SEA-GWNN: Simple and Effective Adaptive Graph Wavelet Neural Network

Swakshar Deb[1], **Sejuti Rahman**[1] *, **Shafin Rahman**[2]

[1]Department of Robotics and Mechatronics Engineering, University of Dhaka, Bangladesh
[2]Department of Electrical and Computer Engineering, North South University, Bangladesh
swakshar.sd@gmail.com, sejuti.rahman@du.ac.bd, shafin.rahman@northsouth.edu

Abstract

The utilization of wavelet-based techniques in graph neural networks (GNNs) has gained considerable attention, particularly in the context of node classification. Although existing wavelet-based approaches have shown promise, they are constrained by their reliance on pre-defined wavelet filters, rendering them incapable of effectively adapting to signals that reside on graphs based on tasks at hand. Recent research endeavors address this issue through the introduction of a wavelet lifting transform. However, this technique necessitates the use of bipartite graphs, causing a transformation of the original graph structure into a bipartite configuration. This alteration of graph topology results in the generation of undesirable wavelet filters, thereby undermining the effectiveness of the method. In response to these challenges, we propose a novel simple and effective adaptive graph wavelet neural network (SEA-GWNN) class that employs the lifting scheme on arbitrary graph structures while upholding the original graph topology by leveraging multi-hop computation trees. A noteworthy aspect of the approach is the focus on local substructures represented as acyclic trees, wherein the lifting strategy is applied in a localized manner. This locally defined lifting scheme effectively combines high-pass and low-pass frequency information to enhance node representations. Furthermore, to reduce computing costs, we propose to decouple the higher-order lifting operators and induce them from the lower-order structures. Finally, we benchmark our model on several real-world datasets spanning four distinct categories, including citation networks, webpages, the film industry, and large-scale graphs and the experimental results showcase the efficacy of the proposed SEA-GWNN.

Introduction

Graphs exhibit versatility and complexity by adeptly representing intricate data structures of various forms, such as molecular compositions, e-commerce dynamics, and scholarly citations, complete with their inherent entities and connections. Recently, there has been a significant focus on wavelet-based techniques for comprehending graph structures, catering to tasks like node classification and graph categorization, as evidenced by research works such as (Xu et al. 2018a, 2022; Zheng et al. 2021; Li et al. 2020; Zheng et al. 2023). Specifically, for node classification, these graphs can be divided into two distinct classes: homophilic and heterophilic. Homophilic graphs embody the assumption that nodes share resemblances in attributes and labels with their neighboring nodes, as observed in instances like citation and coauthor networks. Conversely, the real-world scenario also comprises heterophilic graphs, wherein nodes are linked across differing classes featuring disparate attributes, as seen in contexts like dating platforms and molecular networks. Existing wavelet-based approaches (Xu et al. 2018a; Li et al. 2020; Xu et al. 2022; Zheng et al. 2021) utilized a fixed set of wavelet filters while exploiting the homophilic assumption, where nodes are considered to share the same class label as their neighbors. However, due to this inherent assumption of homophily incorporated in the filter

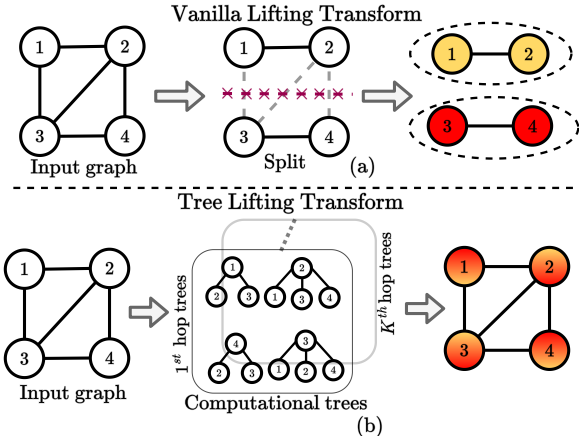


Figure 1: Overview of (a) existing vanilla lifting-based adaptive wavelet approaches vs (b) proposed adaptive wavelet approach, SEA-GWNN. The high and low-frequency feature is represented with Yellow and Red nodes, respectively, and the fusion of different frequencies is indicated by the color gradient node. Existing approaches lead to undesirable wavelet filters due to the alteration of the original graph structure in the split stage. In contrast, SEA-GWNN with our proposed tree lifting transform retains the local connectivity of the original graph through multi-hop computational trees, thereby being able to produce desirable wavelet filters.

*Corresponding author

design, these wavelet-based networks fail to generalize well in the heterophilic setup. Moreover, these approaches primarily designed filters only considering the graph structure, neglecting the signal residing over the graph. Hence, this limits their flexibility to adopt both the homophilic and heterophilic graphs based on the signal residing over the graph.

To overcome these limitations, Xu *et. al.* (Xu et al. 2022) proposed to learn the wavelet filters using both the graph structure and node attributes with neural parameterized lifting scheme (Sweldens and Schroder 1997; Daubechies and Sweldens 1998; Sweldens 1998), The vanilla lifting scheme required a graph to be split into two independent components (e.g. bipartite graph). This requirement of bipartility modifies the original graph structure to learn the wavelet filters, hence leading to an undesirable wavelet basis (Figure 1(a)). Therefore, to prevent this structural information loss caused by the alteration of the original graph, Xu *et. al.* (Xu et al. 2022) proposed the diffusion-based lifting scheme, where prior to the lifting scheme they implemented a diffusion kernel to retain the neighborhood information. In this paper, we identify the following drawbacks of this lifting-based adaptive wavelet approach (Xu et al. 2022). ***(1)*** Xu *et. al.*'s (Xu et al. 2022) initial diffusion-based lifting scheme is only appropriate for the homophilic cases since diffusion wavelet can aggregate unnecessary neighborhood information for the heterophilic graphs, where neighbors contain different labels. ***(2)*** Moreover, after using the diffusion filter, the implementation of the traditional lifting scheme also tempers the input graph topology due to the random split, thus producing undesirable class of wavelet filters. ***(3)*** Furthermore, similar to previous wavelet based methods (Xu et al. 2018a; Zheng et al. 2021; Li et al. 2020), Xu *et. al.* (Xu et al. 2022)'s approach also needed preprocessing steps, which hindered their applicability for numerous real-world tasks. In this paper, we attempt to solve these aforementioned problems. To address these issues, in contrast with the diffusion-based lifting scheme (Xu et al. 2022), we proposed tree-based lifting scheme incorporated within our **S**imple and **E**ffective **A**daptive **G**raph **W**avelet **N**eural **N**etwork (SEA-GWNN). This tree-based lifting scheme first constructs the multiscale tree based on the multi-hoop graph adjacency matrix. Specifically, for each tree, we considered the diagonal entry of the adjacency matrix as the root node and the off-diagonal elements as the leaf nodes. Subsequently, we apply the lifting scheme locally on each of these trees. As these trees are derived from the graph adjacency matrix, which reassembles the local connectivity, they fully retain the overall structure of the graph, preventing the loss of structural information, hence, leading to desirable wavelet filters (Figure 1(b)). Moreover, since trees are an acyclic graph, they guarantee the essential bipartite condition required for the lifting scheme. In addition, during vanilla lifting transform, whereas the high and low-frequency information is separated through different subsets (Figure 1(a)), we combine this information residing over each tree with the fusion operation that further improves the feature representation of each node (Figure 1(b)). Furthermore, to reduce the computational cost, we propose to decouple the higher-order lifting operators and induce them from the lower-order lifting structures with our detachment strategy. Finally, compared to the recently proposed GA-GWNN (Deb, Rahman, and Rahman 2023), we have the following advantages: multiscale neighborhood information, no need for inverse lifting transform, and improved time complexity. In a nutshell, our contributions are below:

