GRAPH WAVELET NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

We present graph wavelet neural network (GWNN), a novel graph convolutional neural network (CNN), leveraging graph wavelet transform to address the shortcoming of previous spectral graph CNN methods that depend on graph Fourier transform. Different from graph Fourier transform, graph wavelet transform does not require matrix eigendecomposition with high computational cost. Moreover, wavelets are sparse and localized in vertex domain, offering high efficiency and good interpretability for convolution. The proposed GWNN significantly outperforms previous spectral graph CNNs in the task of graph semi-supervised classification, on three benchmark datasets: Cora, Citeseer and Pubmed.

1 INTRODUCTION

Convolutional neural networks (CNNs) (LeCun et al., 1998) have been successfully used in many machine learning problems, such as image classification (He et al., 2016) and speech recognition (Hinton et al., 2012), where there are an underlying Euclidean structure. The success of CNNs lies in their ability to leverage the statistical properties of Euclidean data, e.g., translation invariance. However, in many research areas, data are naturally located in a non-Euclidean space, with graph or network being one typical case. The non-Euclidean nature of graph is the main obstacle or challenge when we attempt to generalize CNNs to graph. For example, convolution is not well defined in graph, due to that the size of neighborhood for each node varies dramatically (Bronstein et al., 2017).

Existing methods attempting to generalize CNNs to graph data fall into two categories, spatial methods and spectral methods, according to the way that convolution is defined. Spatial methods define convolution directly on the vertex domain, following the practice of the conventional CNN. For each vertex, convolution is defined as a weighted average function over all vertices located in its neighborhood, with the weighting function characterizing the influence exerting to the target vertex by its neighbors (Monti et al., 2017). The main challenge is to define a convolution operator that can handle neighborhood with different sizes and maintain the weight sharing property of CNN. Although spatial methods gain some initial success and offer us a flexible framework to generalize CNNs to graph, it is still elusive to determine appropriate neighborhood.

Spectral methods define convolution via graph Fourier transform and convolution theorem. Spectral methods leverage graph Fourier transform to convert signals defined in vertex domain into spectral domain, e.g., the space spanned by the eigenvectors of the graph Laplacian matrix, and then filter is defined in spectral domain, maintaining the weight sharing property of CNN. As the pioneering work of spectral methods, spectral CNN (Bruna et al., 2014) exploited graph data with the graph Fourier transform to implement convolution operator using convolution theorem. Some subsequent works devote to making spectral methods spectrum-free (Defferrard et al., 2016; Kipf & Welling, 2017), avoiding high computational cost of the eigendecomposition of Laplacian matrix.

In this paper, we present graph wavelet neural network to implement efficient convolution on graph data. We take graph wavelets instead of the eigenvectors of graph Laplacian as a set of basis, and define the convolution operator via wavelet transform and convolution theorem. Graph wavelet neural network distinguishes itself from spectral CNN by its three desirable properties: (1) Compared with graph Fourier transform, graph wavelet transform does not depend on the eigendecomposition of Laplacian matrix, and thus is efficient; (2) Graph wavelets are sparse, while eigenvectors of Laplacian matrix are dense. As a result, graph wavelet transform is much more efficient than graph Fourier transform; (3) Graph wavelets are localized in vertex domain, reflecting the information diffusion

centered at each node (Tremblay & Borgnat, 2014). This property eases the understanding of graph convolution defined by graph wavelets.

We develop an efficient implementation of the proposed graph wavelet neural network. Convolution in conventional CNN learns an individual convolution kernel for each pair of input channel and output channel, causing a huge number of parameters especially when the number of channels is high. We detach the channel transformation from convolution and learn a sole convolution kernel among all channels, substantially reducing the number of parameters. Finally, we validate the effectiveness of the proposed GWNN by applying it to graph semi-supervised classification. Experimental results demonstrate that GWNN consistently outperforms previous spectral CNNs on three benchmark datasets, i.e., Cora, Citeseer, and Pubmed.

2 OUR METHOD

We use graph wavelet transform to implement graph convolution.

2.1 PRELIMINARY

Let $\mathcal{G} = \{\mathbb{V}, \mathbb{E}, \boldsymbol{A}\}$ be an undirected graph, where $\mathbb{V}$ is the set of nodes with $|\mathbb{V}| = n$, $\mathbb{E}$ is the set of edges, and $\boldsymbol{A}$ is adjacency matrix with $A_{i,j} = A_{j,i}$ to define the connection between node i and node j. The graph Laplacian matrix $\boldsymbol{L}$ is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D}$ is a diagonal degree matrix with $D_{i,i} = \sum_j A_{i,j}$, while the normalized Laplacian matrix is $\boldsymbol{L} = \boldsymbol{I}_n - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ where $\boldsymbol{I}_n$ is the identity matrix. Since $\boldsymbol{L}$ is a real symmetric matrix, it has a complete set of orthonormal eigenvectors $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n)$, known as Laplacian eigenvectors. These eigenvectors have associated real, non-negative eigenvalues $\{\lambda_l\}_{l=1}^n$, identified as the frequencies of the graph. Eigenvectors associated with smaller eigenvalues carry slow varying signal, indicating that nodes that are neighbors to share similar values. In contrast, eigenvectors associated with larger eigenvalues carry faster-varying signal across connected nodes.

