Multi-scale Directed Graph Convolution Neural Network for Node Classification Task

Fengming Li[1], Dong Xu[1,2], Fangwei Liu[1], Yulong Meng[1,2](✉), and Xinyu Liu[3]

[1] College of Computer Science and Technology, Harbin Engineering University, Harbin, China
mengyulong@hrbeu.edu.cn

[2] Modeling and Emulation in E-Government National Engineering Laboratory, Harbin Engineering University, Harbin, China

[3] Jiangsu JARI Technology Group Co. Ltd, Lianyungang, China

Abstract. The existence of problems and objects in the real world which can be naturally modeled by complex graph structure has motivated researchers to combine deep learning techniques with graph theory. Despite the proposal of various spectral-based graph neural networks (GNNs), they still have shortcomings in dealing with directed graph-structured data and aggregating neighborhood information of nodes at larger scales. In this paper, we first improve the Lanczos algorithm by orthogonality checking method and Modified Gram-Schmidt orthogonalization technique. Then, we build a long-scale convolution filter based on the improved Lanczos algorithm and combine it with a short-scale filter based on Chebyshev polynomial truncation to construct a multi-scale directed graph convolution neural network (MSDGCNN) which can aggregate multi-scale neighborhood information of directed graph nodes in larger scales. We validate our improved Lanczos algorithm on the atom classification task of the QM8 quantum chemistry dataset. We also apply the MSDGCNN on various real-world directed graph datasets (including WebKB, Citeseer, Telegram and Cora-ML) for node classification task. The result shows that our improved Lanczos algorithm has much better stability, and the MSDGCNN outperforms other state-of-the-art GNNs on such task of real-world datasets.

Keywords: Graph neural network · Lanczos algorithm · Directed graph · Node classification

1 Introduction

Traditional deep learning techniques based on artificial neurons such as RNN [36] and CNN [22] have achieved great success in classification tasks [15] and recognition tasks [17] in euclidean space data such as images and texts. The success of the deep learning techniques is that they effectively leverage the statistical properties of Euclidean space data such as permutation invariance [46].

However, in the real world, graph-structured data is ubiquitous, and a large number of problems and objects need to be modeled based on complex graph structures. Unfortunately, graph-structured data do not belong to the Euclidean space, which means the data do not possess the aforementioned statistical properties. Thus, applying deep learning techniques on graph-structured data faces great challenges.

In recent years, many researchers have applied deep learning techniques to process graph-structured data in non-Euclidean space and have achieved success in various applications such as recommendation systems [18], program analysis [27], software mining [24], drug discovery [3] and anomaly detection [48]. As mentioned in [45], there are four kinds of graph neural networks currently available, including recurrent graph neural networks, graph convolution neural networks, graph autoencoders and spatio-temporal graph neural networks. Gated graph neural networks [27] aims to learn node embeddings through the construction of a recurrent neural network with gated units. Graph autoencoders [20] encodes nodes or entire graph into a latent vector through an encoder and then decodes the latent vector by a decoder for node-level or graph-level learning tasks. Spatio-temporal graph neural networks [26] aims to learn the hidden patterns of a graph from spatio-temporal graphs. Graph convolution neural networks generalizes the convolution operation from grid data to graph data, aggregating features of nodes and their neighbors through the convolution operation to generate node embeddings. More details about graph neural networks can be found in [45,51].

In spectral-based graph neural networks, there exist two main issues. Firstly, spatial-based graph neural networks such as GAT [42] can naturally be implemented for directed graph data by symmetrizing the adjacency matrix and treating it as an undirected graph. However, spectral-based graph neural networks that involves Fourier transformation require the adjacency matrix of a graph to be symmetric to ensure the matrix's eigenvalues are real. Since the adjacency matrix of a directed graph often does not have symmetry, spectral-based graph neural networks can not be directly applied to directed graphs. Secondly, how to obtain the multi-scale information of graphs by convolution operations has not been effectively explored. Due to the involvement of large sparse matrix eigendecomposition and multiplication in spectral-based graph convolution theory, performing multi-scale convolution operations is itself a computational challenge.

Recently, many researchers ([6,12,16,21], etc.) have used the Magnetic Laplacian Matrix [29] from the field of particle physics to model directed graph data, which involves introducing a phase parameter q to control the strength of the directional flow in a graph and converting the adjacency matrix of the graph to a Magnetic Laplacian Matrix. This approach transforms the real-valued asymmetric adjacency matrix of a directed graph into a complex Hermitian matrix, which partially overcomes the difficulty of processing directed graphs directly with spectral-based graph neural networks. Compared with DGCN [41] and DiagGraph [40], which are specifically used for directed graphs, the method reduces the complexity of neural networks and makes spectral-based directed graph convolution neural networks can be easily trained. However, in the research aforementioned, the filter used for the convolution operation faces extremely high computational overhead in the problem of expanding the convolution scale.

Most of the current spectral-based graph neural networks use the K_{th} order truncation of Chebyshev polynomials to obtain filtering operators, which avoids directly performing eigendecomposition on the Laplacian matrix of a graph and reduces computational complexity to some extent. However, there is still a high computational cost when using the method to aggregate larger-scale information about neighbors of a node. The Lanczos algorithm [23], which is commonly used in quantum systems, has been applied to compute the eigendecomposition of large-scale sparse graphs [39]. In their experiments, the algorithm exhibits lower error and time cost than the Chebyshev polynomial truncation method. Subsequently, AdaLanczosNet [28] was proposed based on the Lanczos algorithm. Experimental results showed that the spectral-based graph neural network with the algorithm has lower training and testing errors as well as better computational efficiency than other spectral-based GNNs with the Chebyshev polynomial truncation method. However, although the aforementioned studies considered the problem of loss of orthogonality of *Krylov* subspace vectors due to rounding errors during the iterative process of the Lanczos algorithm, their methods did not effectively correct the orthogonal vectors. Even worse, they lowered the efficiency of the Lanczos algorithm since executing re-orthogonalization at each iteration. In this paper, we aims to addressed these problems and the main contributions are listed here:

* We improve Lanczos algorithm using Modify Gram-Schmdit orthogonalization technique and the orthogonality checking method.
* We model directed graphs using the Magnetic Laplacian matrix and build a multi-scale graph convolution neural network(MSDGCNN) for node classifi cation tasks combining ChebyShev polynomials and our improved Lanczos algorithm.
* We evaluate the performance of our improved Lanczos algorithm on the QM8 quantum chemistry dataset and validate the effectiveness of MSDGCNN on the WebKB, Telegram, CiteSeer and Cora-ML datasets.