- We introduce a novel lifting-based framework for graph wavelet networks. Our algorithm preserves the original graph topology while concentrating on the surrounding structure from multiple scales as multi-hop computation trees with the bipartite constituent.
- In contrast to the existing lifting transform, where the low and high-frequency information is separated in different subsets, our fusion operation further combines this information to improve the representation of each node. Moreover, we further reduce the computational cost of the network with our proposed detachment strategy of the higher-order lifting operators.
- We evaluated our model on various real-world datasets. The empirical results indicate that our SEA-GWNN notably outperforms existing graph wavelet networks, thus demonstrating the effectiveness of our method.

Method

Lifting-based Adaptive Wavelet transform: Typically, the lifting transform (Sweldens 1998) consists of three fundamental steps (Sweldens and Schroder 1997; Claypoole et al. 2003; Rodriguez et al. 2020): split (lazy wavelet transform), predict, and update. ***(1)*** **Split:** The main purpose of this stage is to divide the graph into two independent components or bipartite constitute, namely even ($\mathcal{E}$) and odd ($\mathcal{O}$) sets, by decomposing the graph signal as: (ϕ): $x(k) = \psi(n) + \phi(m)$, where $\psi(n) = x(2k+1) \in \mathcal{O}$ represents the signal delayed down-sampled signal while $\phi(m) = x(2k) \in \mathcal{E}$ denotes the sampled signal. For digital signals (1D graphs), which are inherently bipartite, the aforementioned splitting process preserves their structure. However, when dealing with arbitrary graph structures that lack bipartite properties, this naive splitting approach results in the loss of structural information. To address this issue, Xu *et. al.* (Xu et al. 2022) introduced a diffusion-based lifting scheme, where prior to the split stage, they applied a diffusion or smoothing kernel over the input graph, which helped to retain the initial neighborhood information. ***(2)*** **Predict:** This stage is equivalent of performing a high pass filtering with the input signal x for wavelet coefficients on the odd set, defined as: $x_o[i] = x[i] - \boldsymbol{\mathcal{P}}(x)[i]$ where $x_o[i]$ is the detail coefficient for the odd node v_i, $\boldsymbol{\mathcal{P}}(x)$ is the linear combination of x with the prediction weight $\boldsymbol{\mathcal{P}}_{ij}$ as $\boldsymbol{\mathcal{P}}(x)[i] = \sum_{j \sim \mathcal{N}_{v_i}} \boldsymbol{\mathcal{P}}_{ij} x[j]$, where $\mathcal{N}_{v_i}$ is the neighbors of v_i. ***(3)*** **Update:** A low frequency information of the signal x is obtained by updating the even coefficients x_e with detail coefficient, x_o, as $x_e[j] = x[j] + \boldsymbol{\mathcal{U}}(x_o)[j]$ where x_e is the approximation coefficients, $\boldsymbol{\mathcal{U}}(x_o)$ is the linear combination of x_o as $\boldsymbol{\mathcal{U}}(x_o)[i] = \sum_{j \sim \mathcal{N}_{v_i}} \boldsymbol{\mathcal{U}}_{ij} x_o[j]$ and $\boldsymbol{\mathcal{U}}_{ij}$ is the update weight. Thus this produces a 2-channel forward lifting filter bank, $\mathbf{F} = \boldsymbol{\mathcal{U}} || \boldsymbol{\mathcal{P}}$, where $||$ is the channel-wise concatenation operation.

