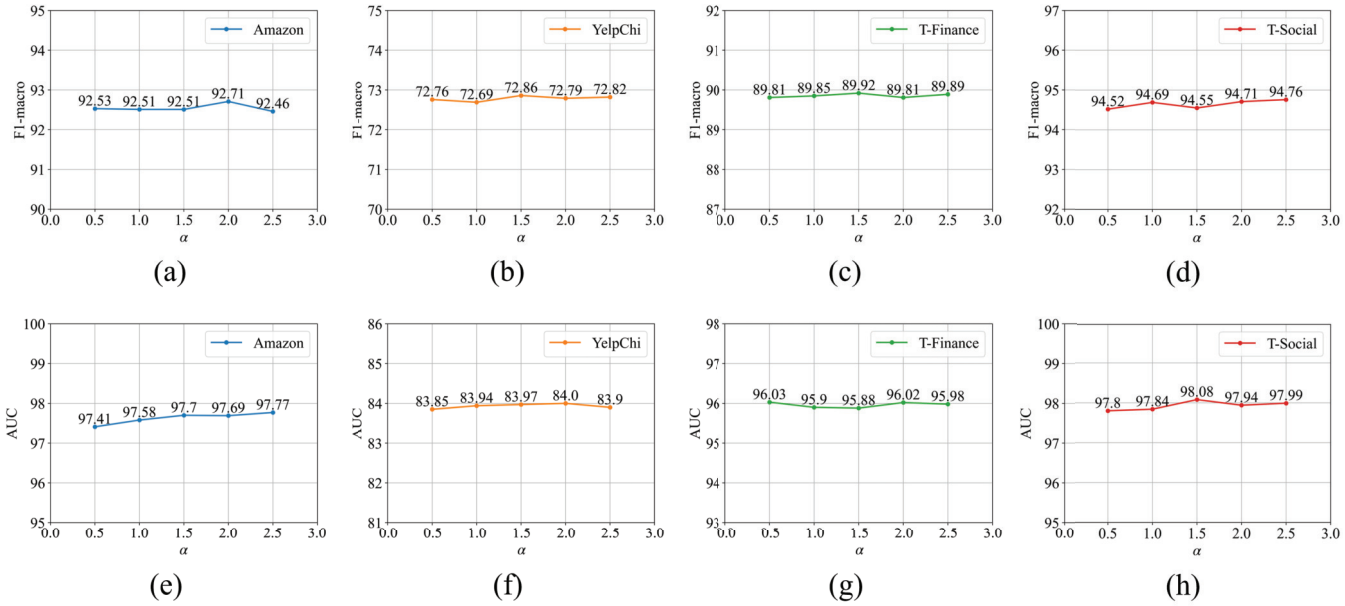

Fig. 7. Parameter sensitivity of α of the OR-gate mixup on all datasets. (a)–(d) F1-macro metric changes on Amazon, YelpChi, T-Finance, and T-Social datasets, respectively. (e)–(h) AUC metric changes on Amazon, YelpChi, T-Finance, and T-Social datasets, respectively.

$g_{00}(\lambda) = g_0(\lambda) \cdot g_0(\lambda)$ and $g_{001}(\lambda) = g_0(\lambda) \cdot g_{01}(\lambda)$ after superposition are shown in Fig. 6(c) and (d), respectively. Finally, the equivalent multiorder low-pass filters constructed through the double-parallel structure are shown in Fig. 6(e). The model using multiorder low-pass filters is named MMGNN-MLow. We also constructed MMGNN variants using only one-single filter [low-pass, bandpass, or high-pass, corresponding to $g_0(\lambda)$, $g_1(\lambda)$, or $g_2(\lambda)$ in Fig. 2(b)], named MMGNN-1Low, MMGNN-1Mid, and MMGNN-1High, respectively.