2 Related Work

2.1 Spectral Graph Convolution Theory

Given an undirected graph $G = (V, E, X)$, where V denotes the set of nodes of G, $|V| = N$; E denotes the set of edges, $E = \{e_{ij} |\ if\ \exists\ v_i \leftrightarrow v_j;\ i, j = 1, ..., N\} \subseteq V \times V$; W is the set of weights of the edges, $W = \{w_{ij} |\ if\ e_{ij} \in E\}$ and $W \in \mathcal{R}^{N\times N}$, if G is an unweighted graph then $w_{ij} \in \{0, 1\}$; X denotes the set of node features, $X = \{x_i |\ i = 1, ..., N\}$; A denotes the adjacency matrix of G and we have:

$$A_{ij} = \begin{cases} w_{ij} & \text{if } \exists\ v_i \leftrightarrow v_j \in E \\ 0 & \text{other cases} \end{cases}$$

According to the spectral graph theory [4], define the unnormalized Laplacian matrix of G: $L_U = D - A \in \mathcal{R}^{N\times N}$, where $D \in \mathcal{R}^{N\times N}$ is the degree diagonal matrix of G and $D_{ii} = \sum_{j=1}^{N} A_{ij}$. Then the normalized Laplacian matrix of G is defined as $L_N = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \in \mathcal{R}^{N\times N}$ where I_N is the identity matrix of order N. Clearly, $L_N = L_N^T$ (L_N^T denotes the transpose of L_N), thus, L_N has non-negative real eigenvalues $\lambda_1, ..., \lambda_N$ and the orthogonal eigenvectors $u_1, ..., u_N$ corresponding to the eigenvalues. Let $U = [u_1|...|u_N] \in \mathcal{R}^{N\times N}, \Lambda = diag(\lambda_1, ..., \lambda_N)$, then we have $L_N = U\Lambda U^T$, where $U = [u_1|...|u_N]$ is called the Fourier orthogonal basis. According to [37], the graph Fourier transform of the feature signal $x_i \in \mathcal{R}^N$ of node $v_i (i = 1, ..., N)$ is $\hat{x}_i = U^T x_i \in \mathcal{R}^N$ and the inverse transform of the graph Fourier transform is $x_i = U\hat{x}_i$. Thus, the convolution operation on the feature signal of node v_i of the graph G in spectral domain is defined as

$$\begin{aligned} y * x_i &= U\left((U^T y) \odot (U^T x_i)\right) \\ &= U(\hat{y} \odot (U^T x_i)) \\ &= U diag(\hat{y}) U^T x_i \end{aligned} \tag{1}$$

where $y \in \mathcal{R}^N$, $\hat{y}$ is called the Fourier coefficient and $\odot$ denotes Hadamard product. Let $g_\theta(\Lambda) = diag(\hat{y})$, according to Eq. (1) the feature signal x_i of node v_i is filtered by $g_\theta(\Lambda) = diag(\hat{y})$. Also, in practice, $g_\theta(\Lambda)$ can be regarded as a function of the diagonal matrix Λ of eigenvalues of the Laplacian matrix L_N of G [43].

2.2 Chebyshev Polynomials Approximate

Due to the involvement of large sparse matrix multiplication and eigendecomposition, computing $g_\theta(\Lambda)$ is computationally expensive. [14] approximated the Fourier filter $g_\theta(\Lambda)$ by truncating the Chebyshev polynomials at order K. Then, [5] used this method to construct ChebNet, which avoids the huge cost of eigendecomposition of the Laplacian matrix of graph G and improves stability in the face of perturbations [25].

Let $\hat{\Lambda} = \frac{1}{\lambda_{max}}\Lambda - I_N$ be the normalized Laplacian eigen matrix, then we have the filter which is based on truncating the Chebyshev polynomials at order K:

$$g_\theta(\hat{\Lambda}) = \sum_{m=0}^{K} \theta_m T_m(\hat{\Lambda}) \tag{2}$$

where the K_{th} order Chebyshev polynomial T_K is recursively defined as $T_0(\hat{\Lambda}) = I_N$, $T_1(\hat{\Lambda}) = \hat{\Lambda}$, $T_K(\hat{\Lambda}) = 2\hat{\Lambda}T_{K-1}(\hat{\Lambda}) + T_{K-2}(\hat{\Lambda})$. Then the feature h_i obtained by the convolution operation on the feature x_i of the node v_i of graph G is

$$\begin{aligned} h_i &= \sum_{m=0}^{K} U\theta_m T_m(\hat{\Lambda}) U^T x_i \\ &= \sum_{m=0}^{K} \theta_m T_m(\hat{L}) x_i \end{aligned} \tag{3}$$

3 Method

In this section, we will provide a detailed description of the construction details of our proposed MSDGCNN which can aggregate larger scale information of a node of G with lower computational overhead by using our improved Lanczos algorithm. Firstly, we magnetize the Laplacian Matrix of the graph G to obtain the Magnetic Laplacian Matrix. Then, we utilize our improved Lanczos algorithm to obtain the low-rank approximation of the eigenvalues and eigenvectors of the Magnetic Laplacian Matrix of graph G. Finally, through the MSDGCNN, we perform convolution operations using short Scale Filters and song Scale Filters, and output the node classification results.