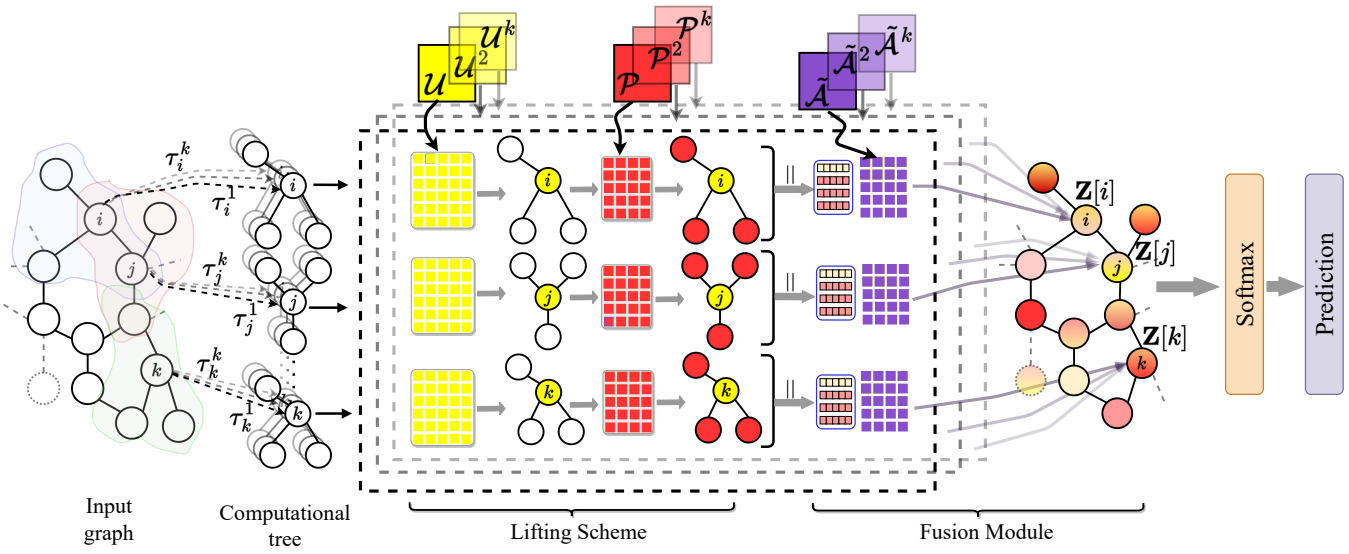


Figure 2: A depiction of the SEA-GWNN at the ℓ-th layer. The nodes with high and low-frequency information are denoted by red and yellow balls, respectively. The color gradient balls represent the combination of both low and high-frequency information, L is the total layer number. Symbols above the arrows denote specific operations as detailed in the paper.

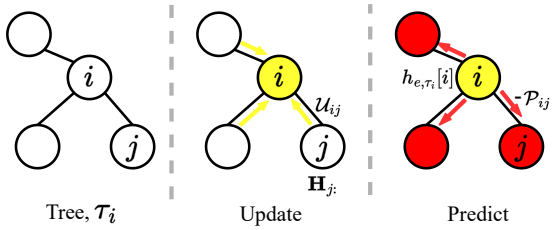


Figure 3: A depiction of our tree lifting scheme for tree τ_i, where $k = 1$ and each color represents frequency.

Problem Formulation: Considering $\mathbf{X} \in \mathbb{R}^{N\times d}$ as the input feature matrix, the output of existing lifting-based adaptive graph wavelet networks is as follows:

$$\mathbf{Y} = \text{Softmax}(\boldsymbol{\Omega}_{-1}(\hat{\mathbf{g}}\boldsymbol{\Omega}(\mathcal{G}, \mathbf{X}))) \in \mathbb{R}^{N\times\mathcal{C}} \quad (1)$$

Here, $\mathbf{Y}$ denotes the output class labels, $\mathcal{C}$ denotes the length of the entire class set, $\hat{\mathbf{g}}$ is the soft thresholding operation, $\boldsymbol{\Omega}(\cdot)$ is the forward lifting transform, and $\boldsymbol{\Omega}_{-1}(\cdot)$ is the inverse lifting transform that take the filtered wavelet signal back to the original domain. During the split stage, the vanilla lifting transform takes the entire graph as input and splits it into two independent components, which leads to the disruption of the original graph structure. In the subsequent section, we address this issue with our novel multi-hop computational tree-based lifting transform.

Proposed Tree Lifting Scheme

The aim of our tree lifting scheme, $\boldsymbol{\Omega}(\cdot)$, is to construct a two channel filter bank (a low and high pass filter) that decomposes the overall graph signal into high and low-frequency components. ***(i) Split***: To address the structural informational loss at the split stage, in contrast to the naive split mechanism (Xu et al. 2022), we introduce a tree-based split mechanism. Consider a graph, $\mathcal{G}$, with a vertex set $\mathbf{V}$ and an edge set $\mathbf{E}$, additionally equipped with a set of self-looped graph adjacency matrix as $\{\mathbf{A}^k \in \mathbb{R}^{N\times N}\}_{k=1}^{K}$ where the hop set $\boldsymbol{\mathcal{K}} = \{1, 2, \ldots, K\}$, each element of this hop set denotes a specific hop size. For a computational tree τ_i^k, where subscript i denotes the root node v_i and the superscript k represents the hop size, we consider the diagonal entry of the i-th row $\mathbf{A}_{ii}^k$ as the root node and the off-diagonal entries of the i-th row $\mathbf{A}_{ij}^k$ where $i \neq j$ as the leaf nodes. Consequently, this process dissects a graph into a set of multihop computational trees as $\tau = \{\tau_i^k | k \in \boldsymbol{\mathcal{K}}, i \in \mathbf{V}\}$. Importantly, this collection of the computational trees preserves the local connectivity or structure of the graph through the utilization of the graph adjacency matrix. Regarding the disjoint even and odd sets for the computational tree τ_i^k, we designate the leaf nodes to the odd set while assigning the root node to the even set. Since trees guarantees bipartite property, this partitioning or assignment of even-odd subsets preserve the overall structure. Following this assignment, we implement the lifting scheme (predict and update operation) on each tree τ_i^k to learn the lifting-based wavelet filters based on the graph attribute. Thus, in contrast to the existing lifting scheme (Xu et al. 2022; Sweldens and Schroder 1997; Sweldens 1998), our proposed tree-based lifting scheme is a locally defined lifting scheme. Since the computational trees are taken as an input to our proposed tree lifting scheme, thus modifying the Eqn 1 as:

$$\mathbf{Y} = \text{Softmax}(\boldsymbol{\Omega}(\tau, \mathbf{H})); \qquad \mathbf{H} = f_\theta(\mathbf{X}) \quad (2)$$

where $\boldsymbol{\Omega}(\cdot)$ is the lifting scheme with learned wavelet transform, where it learns to improve the lazy wavelet transform based on both the node attribute and graph structure, $\mathbf{H} \in \mathbb{R}^{N\times d}$ is the transformed feature matrix through a learnable network $f_\theta(\cdot)$. Note that, unlike Eqn 1, we decou-

ple the feature transformation in Eqn 2 similar to APPNP (Gasteiger, Bojchevski, and Günnemann 2018). Moreover, in contrast to the vanilla lifting based network in Eqn 1, our formulation in Eqn 2 directly utilizes wavelet domain information without incorporating any inverse lifting transform. These modifications minimize the extra computational cost. ***(ii) Update***: For the computational tree τ_i^k, the approximation (smooth) coefficients are derived utilizing the feature vectors of the leaf nodes, where it attempts to predict the representation of the root node, as illustrated in Figure 3. The update stage is carried out as follows:

$$h_{e,\tau_i^k}[i] = \mathbf{H}_{i:} + \sum_{j \sim \mathcal{N}_i} \boldsymbol{\mathcal{U}}_{ij}^k \mathbf{H}_{j:} \tag{3}$$