Table VI shows that the performance of MMGNN-MLow is poor and lower than the BWGNN using the same-order bandpass filters. MMGNN-1Low, MMGNN-1Mid, and MMGNN-1High only use one single filter and focus on partial frequency signals, resulting in poor performance, but better than MMGNN-Mlow. This proves that multiorder low-pass filters are harmful to node anomaly detection, as they repeatedly mine low-frequency signals without focusing on high-frequency signals, which weakens the ability of MMGNN-MLow to distinguish between abnormal and normal features. On the contrary, the multiorder multiscale bandpass filters in our MMGNN are beneficial because they focus more on high-frequency signals at multiscale, improving the distinguishability of node features. The performance difference between MMGNN-MLow and MMGNN on YelpChi and T-Social is greater than that on Amazon and T-Finance, which is related to the heterophily degree of different datasets shown in Table II. The signals in the graph with high heterophily degrees are biased toward the high frequency, while those with low heterophily degrees are biased toward the low frequency. The heterophily degrees of Amazon and T-Finance are small, with 0.0456 and 0.0292, respectively, while YelpChi and T-Social are large, reaching 0.2268 and 0.3761, respectively. The greater the heterophily degree of the graph, the more important the high-frequency signals in the graph, and the benefits of beyond low-pass filters are more obvious. Therefore, in graphs with high heterophily degrees, the performance of models using multiorder bandpass filters and multiorder low-pass filters shows significant differences, while with low heterophily degrees, the performance differences are smaller.

In summary, the results show that multiorder low-pass filters ignore high-frequency signals that weaken node anomaly detection performance. One-single filter only focuses on partial frequency signals, and its performance is also poor. The multiorder multiscale bandpass filters we designed can obtain better performance by fully mining high-frequency signals at multiscale.

I. Parameter Sensitivity

In this part, we investigate the parameter sensitivity of our proposed MMGNN. It is worth noting that the parameters of the MMGNN, such as learning rate, hidden states dimension h, and Beta distribution parameter α of OR-gate mixup are the same for all datasets and have not been carefully adjusted. Here, we mainly analyze the sensitivity of the parameter α, which affects the shape of the Beta distribution probability density curve and then affects the mixup of two samples. The results are shown in Fig. 7. Overall, on all datasets, there is little difference in the performance of the MMGNN when α takes different values. When α is small, the sample mixup will be more biased toward one of the samples (e.g., $\gamma = 0.9$ or $\gamma = 0.1$), and when α is large, the sample mixup will be more toward uniform mixup (e.g., $\gamma = 0.5$). Experimental results show that the MMGNN is not sensitive to parameter α. We set $\alpha = 2.0$ for all datasets so that the sample mixup tends to be uniform. It can fully construct new samples that are not similar to the two mixup samples, reducing excessive memory

of similar samples during model training, thus reducing the risk of overfitting.

VII. Conclusion

Graph node anomaly detection is a task with research significance and application value. In this article, we propose OR-Gate MMGNN from the spectral domain. Specifically, by utilizing the superposition and decomposition of polynomial spectral filters, we construct multiorder multiscale bandpass filters through the double-parallel structure with a preprocessing part and a training part, which can effectively mine high-frequency signals in the graph and reduce computational cost. Finally, we perform OR-gate mixup data augmentation in the spectral space to avoid adding additional views and improve model generalization. Experimental results on four real-world datasets demonstrate the effectiveness of the MMGNN against other state-of-the-art methods.

In the future, we will further research the following aspects.

1) Incorporating uncertainty quantification to measure the credibility of data augmentation samples, as some of the constructed new samples may be harmful to model training.
2) Due to the advantage of multiorder multiscale bandpass filters for distinguishable feature representation learning, features can be decoupled in spectral space and generative models can be used to generate more realistic abnormal samples, further alleviating the problem of rare abnormal patterns.

References

[1] J. Sipple, "Interpretable, multidimensional, multimodal anomaly detection with negative sampling for detection of device failure," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 9016–9025.

[2] D. Chen, R. Liu, Q. Hu, and S. X. Ding, "Interaction-aware graph neural networks for fault diagnosis of complex industrial processes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 6015–6028, Sep. 2023.