3.1 Problem Formulation

Given a directed graph $G = (V, E, X)$, where V denotes the set of nodes of G, $|V| = N$; E denotes the set of edges, $E = \{e_{ij}|\ if\ \exists\ v_i \rightarrow v_j;\ i, j = 1, ..., N\} \subseteq V \times V$; W denotes the set of weights of edges, $W = \{w_{ij}|\ if\ e_{ij} \in E\} \subseteq \mathcal{R}^{N\times N}$, and if G is unweighted then $w_{ij} \in \{0, 1\}$. Let A be the adjacency matrix of G, if there exists $v_i \rightarrow v_j,\ i.e.\ \exists\ e_{ij} \in E$ then $A_{ij} = w_{ij}$, otherwise $A_{ij} = 0$. Let X denotes the set of features of nodes, $X = \{x_i|\ i = 1, ..., N\}$.

3.2 Constructing the Magnetic Laplacian Matrix of *G*

To improve the stability of the training process, let $\tilde{A} = \frac{1}{2}(A^T + A) + I_N$; $\tilde{D}$ is the degree diagonal matrix of $\tilde{A}$, where $\tilde{D}_{ii} = \sum_{j=1}^{N} \tilde{A}_{ij}$ and $\tilde{D}_{ij} = 0\ (i \neq j)$. We have the magnetization process of graph G by follows:

$$\Theta^{(q)} = 2\pi q(A - A^T),\ q \geq 0 \tag{4}$$

$$\begin{aligned} H^{(q)} &= \tilde{A} \odot exp\left(i\Theta^{(q)}\right) \\ &= \tilde{A} \odot \left(\cos(\Theta^{(q)}) + i\sin(\Theta^{(q)})\right) \end{aligned} \tag{5}$$

where $\Theta^{(q)}$ is a skew-symmetric matrix; $H^{(q)}$ is an Hermitian matrix with complex elements; $sin(.)$ and $cos(.)$ are element-wise functions; The phase matrix $\Theta^{(q)}$ is used to capture the directional information of the directed edges of G. For example, $\Theta^{(0)} = 0$ when $q = 0$, $H^{(0)} = \tilde{A}$ degenerates to the adjacency matrix of the undirected graph. When $q \neq 0$, $\Theta^{(q)}$ will be able to capture the directional information of G. If $q = 0.25$ and $\exists\ e_{ij} \in E$ then we have:

$$H^{(0.25)}_{(i,j)} = \pm\frac{iw_{ij}}{2} = -H^{(0.25)}_{(i,j)}$$

In this case, an edge $v_i \rightarrow v_j$ will be regarded as an edge opposite to $v_j \rightarrow v_i$. Thus the Magnetic Laplacian Matrix of G can be defined as

$$L_U^{(q)} = \tilde{D} - H^{(q)} = \tilde{D} - \tilde{A} \odot exp\left(i\Theta^{(q)}\right) \tag{6}$$

$$L_N^{(q)} = I - \left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\right) \odot exp\left(i\Theta^{(q)}\right) \tag{7}$$

where $L_U^{(q)}$ denotes the unnormalized Magnetic Laplacian Matrix of G and $L_N^{(q)}$ denotes the normalized Magnetic Laplacian Matrix of G. From Eqs. (6) and (7), we have two theorems as follows:

Theorem 1. *Both $L_U^{(q)}$ and $L_N^{(q)}$ have non-negative real eigenvalues.*

Proof. The proof of the Theorem 1 can be found in [7].

Theorem 2. *For $\forall\ q \geqslant 0$, the eigenvalues of the normalized Magnetic Laplacian Matrix $L_N^{(q)}$ taking the interval $[0, 2]$.*

Proof. The proof of the Theorem 2 can be found in [50].

The two theorems above, ensure that we can use the K_{th} order Chebyshev polynomial to build short-scale filter.

We follow a similar setup to [50]: let $K = 1$, set $\lambda_{max} \approx 2$ and $\theta_1 = -\theta_0$. Let $\tilde{L} = \frac{2}{\lambda_{max}} L_N^{(q)} - I_N$, then the output of the node feature x_i after filtering is:

$$\begin{aligned} h_i &= \sum_{m=0}^{1} \theta_m T_m(\tilde{L}) x_i \\ &= \theta_0 (I_N + (\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}) \odot exp(i\Theta^{(q)})) x_i \end{aligned} \tag{8}$$

3.3 Imporved Lanczos Algorithm

Lanczos algorithm is a kind of *Krylov* subspace iteration method, which aims to find a set of standard orthogonal bases $Q_k = [q_1|...|q_k] \in C^{N\times k}$ (where Q_k^H denotes the conjugate transpose of Q_k and $Q_k^H Q_k = I_k$) and a tridiagonal matrix $T_k \in R^{k\times k}$, thus approximating the eigenvalues and eigenvectors of the Hermitian matrix A. Given the Hermitian matrix $A \in C^{N\times N}$, if $Q_k^H A Q_k = T_k$ is a tridiagonal matrix and $Q_k^H Q_k = I_k$ then:

$$\begin{aligned} \mathcal{K}_k(A, q_1, k) &= Q_k Q_k^H K_k(A, q_1, k) \\ &= Q_k [e_1 | T_k e_1 | ... | T_k^{k-1} e_1] \end{aligned}$$

is the QR decomposition of $\mathcal{K}_k(A, q_1, k)$. Where e_1 and q_1 are the first columns of the identity matrix I_N and Q_k, respectively. Using the orthogonal matrix whose first column is q_1 to tridiagonalize A can effectively generate the columns of Q_k. Define the tridiagonal matrix $T_k \in \mathcal{R}^{k\times k}$:

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{pmatrix}$$

By transforming $Q_k^H A Q_k = T_k$ to $AQ_k = Q_k T_k$, for $i = 1, ..., k-1$, we have:

$$Aq_i = \beta_{i-1} q_{i-1} + \alpha_i q_i + \beta_i q_{i+1}$$

where $\beta_0 q_0 \equiv 0$, according to the orthogonality of vector q there is: $\alpha_i = q_i^H A q_i$. By shifting the term, the vector r_i is defined as

$$r_i = (A - \alpha_i I_N) q_i - \beta_{i-1} q_{i-1}$$

Then we have $q_{i+1} = \frac{r_i}{\beta_i}$, where $\beta_i = \pm \|r_i\|_2$. If $r_i = 0$ then the iteration stops, at which point useful information about the *Krylov* invariant subspace has been obtained.