where $h_{e,\tau_i^k}[i] \in \mathbb{R}^d$ represents the low-pass information related to the root node v_i at tree τ_i^k, $\boldsymbol{\mathcal{U}}^k \in \mathbb{R}^{N \times N}$ is the k-th hop update operator, similar to attention based adjacency matrix, that is learned through the network (discussed in the Sec. Structure Aware Lifting Operators). ***(iii) Predict***: The predict stage aims to extract the high-frequency feature associated with root node by eliminating the local correlation present in the signal. To do so, consider a tree τ_i^k, the root node v_i from the even set is aimed to predict its leaf nodes from the odd set as shown in Figure 3. Thus, whenever the is a high local correlation between the root and its neighbor in a tree, this rises to a small residual information (small detail coefficient). We define the prediction stage as:

$$h_{o,\tau_i^k}[i,j] = \mathbf{H}_{j:} - \boldsymbol{\mathcal{P}}_{ij}^k h_{e,\tau_i^k}[i] \tag{4}$$

where each $h_{o,\tau_i^k}[i,j] \in \mathbb{R}^d$ represents the high-pass signal related to root node v_i concerning the neighbor node v_j at tree τ_i^k, $\boldsymbol{\mathcal{P}}^k \in \mathbb{R}^{N \times N}$ is the k-th hop predict operator (discussed in Sec. Structure Aware Lifting Operators). In the aforementioned tree-lifting scheme, similar to the vanilla lifting scheme (Xu et al. 2022; Sweldens 1998; Sweldens and Schroder 1997), for each tree τ_i^k the low and high-frequency nodes are separated through even and odd subsets, respectively. Although, this approximation and detail components acquired through the predict and update operations are different, they are in complementary aspects. Hence, we aggregate these information together to obtain the overall representation with the help of a fusion operation. More specifically, after performing the lifting scheme locally at tree τ_i^k, where the root node v_i captures the low-frequency information of the surroundings and the high-frequency information is captured by the leaf nodes, we fuse these decomposed signals associated with the tree τ_i^k as follows: $z_k[i] = \alpha \sum_{j \in \mathcal{N}_i} \tilde{\mathcal{A}}_{ij}^k h_{o,\tau_i^k}[i,j] + (1-\alpha) h_{e,\tau_i^k}[i]$. With the help of Eqn. 3 & 4, we write this with respect to the input feature vector as: $z_k[i] = \alpha(\sum_{j \in \mathcal{N}_i} \tilde{\mathcal{A}}_{ij}^k \mathbf{H}_{j:}) + (\sum_{j \in \mathcal{N}_i} \boldsymbol{\mathcal{U}}_{ij}^k \mathbf{H}_{j:} + \mathbf{H}_{i:})(1 - \alpha - \alpha\beta_k \sum_{j \in \mathcal{N}_i} \tilde{\mathcal{A}}_{ij} \boldsymbol{\mathcal{P}}_{ij})$, when $k > 1$, β_k is the learnable scalar for the k-th hop, otherwise β_k is set to 1. This form is expressed into a single matrix expression for all the k hoop computational trees as:

$$\mathbf{Z}_k = \alpha \cdot \tilde{\mathcal{A}}^k \mathbf{H} + \tilde{\boldsymbol{\mathcal{U}}}^k \mathbf{H} \circledast \boldsymbol{\xi}^k \tag{5}$$

Here, $\mathbf{Z}_k \in \mathbb{R}^{N \times d}$ is the learned feature matrix for all the k hop trees, $\tilde{\mathcal{A}}^k$ is the k-th hop normalize adjacency matrix without the self-loop, the hyperparameter $\alpha \in [0,1]$ is referred as the balancing factor since it controls the frequency spectrum of the wavelet filter, $\mathcal{I}\mathscr{D}$ is the identity operator, $\boldsymbol{\xi}^k = (1-\alpha)E - \alpha\beta_k E^T(\boldsymbol{\mathcal{P}} \odot \tilde{\mathcal{A}}) \in \mathbb{R}^N$, where E is the all one vector of size N, $\odot$ is the hadamard product, and $\tilde{\boldsymbol{\mathcal{U}}} = \boldsymbol{\mathcal{U}} + \mathcal{I}\mathscr{D}$, and $(\mathbf{H} \circledast \boldsymbol{\xi}^k)_{i:} = \boldsymbol{\xi}_i^k \cdot \mathbf{H}_{i:}$, intuitively, this only modulates the feature vector of the node v_i, $\mathbf{H}_{i,:}$, with a scalar $\boldsymbol{\xi}_i^k$. Finally, the overall representation $\mathbf{Z} = \sum_{k \in \mathcal{K}} \gamma_k \mathbf{Z}_k$ incorporates the extracted multiscale feature information with learnable parameter $\gamma_k \in \mathbb{R}$. An overview of the SEA-GWNN is given in Algorithm 1.

Algorithm 1: SEA-GWNN model training.

Input: Graph ($\mathcal{G}$) adjacency matrix $\mathcal{A}$, node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, set of ground truth $\mathcal{Y}$, max layer L
Output: Predicted class label: $\mathbf{Y}$.

Initialize model parameters
$\mathbf{H}_0, \tilde{\mathbf{H}}_0 \leftarrow \mathbf{X}$;
$\boldsymbol{\mathcal{U}} \leftarrow \text{ATT}_{gat}(\mathcal{G}, \mathbf{X})$; /* See Eqn 6 */
$\boldsymbol{\mathcal{P}} \leftarrow \frac{1}{2}\boldsymbol{\mathcal{U}}$
$\tilde{\boldsymbol{\mathcal{U}}} \leftarrow \boldsymbol{\mathcal{U}} + \mathcal{I}\mathscr{D}$
for *hop size,* $k \leftarrow 1 \ldots K$ **do**
  $\mathbf{H}_k \leftarrow \mathcal{A}\mathbf{H}_{k-1}$
  $\tilde{\mathbf{H}}_k \leftarrow \tilde{\boldsymbol{\mathcal{U}}}\tilde{\mathbf{H}}_{k-1}$
  $\boldsymbol{\xi}^k \leftarrow (1-\alpha)E - \alpha\beta_k E^T(\boldsymbol{\mathcal{P}} \odot \tilde{\mathcal{A}})$
  $\mathbf{Z}_k \leftarrow \alpha\mathbf{H}_k + \tilde{\mathbf{H}}_k \circledast \boldsymbol{\xi}^k$; /* See Eqn 5 */
  $\mathbf{Z} \mathrel{+}= \gamma_k \mathbf{Z}_k$
end
$\mathbf{Y} \leftarrow \text{Softmax}(\mathbf{Z})$
$\mathcal{L} = \text{Loss}(\mathbf{Y}, \mathcal{Y})$
Backpropagation and update parameters