[3] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 2077–2085.

[4] Y. Yang, Y. Xu, Y. Sun, Y. Dong, F. Wu, and Y. Zhuang, "Mining fraudsters and fraudulent strategies in large-scale mobile social networks," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 1, pp. 169–179, Jan. 2021.

[5] X. Ma et al., "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12012–12038, Dec. 2023.

[6] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proc. SIAM Int. Conf. Data Mining*, 2019, pp. 594–602.

[7] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng, "Alleviating the inconsistency problem of applying graph neural network to fraud detection," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 1569–1572.

[8] T. Zhao, T. Jiang, N. Shah, and M. Jiang, "A synergistic approach for graph anomaly detection with pattern mining and feature learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2393–2405, Jun. 2022.

[9] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly detection on attributed networks via contrastive self-supervised learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2378–2392, Jun. 2022.

[10] K. Ding, K. Shu, X. Shan, J. Li, and H. Liu, "Cross-domain graph anomaly detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2406–2415, Jun. 2022.

[11] S. Bandyopadhyay, N. Lokesh, S. V. Vivek, and M. N. Murty, "Outlier resistant unsupervised deep architectures for attributed network embedding," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 25–33.

[12] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 315–324.

[13] Y. Liu et al., "Pick and choose: A GNN-based imbalanced learning approach for fraud detection," in *Proc. Web Conf.*, Apr. 2021, pp. 3168–3177.

[14] Y. Gao, X. Wang, X. He, Z. Liu, H. Feng, and Y. Zhang, "Alleviating structural distribution shift in graph anomaly detection," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, Feb. 2023, pp. 357–365.

[15] Y. Gao et al., "Addressing heterophily in graph anomaly detection: A perspective of graph spectrum," in *Proc. ACM Web Conf.*, 2023, pp. 1528–1538.

[16] J. Li, R. Zheng, H. Feng, M. Li, and X. Zhuang, "Permutation equivariant graph framelets for heterophilous graph learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 11634–11648, Sep. 2024.

[17] M. Li et al., "Guest editorial: Deep neural networks for graphs: Theory, models, algorithms, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4367–4372, Apr. 2024.

[18] X. Zheng et al., "Graph neural networks for graphs with heterophily: A survey," 2022, *arXiv:2202.07082*.

[19] J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking graph neural networks for anomaly detection," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 21076–21089.

[20] Z. Chai et al., "Can abnormality be detected by graph neural networks?," in *Proc. 29th Int. Joint Conf. Artif. Intell. (IJCAI)*, Vienna, Austria, 2022, pp. 23–29.

[21] F. Wu, A. H. Souza, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6861–6871.

[22] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine, "Analyzing the expressive power of graph neural networks in a spectral perspective," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–26.

[23] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 1–8.

[24] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 3438–3445.

[25] K. Zhou et al., "Dirichlet energy constrained learning for deep graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 21834–21846.

[26] J. Wang, J. Liang, J. Liang, and K. Yao, "GUIDE: Training deep graph neural networks via guided dropout over edges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4465–4477, Apr. 2024.

[27] T. Zhao et al., "Graph data augmentation for graph machine learning: A survey," 2022, *arXiv:2202.08871*.

[28] J. Duan et al., "Graph anomaly detection via multi-scale contrastive learning networks with augmented view," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 6, pp. 7459–7467.

[29] D. Jin et al., "CGMN: A contrastive graph matching network for self-supervised graph similarity learning," 2022, *arXiv:2205.15083*.

[30] Y. Luo et al., "Automated data augmentations for graph classification," 2022, *arXiv:2202.13248*.

[31] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.

[32] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.

[33] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[34] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.