Although the Lanczos algorithm is more computationally efficient than the power method, rounding errors during its iteration process can cause the orthonormal basis $q_1, ..., q_k$ of the *Krylov* subspace to lose orthogonality [31]. To overcome the problem above, it is necessary to perform reorthogonalization of the vectors $q_1, ..., q_i (i = 1, ..., k)$ at each iteration of the Lanczos algorithm. However, this approach significantly reduces the computational efficiency of the algorithm. Fortunately, numerical experiments and theoretical analyses indicate that the results of the Lanczos algorithm still have high accuracy when the loss of orthogonality of the basis vectors is within an acceptable scope ([11,32,38], etc.). Therefore, we turn to check the orthogonality of the orthonormal basis generated by the Lanczos algorithm at each iteration, and use the Modify Gram-Schmdit orthogonalization technique ([30] suggest that Modify Gram-Schmdit is a more effective method.) to reorthogonalize the basis vectors when the loss of orthogonality exceeds a certain threshold.

Our improved Lanczos algorithm is shown in Algorithm 1. In Step 6, we perform an orthogonality check on the *Krylov* subspace basis generated by the iteration, where ϵ_M is machine precision and the $max()$ is an element-wise function which finds the maximum value of matrix. In Step 13, we perform a Schur decomposition on the three-term recurrence relation matrix T_k, i.e. $V_k^H T_k V_k = diag(r_1, ..., r_k)$, to obtain the diagonal matrix $R_k \in \mathcal{R}^{k \times k}$ which is consisted of Ritz values. In Step 14, we construct the matrix $Y_k = [y_1|...|y_k]$, where y_i and r_i form a Ritz pair. Finally, the matrix Y_k formed by Ritz vectors and diagonal matrix R_k obtained from k steps of the algorithm can be used to approximate the Hermitian matrix A, i.e. $A \approx Y_k R_k Y_k^H$. The upper bound on the approximation error of the Lanczos algorithm is given by Theorem 3:

Theorem 3. *Let $U \Lambda U^H$ be the Schur decomposition of a Hermitian matrix $S \in \mathcal{C}^{N \times N}$, where $\Lambda = diag(\lambda_1, ..., \lambda_N)$, $\lambda_1 \geq ... \geq \lambda_N$; $U = [u_1, ..., u_N]$. Let $\mathcal{U}_j = span\{u_1, ..., u_j\}$, the initial vector for K-step Lanczos algorithm is q, and it outputs an orthogonal matrix $Q \in \mathcal{C}^{N \times K}$ and a tridiagonal matrix $T \in \mathcal{R}^{K \times K}$. For any $j (1 < j < K < N)$, we have:*

$$\left\|S - QTQ^H\right\|_F^2 \leq \sum_{i=1}^{j} \lambda_i^2 \left(\frac{\sin(q, \mathcal{U}_i) \prod_{k=1}^{j-1} (\lambda_k - \lambda_N) / (\lambda_k - \lambda_j)}{\cos(q, u_i) T_{K-i}(1 + 2\gamma_i)} \right)^2 + \sum_{i=j+1}^{N} \lambda_i^2$$

Algorithm 1: Improved Lanczos Algorithm

```
Input: $A \in C^{N\times N}, A^H = A; q_1$ is a unit − 2 norm vector $\in$
       $C^N$; LanczosStep $\in N_+$, LanczosStep $\ll N$.
Result: Ritz vector $y_1, ..y_k$ and Ritz value $r_1, .., r_k$.
Data: set $k = LanczosStep, i = 0, \beta_0 = 1$
1  while $i \leq k$ and $\beta_i \neq 0$ do
2      $q_{i+1} \leftarrow \frac{r_i}{\beta_i}$;
3      $i \leftarrow i + 1$;
4      $\alpha_i \leftarrow q_i^H A q_i$;
5      $r_i \leftarrow (A - \alpha_i I)q_i - \beta_{i-1} q_{i-1}$;
6      if $\max(|I_i - Q_i^H Q_i|) > \sqrt{\epsilon_M}$ then
7          do Modify Gram-Schmdit Process
8      end
9      $\beta_i \leftarrow \|r_i\|_2$;
10 end
11 $Q_k = [q_1, ..., q_k]$;
12 $T_k \leftarrow Tridiagonal[\alpha_1, ..., \alpha_k]$ and $[\beta_1, ..., \beta_k - 1]$;
13 $R_k, V_k \leftarrow$ decompose $T_k$;
14 $Y_k \leftarrow Q_k V_k$;
15 return $Y_k = [y_1|...|y_k], R_k = diag(r_1, ..., r_k)$;
```

where $T_{K-i}(x)$ is a Chebyshev polynomial of order $K - i$, $\gamma_i = \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_N}$. The proof details can be found in [10].

3.4 MSDGCNN Architecture

In this section, we propose a multi-scale directed graph convolution neural network (MSDGCNN) that combines Chebyshev polynomials and the improved Lanczos algorithm. The MSDGCNN uses K-order truncation of Chebyshev polynomials to approximate low-order filters of G during short-scale convolution operations and uses our improved Lanczos algorithm to obtain high-order filters of G during long-scale convolution operations. The network architecture is shown in Fig. 1.