Structure Aware Lifting Operators

In this part, first, we will discuss the construction of the first hop lifting operators (predict and update operator), then we will generalize this for the higher-order lifting operators. The first hop prediction and update operators (first hop lifting operators), $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{U}}$, are learned with a structure-aware attention mechanism. While previous studies employed predefined lifting operators (Narang and Ortega 2009) or altered graph's topology (i.e., removal of edges) (Xu et al. 2022) to acquire these operators, we learn them with the original graph topology that captures pairwise node relationships. Additionally, in contrast to the linear lifting operators presented in (Xu et al. 2022), we employed data-dependent non-linear lifting operators. Specifically, motivated by the success of Graph Attention Convolutional Network (GAT) (Veličković et al. 2018), our attention mechanism is similarly defined as:

$$\mathbf{W}_{ij} = \text{ATT}_{gat}(\mathcal{G}, h_i, h_j) = \sigma(\mathbf{b}_s[h[i] || h[j]]) \tag{6}$$

Here, $\mathbf{b}_s \in \mathbb{R}^{2d}$ is the trainable parameters shared across all nodes, $||$ denotes the channel-wise concatenation operation and σ is the nonlinear activation LeakyRelu. The trainable

matrix $\mathbf{W}$ captures the relationships between nodes pair, and this information is injected in the calculation of predict, and update operators as: $\boldsymbol{\mathcal{U}} = \mathbf{W}$, $\boldsymbol{\mathcal{P}} = \frac{1}{2}\mathbf{W}$ to ensure at least the first order vanishing moment (Xu et al. 2022). These lifting operators are then concatenated channel-wise as: $\mathbf{F} = \boldsymbol{\mathcal{P}} \| \boldsymbol{\mathcal{U}}$, where $\mathbf{F} \in \mathbb{R}^{N \times N \times 2}$, that induce a two-channel forward lifting filter bank. Finally, in the case of Eqn 5, instead of learning each higher-order update operator $\boldsymbol{\mathcal{U}}^k$ $(k > 1)$ separately, we induce this from the lower order structure $\boldsymbol{\mathcal{U}}$, with matrix multiplication operation similar to the calculation of $\tilde{\mathcal{A}}^k$, hence significantly reduces the computational cost. Furthermore, there is no need to precompute $\boldsymbol{\mathcal{U}}^k$ that requires extra storage, since we calculate $\boldsymbol{\mathcal{U}}^k\mathbf{H}$ with right-to-left multiplication as $\boldsymbol{\mathcal{U}}(\boldsymbol{\mathcal{U}}(\ldots(\boldsymbol{\mathcal{U}}H))$ (see Algorithm 1).

Experiment

Dataset Description: We use three common homophilic citation graphs **Cora**, **Citeseer**, and **PubMed** (Sen et al. 2008) and five heterophilic datasets namely **Cornell**, **Wisconsin** (Pei et al. 2019), **Film** (Tang et al. 2009), **Chameleon** and **Squirrel**. Moreover, we utilize four large-scale graphs namely **Penn94** (Traud, Mucha, and Porter 2012) **Genius**, and **Ogbn-Arxiv** (Hu et al. 2020).

Parameter Setup: Similar to (Pei et al. 2019), the nodes in each class were randomly split into three groups for training, validation, and testing in a ratio of $48\%, 32\%, 20\%$. The performance of all models was evaluated on the test sets over 10 random splits proposed by (Pei et al. 2019). The hyperparameters for the baseline methods were set as previously specified by the respective papers. For our algorithm, all the hyperparameters were determined through grid search. Specifically, weight decay factors were explored in the range of $[0.01, 0.000005]$, while dropout probabilities were selected from $[0, 0.9]$ with a step of 0.1. The hyperparameters were fine-tuned by monitoring their performance on the validation sets. Training continued for a maximum of 1000 epochs, with the help of an early stopping set at 150 epochs. The model parameters that show the best result on the validation set were chosen for evaluation on the test set. The mean test accuracy, along with the standard deviation, was computed from 10 distinct runs.

Overall Result

Full supervised Node Classification: The results in Table 1 clearly indicate that GCN (Kipf and Welling 2017) outperforms the DEEPWALK (Perozzi, Al-Rfou, and Skiena 2014) algorithm in every dataset. This performance gain signifies the importance of utilizing both graph topology and node attributes in the case of GCN, while DeepWalk solely focuses on graph structure through random walks. GAT (Veličković et al. 2018) further enhances the GCN (Kipf and Welling 2017) by incorporating the attention mechanism for message propagation. On the contrary, GWNN (Xu et al. 2018a) offered a sparse and localized wavelet basis by introducing wavelet-based convolution with diffusion kernel. However, approaches such as GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), GWNN (Xu et al. 2018a), and DEEPWALK (Perozzi, Al-Rfou, and Skiena 2014) solely exploit the homophily assumption, showing a poor performance for heterophilic cases (Table 1). To address this, G-GCN (Pei et al. 2019), MIXHOP (Abu-El-Haija et al. 2019), H2GCN (Zhu et al. 2020), APPNP (Gasteiger, Bojchevski, and Günnemann 2018), and D-GCN (Park et al. 2022) searched for more higher-order homophilic neighbors by increasing the hop size that in-turn improve results on heterophilic graphs (see Table 1). Nevertheless, these approaches mainly focus on capturing strong homophilic information in the multi-hop neighborhood. GPR-GCN (Chien et al. 2021) tackles this by providing weights to each hop that allow to learn appropriate filters. However, GPR-GCN treats each connected node pair equally, and its utilization of the Monomial basis function for approximating the spectral graph filter poses challenges in terms of optimization. Although, recently, BM-GCN (He et al. 2022) and HOG-GCN (Wang and Zhang 2022) utilized the pseudo labels to re-weight each connected node pairs with the guidance of heterophilic information, these approaches aim to reduce the information propagation between heterophilic neighbors that may posses vital information. Moreover, these algorithms tend to utilize a single filter that partially exploits the whole feature information. In response to these challenges, we propose SEA-GWNN, an advancement over the existing GWNN (Xu et al. 2018a). SEA-GWNN adaptively learns wavelet filters with the proposed tree-lifting scheme without relying on any prior assumptions over the graph data. Additionally, our tree-lifting-based SEA-GWNN exploits higher-order neighborhood information and captures local sub-graph structures at various scales with the incorporation of multiscale computational trees. Moreover, SEA-GWNN captures the complete input information by learning a two-channel wavelet filter bank that covers the whole frequency spectrum. Finally, SEA-GWNN improves the node-level representation further by combining low and high-frequency information which explains its superiority over previous methods (Table 1). Table 2 shows similar results, where SEA-GWNN consistently outperforms the existing methods, in the case of large-scale graphs. **Semi-Supervised Node Classification:** Table 3 presents a performance comparison among various wavelet-based methods. When comparing predefined wavelet filter-based approaches, UFG-CONV (Zheng et al. 2021) surpasses both HANET (Li et al. 2020) and GWNN (Xu et al. 2018a) by employing various band-pass filters that complement the node representations. On the contrary, LGWNN (Xu et al. 2022), which is an adaptive wavelet approach, achieves performance comparable to UFGCONV (Zheng et al. 2021). Nevertheless, we believe that this resemblance resulted from undesirable wavelet filters produced due to the alteration of the input graph topology in LGWNN (Xu et al. 2022). In line with this observation, our proposed SEA-GWNN also being an adaptive wavelet-based method, significantly outperforms previous approaches. Importantly, SEA-GWNN incorporates the tree lifting scheme that fully retains the graph structure, thus resulting in a desirable class of wavelet filters. This underscores the effectiveness of our adaptive wavelet filtering achieved with the proposed tree-lifting transform.