[36] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[37] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
[38] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Jul. 2019.
[39] B. Li, Y. Hou, and W. Che, "Data augmentation approaches in natural language processing: A survey," *AI Open*, vol. 3, pp. 71–90, Jan. 2022.
[40] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 12, pp. 11015–11023.
[41] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," 2019, *arXiv:1907.10903*.
[42] W. Feng et al., "Graph random neural networks for semi-supervised learning on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 22092–22103.
[43] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 5812–5823.
[44] K. Kong et al., "Flag: Adversarial data augmentation for graph neural networks," 2020, *arXiv:2010.09891*.
[45] M. Sun, J. Xing, H. Wang, B. Chen, and J. Zhou, "MoCL: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 3585–3594.
[46] X. Han, Z. Jiang, N. Liu, and X. Hu, "G-mixup: Graph data augmentation for graph classification," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 8230–8248.
[47] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.
[48] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.
[49] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.
[50] H. M. de Oliveira and G. A. A. de Araujo, "Compactly supported one-cyclic wavelets derived from beta distributions," 2015, *arXiv:1502.02166*.
[51] C. Ying et al., "Do transformers really perform badly for graph representation?," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 28877–28888.
[52] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews," in *Proc. 22nd Int. Conf. World Wide Web*, May 2013, pp. 897–908.
[53] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 985–994.
[54] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2018, *arXiv:1810.00826*.
[55] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," 2019, *arXiv:1904.07785*.
[56] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "ADBench: Anomaly detection benchmark," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 32142–32159.
[57] K. Liu et al., "BOND: Benchmarking unsupervised outlier node detection on static attributed graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 27021–27035.
[58] J. Tang, F. Hua, Z. Gao, P. Zhao, and J. Li, "GADBench: Revisiting and benchmarking supervised graph anomaly detection," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, pp. 1–26.

Zekang Li received the B.S. degree in mechanical design, manufacturing and automation from Tianjin University, Tianjin, China, in 2022, and the M.S. degree in computer science and technology from the College of Intelligence and Computing, Tianjin University, in 2025.

His research interests include graph node anomaly detection and graph representation learning.

Ruonan Liu (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 2013, 2015, and 2019, respectively.

She was a Post-Doctoral Researcher with the School of Computer Science, Carnegie Mellon University, in 2019, and an Alexander von Humboldt Fellow with the University of Duisburg-Essen, Germany, from 2022 to 2024. She currently is an Associate Professor with the Department of Automation, Shanghai Jiao Tong University. Her research interests include machine learning, intelligent manufacturing, and vision-language navigation.

Dr. Liu has been awarded the 2021 Outstanding Paper Award by IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the Runnerup Paper Award in IJCAI-W 2024, the Best Paper Award Finalist in IEEE ARM 2024, the Best Paper Award in RCAE 2024 and selected in the Young Elite Scientist Sponsorship Program by CAST in 2022.

Dongyue Chen received the B.S. and M.S. degrees from Southwest Jiaotong University, Chengdu, China, in 2015 and 2018, respectively, and the Ph.D. degree from Tianjin University, Tianjin, China, in 2023.

She is currently a Post-Doctoral Researcher with the College of Intelligence and Computing, Tianjin University. Her research interests include artificial intelligence, incremental learning, trustworthy deep learning, and fault diagnosis of mechanical systems.

Qinghua Hu (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Harbin Institute of Technology, Harbin, China, in 1999, 2002, and 2008, respectively.

After that, he joined the Department of Computing, The Hong Kong Polytechnical University, Hong Kong, as a Post-Doctoral Fellow. He is currently a Chair Professor at the College of Intelligence and Computing, Tianjin University, Tianjin, China. He has published more than 300 peer-reviewed papers in IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and so on. His research interest is focused on uncertainty modeling, multimodality learning, incremental learning, and continual learning these years, funded by the National Natural Science Foundation of China and The National Key Research and Development Program of China.

Dr. Hu was a recipient of the best paper award of ICMLC 2015 and ICME 2021. He is an Associate Editor of IEEE TRANSACTIONS ON FUZZY SYSTEMS, *Acta Automatica Sinica*, and *Acta Electronica Sinica*.