Short-Scale Filter. Let $X^{(0)}_{N\times F_0} = [x^{(0)}_1|...|x^{(0)}_N]$, $x_i^{(0)\in\mathcal{R}^{F_0}}$ is the matrix of initial node features, where the feature dimensions of each node is F_0. Let $Z^{(0)} = X^{(0)}$, where $Z^{(l)}$ is the output features of the l_{th} convolution layer. The output of the short-scale convolution filter in the $l_{th}(l = 1, ..., L)$ layer can be represented as $h^{(l)}_{short}$. According to Eq. (8), we have:

$$h^{(l)}_{short} = \theta_0(I_N + (\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}) \odot exp(i\Theta^{(q)}))Z^{(l-1)} \tag{9}$$

Long-Scale Filter. To obtain the structure information of G in larger scale, we use the low-rank approximation of the Magnetic Laplacian Matrix of G obtained

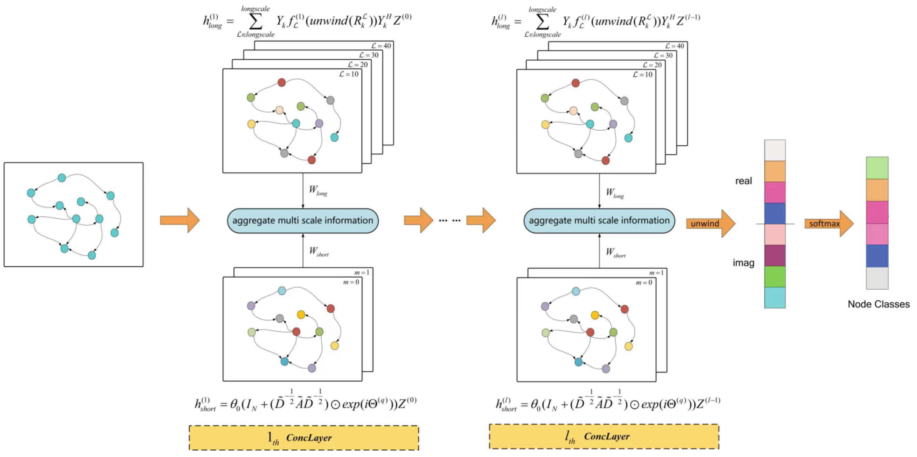
$h_{long}^{(1)} = \sum_{\mathcal{L}\in longscale}^{longscale} Y_k f_{\mathcal{L}}^{(1)}(unwind(R_k^{\mathcal{L}}))Y_k^H Z^{(0)}$
$h_{long}^{(l)} = \sum_{\mathcal{L}\in longscale}^{longscale} Y_k f_{\mathcal{L}}^{(l)}(unwind(R_k^{\mathcal{L}}))Y_k^H Z^{(l-1)}$
$\mathcal{L}=40$
$\mathcal{L}=30$
$\mathcal{L}=20$
$\mathcal{L}=10$
W_{long}
aggregate multi scale information
W_{short}
$m=1$
$m=0$
$h_{short}^{(1)} = \theta_0(I_N + (\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})\odot exp(i\Theta^{(q)}))Z^{(0)}$
$h_{short}^{(l)} = \theta_0(I_N + (\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})\odot exp(i\Theta^{(q)}))Z^{(l-1)}$
1_{th} ConcLayer
l_{th} ConcLayer
unwind
real
imag
softmax
Node Classes

Fig. 1. Network architecture: for example we set $longscale = [10, 20, 30, 40]$ and use short scale filter given by Eq. (8) with K=1.

by the Algorithm 1, i.e. $L_N^{(q)} \approx Y_k R_k Y_k^H$, then the long-scale convolution filter output $h_{long}^{(l)}$ of the l_{th} layer is:

$$h_{long}^{(l)} = \sum_{\mathcal{L}\in longscale}^{longscale} Y_k f_{\mathcal{L}}^{(l)}(unwind(R_k^{\mathcal{L}}))Y_k^H Z^{(l-1)} \tag{10}$$

where the function $f_{\mathcal{L}}$ is a shallow neural network formed by MLP, the function $unwind()$ unwinds the diagonal matrix R_k into a one-dimensional vector, and $longscale$ is a long-scale list. For example, we set $longscale = [10, 20, 30, 40]$, and the maximum value of the long-scale list $longscale$ should not exceed the number of nodes i.e. N. We set up a shallow neural network for each scale $\mathcal{L}$ in our long-scale list.

Aggregate Scale. According to Eqs. (9) and (10), we can finally construct the l_{th} convolution layer.

$$Z^{(l)} = \sigma\left(W_{short}^{(l)} h_{short}^{(l)} + W_{long}^{(l)} h_{long}^{(l)} + B^{(l)}\right) \tag{11}$$

where $W_{short}^{(l)}$ and $W_{long}^{(l)}$ are the weight parameters of the short-scale and the long-scale convolution filters, respectively. $B^{(l)}$ is the bias parameter matrix and $\sigma()$ is the complex ReLU activation function [50]. After passing through L convolution layers, the output $Z_{N\times F_L}^{(L)}$ is obtained, then we unwind $Z^{(L)}$ into real and imaginary parts, and use the softmax function to output the result of node classification:

$$class = softmax\left(unwind(Z^{(L)})W_{unwind}\right)$$

where the $W_{unwind} \in \mathcal{R}^{2F_L \times Numclass}$ is the weight parameter matrix of the real and imaginary parts after unwinding $Z^{(L)}$. $Numclass$ is the number of node classes in the final output.

4 Experiment

In this section, we illustrate our experiments. We validate the effectiveness of our improved Lanczos algorithm and the MSDGCNN we built with different datasets, respectively. In this paper, we choose node classification task to evaluate MSDGCNN, and our method can be easily applied on link prediction task by reversing the graph (In the reversed graph, edges and nodes correspond to nodes and edges, respectively, in the original graph.) and clustering task by changing loss functions.