Method/ Acc.(%)	homophilic graph datasets			heterophilic graph datasets				
	Cora	Citeseer	PubMed	Film	Chameleon	Squirrel	Cornell	Wisconsin
DeepWalk (2014)	80.08 ±1.8	53.59 ±2.6	81.14 ±0.5	23.74 ±0.5	26.54 ±0.9	35.33 ±1.1	44.12 ±9.1	53.51 ±5.1
GCN (2017)	85.77 ±0.1	73.68 ±0.2	88.13 ±0.6	28.78 ±1.5	28.18 ±0.7	36.89 ±1.3	52.70 ±0.5	45.88 ±0.7
GWNN (2018a)	85.31 ±0.2	73.93 ±0.1	88.14 ±0.1	26.62 ±1.4	48.40 ±0.7	31.48 ±1.5	61.89 ±0.6	48.04 ±0.7
GAT (2018)	86.37 ±0.2	74.32 ±0.2	87.62 ±0.4	28.99 ±1.4	42.93 ±0.5	30.62 ±2.1	54.32 ±0.3	49.41 ±0.9
APPNP(2018)	87.87 ±0.2	76.53 ±0.2	89.40 ±1.4	34.86 ±1.1	54.30 ±0.6	34.77 ±0.3	73.51 ±1.1	69.02 ±1.6
G-GCN (2019)	84.91 ±1.1	75.16 ±1.9	88.41 ±0.6	32.39 ±1.5	61.06 ±0.5	38.28 ±0.3	55.68 ±8.4	62.55 ±5.2
Mixhop(2019)	87.61 ±0.8	76.26 ±1.3	85.31 ±0.6	32.22 ±2.3	60.50 ±2.5	43.80 ±1.5	73.51 ±6.3	75.88 ±4.9
H2GCN(2020)	87.69 ±1.3	75.95 ±2.1	88.78 ±0.5	36.71 ±1.4	58.38 ±1.7	37.90 ±2.1	78.92 ±5.2	82.57 ±3.2
AM-GCN (2020)	86.66 ±1.3	76.01 ±1.9	86.78 ±0.6	33.60 ±1.1	68.46 ±1.7	40.02 ±0.9	78.38 ±4.9	81.76 ±4.9
CPGNN (2021)	87.18 ±1.1	75.52 ±1.8	89.08 ±0.6	35.51 ±1.8	65.24 ±0.8	45.00 ±1.4	63.51 ±5.8	81.76 ±6.7
GPR-GNN(2021)	86.70 ±1.1	75.12 ±1.9	87.38 ±0.6	36.47 ±1.3	65.42 ±2.1	49.93 ±0.5	82.97 ±5.6	83.92 ±3.1
BM-GCN (2022)	87.71 ±1.5	75.94 ±2.6	**90.25** ±0.7	36.32 ±1.4	69.58 ±2.9	51.41 ±1.1	79.14 ±8.4	82.82 ±8.9
HOG-GCN (2022)	87.04 ±1.1	76.15 ±1.9	88.79 ±0.4	36.82 ±0.9	53.07 ±1.2	39.09 ±0.8	84.32 ±4.3	86.67 ±3.4
D-GCN (2022)	87.48 ±0.8	76.83 ±1.2	89.49 ±0.3	37.07 ±0.8	70.90 ±1.1	62.56 ±1.3	85.95 ±2.7	87.06 ±3.4
SEA-GWNN	**88.75** ±0.1	**77.13** ±0.2	89.75 ±0.1	**37.10** ±1.8	**72.57** ±0.18	**63.13** ±0.13	**85.95** ±4.6	**87.25** ±3.3

Table 1: The average accuracy for node classification across nine datasets in the case of full-supervise setup is shown and the highest performances are highlighted in bold font.

Methods	Penn94	Genius	Ogbn-Arxiv
MLP (2020)	73.6 ±0.4	86.6 ±0.1	55.0 ±0.1
Label Prop. (2020)	74.1 ±0.5	67.1 ±0.2	68.3 ±0.1
GCN (2017)	82.5 ±0.3	87.4 ±0.4	71.7 ±0.3
ChebNet (2016)	82.5 ±0.3	89.3 ±0.3	71.7 ±0.2
GAT (2018)	81.5 ±0.5	55.8 ±0.9	71.9 ±0.1
GCNJK (2018b)	81.6 ±0.5	89.3 ±0.2	72.2 ±0.2
GCNII (2020)	82.9 ±0.5	90.2 ±0.1	72.7 ±0.2
GPRGNN (2021)	81.3 ±0.2	90.1 ±0.3	71.7 ±0.2
SEA-GWNN	**84.4** ±0.3	**90.5** ±0.1	**72.9** ±0.5

Table 2: The mean accuracy on large-scale graphs and the best performances are highlighted in bold font.

Methods	Cora	Citeseer	PubMed
GraphSAGE (2017)	74.5 ±0.8	67.2 ±1.0	76.8 ±0.6
GAT (2018)	83.0 ±0.7	72.5 ±0.7	79.0 ±0.3
HANet (2020)	81.9	70.1	79.3
GWNN (2018a)	81.6 ±0.7	70.5 ±0.6	78.6 ±0.3
UFGConvS (2021)	83.0 ±0.5	71.0 ±0.6	79.4 ±0.4
UFGConvR (2021)	83.6 ±0.6	72.7 ±0.6	79.9 ±0.1
LGWNN (2022)	83.4 ±0.6	71.1 ±0.4	79.5 ±0.5
SEA-GWNN	**84.4** ±0.6	**72.8** ±0.3	**80.7** ±0.2

Table 3: The average accuracy of semi-supervised node classification across three datasets is shown and the best performances are highlighted in bold font.

Analysis of Parameters

Balancing Factor: We also investigate the sensitivity of the parameter α in SEA-GWNN. We take the **Chameleon**, **Squirrel**, **Citeseer**, and **Cora** datasets as examples. In our experimental study for the **Chameleon** and **Squirrel** in figure 4 (a), (b), we observe that as we increase the value of the hyperparameter α, the test accuracy of the SEA-GWNN gradually increases and achieved the best test performance at $\alpha = 1$. This is because both **Chameleon** and **Squirrel** have strong heterophily incorporated into them (low $\mathcal{H}(\mathcal{G})$), thus it is important to recognize the high-frequency pattern residing over these graphs and the higher values of α represent strong importance to the high-frequency features. Conversely, for the **Cora** and **Citeseer** datasets, we achieved the highest test accuracy at a smaller value of α, specifically at $\alpha = 0.85$ (figure 4(c),(d)). Since both **Cora** and **Citeseer** exhibit the homophily characteristic whereby each node has the same label that is expressed by their neighbors. Thus in line with this homophilic characteristic, smaller values of α incorporate further low-frequency content of the graph signal, thereby increasing the overall performance.

Hop Size: We assess the node classification accuracy of SEA-GCN using different hop sizes k, ranging from 2 to 32. The results are presented in Table 4. Both **Cora** and **PubMed** datasets exhibit similar trends in performance. When the hop size k is progressively increased, SEA-GWNN demonstrates consistent and competitive performance, benefiting from the utilization of multiscale higher-order structural information. Nonetheless, excessively large values of k lead to a decline in performance due to the overfitting issue commonly encountered in deep learning.

Visualization

To visually highlight the efficacy of our proposed method, we conducted a comparison between the output of GWNN (Xu et al. 2018a) and our model SEA-GWNN on the Cora dataset using t-SNE (Van der Maaten and Hinton 2008), as illustrated in Figure 5, where points with identical colors share the same label. The visual shows the limitation of the GWNN, as evidenced by the dispersed points of similar colors and the unwanted mixing of certain points with different colors. In contrast, our SEA-GWNN, utilizing adaptive wavelet filters, exhibits a visualization that distinctly separates various clusters when compared to GWNN. Consequently, this result further validates the claim that our proposed adaptive filtering can achieve superior performance

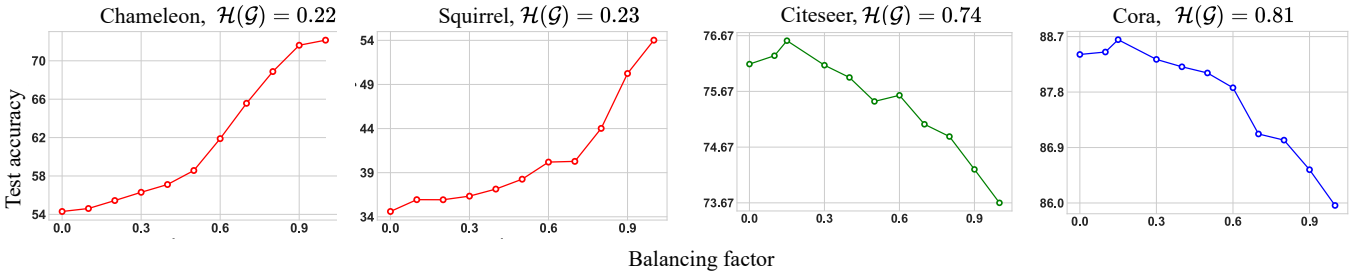
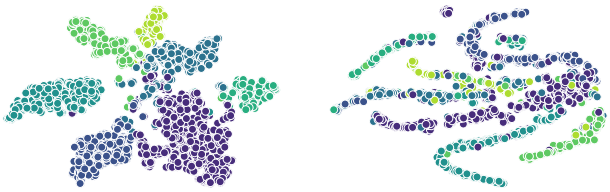
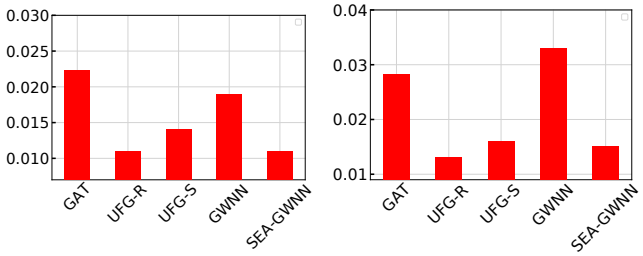


Figure 4: Analysis of the hyperparameter, balancing factor α, in both homophilic and heterophilic datasets.



(a) SEA-GWNN (b) GWNN

Figure 5: Visualization results on Cora dataset. In comparison to GWNN, SEA-GWNN has more distinct clusters.



(a) Cora (b) Citeseer

Figure 6: Empirical results of runtime (sec/epoch).

Datasets/ Accuracy (%)	The number of hop size k					
	2	4	6	8	16	32
Cora	82.46	84.04	**84.38**	84.34	82.52	80.86
PubMed	79.90	79.44	79.84	80.34	**80.74**	79.92

Table 4: Node classification accuracy of SEA-GWNN with graph convolutional layers k varying from 2 to 32.

gain in contrast to fixed wavelet filters.