4.1 Datasets

The QM8 dataset is derived from summarizing chemical structures and quantum chemistry computation output values of molecules [34]. The dataset contains 21,786 molecules with 6 different types of chemical bonds and 70 different types of atoms. In our experiment, each molecule is treated as a graph, where each type of chemical bond represents a type of connection (or edge). Each atom is treated as a node in the graph. Since there may be multiple types of chemical bonds between two different atoms in a molecule, the graph for each molecule is a multi-graph. We evaluated our improved Lanczos algorithm for the node classification task on the QM8 quantum chemistry dataset.

Table 1. Real world dataset details.

Dataset	Nodes	Edges	Class	Features
Telegram	245	8912	4	1
Citeseer	3312	4715	6	3703
Cora-ML	2995	8416	7	2879
Wiscosin	251	499	5	1703
Cornell	183	295	5	1703
Texas	183	309	5	1703

The Telegram dataset [2] is a network of pairwise interactions between 245 telegram channels, comprising 8,912 relationships between channels. The nodes in the telegram channel network are divided into four classes according to the information provided in [2]. The Wisconsin, Cornell and Texas datasets are taken from the WebKB dataset which model the relationships between different university websites [33]. In our experiment, the nodes in these datasets are labeled to cover identity information for five categories of personnel. The Cora-ML and CiteSeer datasets, provided by [1], are citation networks for scientific articles. The articles in these datasets belong to seven and five different scientific fields, respectively. We perform node classification tasks on these datasets to evaluate the performance of our multi-scale directed graph convolution neural network. The detailed information about the datasets is presented in Table 1.

4.2 Implementation Details

We implement our improved Lanczos algorithm using the Python language and build our multi-scale directed graph convolution neural network using the Pytorch1.13 deep learning framework. We use the Ubuntu20.04 operating system as our system environment. We are conducting experiments using a computer equipped with an Intel i5-12600K with 64GB DDR4 memory and Nvidia RTX 3060 with 12GB memory.

Evaluation of Improved Lanczos Algorithm. We use the method provided by DeepChem [35] to split the QM8 quantum chemistry dataset. Also, we use the approach provided by [9,44], which means that we use MSE as loss function to update the parameters in training and use MAE function to evaluate the training performance. We apply the improved Lanczos algorithm to AdaLanczosNet [28] by replacing the original Lanczos algorithm and compare it with the original AdaLanczosNet which is the baseline in the experiment. To ensure the reliability of the experiment, our hyperparameter settings follow the same long-scale list $longscale = [5, 7, 10, 20, 30]$ and short-scale list $shortscale = [1, 2, 3]$ as in [28]. We set the threshold for rounding error of the improved Lanczos algorithm to $\epsilon_M = 1.192 \times 10^{-7}$ which is the machine precision of 32-bit floating-point numbers. In the experiment, since the number of atoms contained in each molecular graph is different and the average number of atoms per molecule is around 16, we set *LanczosStep* to 10 and train both the AdaLanczosNet using the original Lanczos algorithm and the AdaLanczosNet using the improved Lanczos algorithm for 200 epochs.

Evaluation of MSDGCNN. We use the splliting method, as in many literatures ([33,49], etc.) mentioned, to split the WebKB and Telegram datasets, i.e., 60%, 20%, 20% for training, validation and testing sets. During training process, we also use the data of testing set without the node labels. The spliting method of Cora-ML and CiteSeer we used is similar to [40]. We randomly split each dataset into 10 subsets. When applying our improved Lanczos algorithm, we set the size of *LanczosStep* to half of the number of nodes in each dataset (i.e. $LanczosStep \leq \frac{N}{2}$), therefore, the number of iterations of our improved Lanczos algorithm will not exceed $\frac{N}{2}$. We set different long-scale lists for different datasets with different node numbers (e.g. for the node classification task on the Telegram dataset, we set $longscale = [15, 25, 35, 45]$.), and adjust the dimensions of the hidden layers of the function $f_{\mathcal{L}}$ from Eq. (10) according to the node feature dimensions and number of nodes in the dataset. To speed up the training process of our network, we pre-train the short-scale convolution filters and use the trained parameters as the initialization parameters for the model for all node classification tasks. We use phase parameters q as hyperparameters in experiment to set the appropriate direction strength for each dataset. We choose several popular graph neural networks as our baselines, including MagNet [50], APPNP [8], GraphSAGE [13], GIN [47], GAT [42], ChebNet [5], GCN [19], as well as DGCN [41] and DiGraph [40]. According to [45], These graph neural networks, can be divided into spatial-based as well as spectral-based GNNs and specialized directed graph GNNs. For all baseline models, we choosed best hyperparameters from literature and used the best result as baselines. We set the number of convolution layers to 2 for all models and set the order of the Chebyshev polynomial to 1. In our experiments, we train all models for 4000 epochs using the Adam optimizer.

4.3 Experimental Results and Analysis

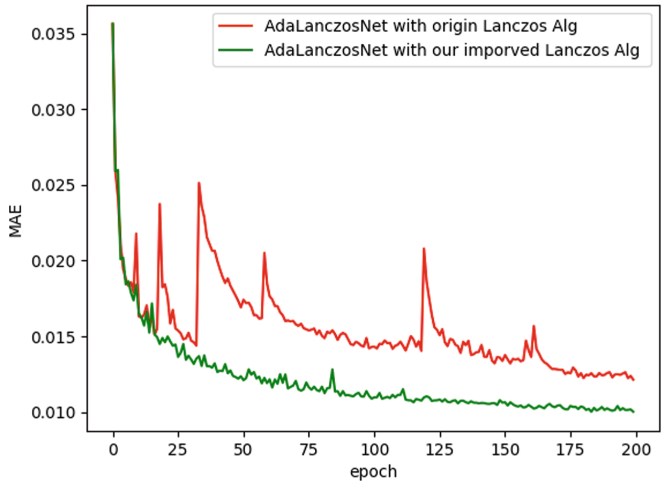

Fig. 2. Validation MAE on QM8 dataset during the training process

Results and Analysis on Improved Lanczos Algorithm. The validation MAE during the experiment on the QM8 dataset is shown in Fig. 2. It is evident that AdaLanczosNet with our improved Lanczos algorithm has better stability during the training process compared to AdaLanczosNet with the original Lanczos algorithm. The test MAE is shown in Table 2, obliviously, our improved algorithm achieved lower classification error by correctly modifying the orthogonal basis. Through experiments, we can clearly observe that the loss of orthogonality

Table 2. Test MAE on QM8 dataset.

Method	Test MAE($\times 10^{-3}$)
Origin AdaLanczosNet	11.77($\pm$0.8)
AdaLanczosNet with our improvement Lanczos Alg	9.89($\pm$0.04)

Table 3. Node classification accuracy on real world datasets.

Method	DataSet					
	Telegram	CiteSeer	Cora-ML	Wisconsin	Cornell	Texas
GCN	73.4($\pm$5.8)%	66.0($\pm$1.5)%	82.0($\pm$1.1)%	55.9($\pm$5.4)%	59.0($\pm$6.4)%	58.7($\pm$3.8)%
ChebNet	70.2($\pm$6.8)%	66.7($\pm$1.6)%	80.0($\pm$1.8)%	81.6($\pm$6.3)%	79.8($\pm$5.0)%	79.2($\pm$7.5)%
APPNP	67.3($\pm$3.0)%	66.9($\pm$1.8)%	**82.6**($\pm$1.4)%	51.8($\pm$7.4)%	58.7($\pm$4.0)%	57.0($\pm$4.8)%
SAGE	56.6($\pm$6.0)%	66.0($\pm$1.5)%	82.3($\pm$1.2)%	83.1($\pm$4.8)%	80.0($\pm$6.1)%	84.3($\pm$5.5)%
GIN	74.4($\pm$8.1)%	63.3($\pm$2.5)%	78.1($\pm$2.0)%	58.2($\pm$5.1)%	57.9($\pm$5.7)%	65.2($\pm$6.5)%
GAT	72.6($\pm$7.5)%	67.3($\pm$1.3)%	81.9($\pm$1.0)%	54.1($\pm$4.2)%	57.6($\pm$4.9)%	61.2($\pm$5.0)%
DGCN	90.4($\pm$5.6)%	66.3($\pm$2.0)%	81.3($\pm$1.4)%	65.5($\pm$4.7)%	66.3($\pm$2.0)%	71.7($\pm$7.4)%
Digraph	82.0($\pm$3.1)%	62.6($\pm$2.2)%	79.4($\pm$1.8)%	59.6($\pm$3.8)%	66.8($\pm$6.2)%	64.9($\pm$8.1)%
DigraphIB	64.1($\pm$7.0)%	61.1($\pm$1.7)%	79.3(1.2)%	64.1($\pm$7.0)%	64.4($\pm$9.0)%	64.9($\pm$13.7)%
MagNet	87.6($\pm$2.9)%	67.5($\pm$1.8)%	79.8($\pm$2.5)%	85.7($\pm$3.2)%	84.3($\pm$7.0)%	83.3($\pm$6.1)%
Our Method	**92.5**($\pm$4.7)%	**69.1**($\pm$1.7)%	81.7($\pm$1.5)%	**87.4**($\pm$5.8)%	**86.7**($\pm$4.3)%	**89.4**($\pm$8.2)%
q	0.15	0.0	0.0	0.05	0.25	0.15
$longscale$	$[15, 25, 35, 45]$	$[10, 20, 30, 40]$	$[10, 25, 30, 45]$	$[10, 20, 25, 35]$	$[5, 10, 15, 25]$	$[5, 10, 15, 20]$
$LanczosStep$	125	1500	1400	125	90	90

caused by rounding errors during the iteration process of the Lanczos algorithm has a significant impact on both training stability and final classification accuracy.

Results and Analysis on MSDGCNN. Our multi-scale directed graph convolution neural network achieved outstanding performance in node classification tasks on multiple datasets as shown in Table 3. We achieved the best classification accuracy on the large-scale CiteSeer dataset. Although we did not achieve the highest classification accuracy on the Cora-ML dataset, our method outperformed MagNet which only uses short-scale convolution filters by nearly 2%. This indicates that our long-scale convolution filter constructed by improved Lanczos algorithm is effective. In experiment on four small-scale datasets (including Wisconsin, Cornell, Texas and Telegram), our method achieved the best node classification accuracy. Experimental results on the real-world directed graph datasets demonstrate that the MSDGCNN has better overall performance compared to most other state-of-the-art GNNs.

5 Conclusion

We improved the Lanczos algorithm by using the modified Gram-Schmidt orthogonalization technique combining orthogonalization check method, and the result on QM8 dataset demonstrated that our improved algorithm has better stability. Similarly, our designed Multi-scale Directed Graph Convolution Neural Network (MSDGCNN) which aggregates larger scale information of a node with lower computational overhead showed outstanding performance on numerous real-world directed graph datasets. However, from the results of experiments on MSDGCNN, it can be observed that increasing the convolution scale of the convolution layer has limited contribution to improving the accuracy of node classification tasks on large-scale sparse graphs, and increasing the convolution scale will further increase the training time and difficulty at same time. Furthermore, there is still a need for further discussion on how to utilize spectral-based graph convolution neural networks to handle multi-graph structures with multiple types of edges between nodes.

Acknowledgements. This work is financially supported by: The National Key R&D Program of China (No. 2020YFB1712600); The Fundamental Research Funds for Central University (No. 3072022QBZ0601); and The State Key Laboratory of Underwater Robotics Technology at Harbin Engineering University (No. KY70100200052).