Time Complexity Analysis

The calculation of $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{P}}$ i.e., lifting operators, can be parallelly executed for all the edges without any computationally intensive matrix operations such as eigendecompositions. Moreover, owing to the inherent sparsity of graphs (e.g. $\tilde{\mathcal{A}}$ and $\boldsymbol{\mathcal{U}}$), the time complexity of message propagation with PyTorch geometry (Fey and Lenssen 2019) yields $\mathbf{O}(|\mathbf{E}|d)$ at each layer. Hence for the first hop, this yields time complexity of $\mathbf{O}(Nd^2 + |\mathbf{E}|d)$. In addition, due to the detachment of lifting operators $\boldsymbol{\mathcal{P}}^k$ and $\boldsymbol{\mathcal{U}}^k$ for the higher order structures ($k > 1$), the total time complexity of the K hops SEA-GWNN is $\mathbf{O}(Nd^2 + K|\mathbf{E}|d)$. Moreover, in Fig 6 we empirically compare the computational time (sec/epoch) for various wavelet-based models (e.g. UFGConv-R, UFGConv-S, GWNN). To ensure a fair comparison, all models are evaluated on an NVIDIA RTX 3080 GPU. We can observe from Figure 6 that the GWNN (Xu et al. 2018a) has a much higher computational cost compared to SEA-GWNN due to the use of graph-dependent diagonal matrix multiplication operation. More importantly, unlike current GWNN approaches (Xu et al. 2022, 2018a; Zheng et al. 2021), which necessitate a preprocessing step to compute filters (the preprocessing time for GWNN and UFGConv is not taken into account in figure 6), SEA-GWNN utilizes proposed tree lifting scheme to learn the wavelet filters in place without requiring any extra time for the preprocessing step. This characteristic makes SEA-GWNN more adaptable for dynamic graphs and diverse real-world applications.

Conclusion

In this work, we proposed SEA-GWNN, a graph wavelet-based architecture, capable of learning wavelet filters for random graphs without imposing any assumptions over the graph e.g., homophily or heterophily. Additionally, SEA-GWNN incorporated with the proposed tree-lifting transform can construct wavelet filters while maintaining the original graph structure. This lifting scheme is locally applied to each multi-hop computation tree, where the bipartite condition is upheld. In addition, we further improve the node-level feature representation by employing the fusion operation. Moreover, to reduce the computational cost, instead of learning each lifting operator separately, we proposed a detachment strategy, where the higher-order operators are induced from the lower-order structure with right-to-left matrix multiplication. Furthermore, in contrast to the existing GWNN that implemented a uni-channel filter bank approach while partially retaining the input information, our SEA-GWNN adaptively learns a two-channel wavelet filter bank that can fully capture this information by covering the entire frequency spectrum. Finally, the effectiveness and efficiency of our proposed SEA-GWNN are justified by the extensive evaluation on various homophilic, heterophilic and large-scale graph datasets.

References

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Ver Steeg, G.; and Galstyan, A. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, 21–29. PMLR.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and deep graph convolutional networks. In *International conference on machine learning*, 1725–1735. PMLR.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*.

Claypoole, R. L.; Davis, G. M.; Sweldens, W.; and Baraniuk, R. G. 2003. Nonlinear wavelet transforms for image coding via lifting. *IEEE Transactions on Image Processing*, 12(12): 1449–1459.

Daubechies, I.; and Sweldens, W. 1998. Factoring wavelet transforms into lifting steps. *Journal of Fourier analysis and applications*, 4(3): 247–269.

Deb, S.; Rahman, S.; and Rahman, S. 2023. GA-GWNN: Generalized Adaptive Graph Wavelet Neural Network. *Pattern Recognition Letters*.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Gasteiger, J.; Bojchevski, A.; and Günnemann, S. 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

He, D.; Liang, C.; Liu, H.; Wen, M.; Jiao, P.; and Feng, Z. 2022. Block modeling-guided graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, 4022–4029.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Li, M.; Ma, Z.; Wang, Y. G.; and Zhuang, X. 2020. Fast Haar transforms for graph neural networks. *Neural Networks*, 128: 188–198.

Narang, S. K.; and Ortega, A. 2009. Lifting based wavelet transforms on graphs. In *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, 441–444. Asia-Pacific Signal and Information Processing Association, 2009 Annual ....

Park, J.; Yoo, S.; Park, J.; and Kim, H. J. 2022. Deformable graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 7949–7956.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2019. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

Rodriguez, M. X. B.; Gruson, A.; Polania, L.; Fujieda, S.; Prieto, F.; Takayama, K.; and Hachisuka, T. 2020. Deep adaptive wavelet network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3111–3119.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Sweldens, W. 1998. The lifting scheme: A construction of second generation wavelets. *SIAM journal on mathematical analysis*, 29(2): 511–546.

Sweldens, W.; and Schroder, P. 1997. Building your own wavelets at home. *Wavelets in Computer Graphics*, 1587.

Tang, J.; Sun, J.; Wang, C.; and Yang, Z. 2009. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 807–816.

Traud, A. L.; Mucha, P. J.; and Porter, M. A. 2012. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16): 4165–4180.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, X.; and Zhang, M. 2022. How powerful are spectral graph neural networks. In *International Conference on Machine Learning*, 23341–23362. PMLR.

Wang, X.; Zhu, M.; Bo, D.; Cui, P.; Shi, C.; and Pei, J. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining*, 1243–1253.

Xu, B.; Shen, H.; Cao, Q.; Qiu, Y.; and Cheng, X. 2018a. Graph Wavelet Neural Network. In *International Conference on Learning Representations*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018b. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, 5453–5462. PMLR.

Xu, M.; Dai, W.; Li, C.; Zou, J.; Xiong, H.; and Frossard, P. 2022. Graph Neural Networks With Lifting-Based Adaptive Graph Wavelets. *IEEE Transactions on Signal and Information Processing over Networks*, 8: 63–77.

Zheng, X.; Zhou, B.; Gao, J.; Wang, Y.; Lió, P.; Li, M.; and Montufar, G. 2021. How Framelets Enhance Graph Neural Networks. In *International Conference on Machine Learning*, 12761–12771. PMLR.

Zheng, X.; Zhou, B.; Li, M.; Wang, Y. G.; and Gao, J. 2023. MathNet: Haar-like wavelet multiresolution analysis for graph representation learning. *Knowledge-Based Systems*, 273: 110609.

Zhu, J.; Rossi, R. A.; Rao, A.; Mai, T.; Lipka, N.; Ahmed, N. K.; and Koutra, D. 2021. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11168–11176.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33: 7793–7804.