References

1. Bojchevski, A., Günnemann, S.: Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815 (2017)
2. Bovet, A., Grindrod, P.: The activity of the far right on telegram (2020)
3. Chen, B.: Molecular Graph Representation Learning and Generation for Drug Discovery. Ph.D. thesis, Massachusetts Institute of Technology (2022)
4. Chung, F.R.: Spectral graph theory, vol. 92. American Mathematical Soc. (1997)
5. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems 29 (2016)
6. Fanuel, M., Alaiz, C.M., Suykens, J.A.: Magnetic eigenmaps for community detection in directed networks. Phys. Rev. E **95**(2), 022302 (2017)
7. Furutani, S., Shibahara, T., Akiyama, M., Hato, K., Aida, M.: Graph signal processing for directed graphs based on the hermitian laplacian. In: Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C. (eds.) ECML PKDD 2019. LNCS (LNAI), vol. 11906, pp. 447–463. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46150-8_27
8. Gasteiger, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997 (2018)
9. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR (2017)
10. Golub, G.H., Van Loan, C.F.: Matrix computations. JHU press (2013)
11. Grcar, J.F.: Analyses of the Lanczos Algorithm and of the Approximation Problem in Richardson's Method. University of Illinois at Urbana-Champaign (1981)

12. Guo, K., Mohar, B.: Hermitian adjacency matrix of digraphs and mixed graphs. J. Graph Theory **85**(1), 217–248 (2017)
13. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems 30 (2017)
14. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Appl. Comput. Harmon. Anal. **30**(2), 129–150 (2011)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
16. He, Y., Perlmutter, M., Reinert, G., Cucuringu, M.: Msgnn: a spectral graph neural network based on a novel magnetic signed laplacian. In: Learning on Graphs Conference, pp. 40–1. PMLR (2022)
17. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
18. Huang, T., Dong, Y., Ding, M., Yang, Z., Feng, W., Wang, X., Tang, J.: Mixgcf: An improved training method for graph neural network-based recommender systems. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 665–674 (2021)
19. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
20. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
21. Ko, T., Choi, Y., Kim, C.K.: A spectral graph convolution for signed directed graphs via magnetic laplacian. Neural Networks (2023)
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
23. Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators (1950)
24. LeClair, A., Haque, S., Wu, L., McMillan, C.: Improved code summarization via a graph neural network. In: Proceedings of the 28th International Conference on Program Comprehension, pp. 184–195 (2020)
25. Levie, R., Huang, W., Bucci, L., Bronstein, M., Kutyniok, G.: Transferability of spectral graph convolutional neural networks. J. Mach. Learn. Res. **22**(1), 12462–12520 (2021)
26. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting. arXiv preprint arXiv:1707.01926 (2017)
27. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015)
28. Liao, R., Zhao, Z., Urtasun, R., Zemel, R.S.: Lanczosnet: multi-scale deep graph convolutional networks. arXiv preprint arXiv:1901.01484 (2019)
29. Lieb, E.H., Loss, M.: Fluxes, laplacians, and kasteleyn's theorem. Statistical Mechanics: Selecta of Elliott H. Lieb, pp. 457–483 (2004)
30. Paige, C.C.: Computational variants of the lanczos method for the eigenproblem. IMA J. Appl. Math. **10**(3), 373–381 (1972)
31. Paige, C.C.: Error analysis of the lanczos algorithm for tridiagonalizing a symmetric matrix. IMA J. Appl. Math. **18**(3), 341–349 (1976)
32. Parlett, B.N., Scott, D.S.: The lanczos algorithm with selective orthogonalization. Math. Comput. **33**(145), 217–238 (1979)
33. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-gcn: geometric graph convolutional networks. arXiv preprint arXiv:2002.05287 (2020)

34. Ramakrishnan, R., Hartmann, M., Tapavicza, E., Von Lilienfeld, O.A.: Electronic spectra from tddft and machine learning in chemical space. J. Chem. Phys. **143**(8), 084111 (2015)
35. Ramsundar, B., Eastman, P., Walters, P., Pande, V., Leswing, K., Wu, Z.: Deep Learning for the Life Sciences. O'Reilly Media (2019)
36. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
37. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process. Mag. **30**(3), 83–98 (2013)
38. Simon, H.D.: The lanczos algorithm with partial reorthogonalization. Math. Comput. **42**(165), 115–142 (1984)
39. Susnjara, A., Perraudin, N., Kressner, D., Vandergheynst, P.: Accelerated filtering on graphs using lanczos method. arXiv preprint arXiv:1509.04537 (2015)
40. Tong, Z., Liang, Y., Sun, C., Li, X., Rosenblum, D., Lim, A.: Digraph inception convolutional networks. Adv. Neural. Inf. Process. Syst. **33**, 17907–17918 (2020)
41. Tong, Z., Liang, Y., Sun, C., Rosenblum, D.S., Lim, A.: Directed graph convolutional network. arXiv preprint arXiv:2004.13970 (2020)
42. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
43. Wu, L., Cui, P., Pei, J., Zhao, L., Guo, X.: Graph neural networks: foundation, frontiers and applications. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 4840–4841 (2022)
44. Wu, Z., et al.: Moleculenet: a benchmark for molecular machine learning. Chem. Sci. **9**(2), 513–530 (2018)
45. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst. **32**(1), 4–24 (2020)
46. Xu, B., Shen, H., Cao, Q., Qiu, Y., Cheng, X.: Graph wavelet neural network. arXiv preprint arXiv:1904.07785 (2019)
47. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
48. Zhang, C., et al.: Deeptralog: trace-log combined microservice anomaly detection through graph-based deep learning (2022)
49. Zhang, J., Hui, B., Harn, P.W., Sun, M.T., Ku, W.S.: Mgc: a complex-valued graph convolutional network for directed graphs. arXiv e-prints pp. arXiv-2110 (2021)
50. Zhang, X., He, Y., Brugnone, N., Perlmutter, M., Hirn, M.: Magnet: a neural network for directed graphs. Adv. Neural. Inf. Process. Syst. **34**, 27003–27015 (2021)
51. Zhou, J., et al.: Graph neural networks: a review of methods and applications. AI Open **1**, 57–81 (2020)