2.2 GRAPH FOURIER TRANSFORM

Taking the eigenvectors of normalized Laplacian matrix as a set of bases, graph Fourier transform of a signal $\boldsymbol{x} \in R^n$ on graph $\mathcal{G}$ is defined as $\hat{\boldsymbol{x}} = \boldsymbol{U}^\top \boldsymbol{x}$, and the inverse graph Fourier transform is $\boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}$ (Shuman et al., 2013). Graph Fourier transform, according to convolution theorem, offers us a way to define the graph convolution operator, denoted as $*_\mathcal{G}$. Denoting with $\boldsymbol{y}$ the convolution kernel, $*_\mathcal{G}$ is defined as

$$\boldsymbol{x} *_\mathcal{G} \boldsymbol{y} = \boldsymbol{U}\big((\boldsymbol{U}^\top \boldsymbol{y}) \odot (\boldsymbol{U}^\top \boldsymbol{x})\big), \tag{1}$$

where $\odot$ is the element-wise Hadamard product.

However, there are some limitations when using Fourier transform to implement graph convolution: (1) Eigendecomposition of Laplacian matrix to obtain Fourier basis $\boldsymbol{U}$ is of high computational cost with $O(n^3)$; (2) Graph Fourier transform is inefficient, since it involves the multiplication between a dense matrix $\boldsymbol{U}$ and the signal $\boldsymbol{x}$; (3) Graph convolution defined through Fourier transform is not localized in vertex domain, i.e., the influence to the signal on one node is not localized in its neighborhood. To address these limitations, ChebyNet (Defferrard et al., 2016) restricts convolution kernel to the polynomial of diag$(\{\lambda_l\}_{l=1}^n)$. However, such a polynomial approximation also limits the flexibility to define appropriate convolution on graph. Different from ChebyNet, we address the aforementioned three limitations through replacing graph Fourier transform with graph wavelet transform, retaining the flexibility of convolution kernel.

2.3 GRAPH WAVELET TRANSFORM

Similar to graph Fourier transform, graph wavelet transform projects graph signal from vertex domain into spectral domain. Graph wavelet transform employs a set of wavelets as basis, defined as $\psi_s = (\psi_{s1}, \psi_{s2}, ..., \psi_{sn})$, where each wavelet ψ_{si} corresponds to a signal on graph diffused away from node i and s is a scaling parameter. Mathematically, ψ_{si} can be written as

$$\psi_s = \boldsymbol{U}\boldsymbol{G}_s\boldsymbol{U}^\top, \tag{2}$$

where $\boldsymbol{U}$ is Laplacian eigenvectors, $\boldsymbol{G}_s$=diag$\big(g(s\lambda_1), ..., g(s\lambda_n)\big)$ is a scaling matrix and $g(s\lambda_i) = e^{\lambda_i s}$.

Using graph wavelets as basis, graph wavelet transform of a signal $\boldsymbol{x}$ on graph is defined as $\hat{\boldsymbol{x}} = \psi_s^{-1}x$ and the inverse graph wavelet transform is $x = \psi_s\hat{\boldsymbol{x}}$. Note that ψ_s^{-1} can be obtained by simply replacing the $g(s\lambda_i)$ in ψ_s with $g(-s\lambda_i)$ corresponding to a heat kernel (Donnat et al., 2018). Replacing the graph Fourier transform in Equation (1) with graph wavelet transform, we obtain the graph convolution as

$$\boldsymbol{x} *_{\mathcal{G}} \boldsymbol{y} = \psi_s((\psi_s^{-1}\boldsymbol{y}) \odot (\psi_s^{-1}\boldsymbol{x})). \tag{3}$$

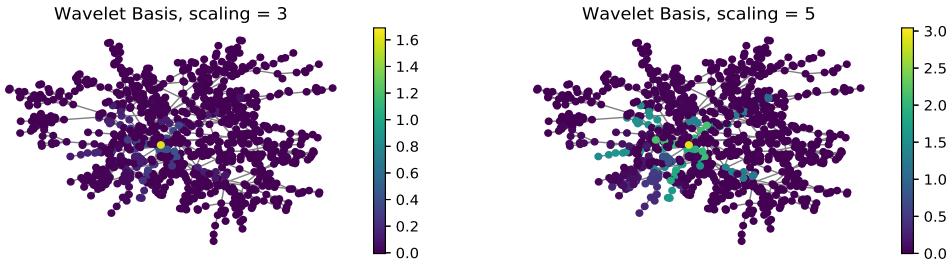
Compared to graph Fourier transform, graph wavelet transform has the following benefits when being used to define graph convolution:

1. Graph wavelets are obtained without requiring the eigendecomposition of Laplacian matrix. In Hammond et al. (2011), a method is proposed to use Chebyshev polynomials to efficiently approximate ψ_s and ψ_s^{-1}.

2. The matrix ψ_s and ψ_s^{-1} are both sparse. Therefore, graph wavelet transform is much more computationally efficient than graph Fourier transform. For example, in the Cora dataset, more than 97% elements in ψ_s^{-1} are zero while only less than 1% elements in $\boldsymbol{U}$ are zero (Table 4). To illustrate the sparseness of the wavelets, we show two wavelets in an example network (Figure 1), depicted using GSP toolbox (Perraudin et al., 2014).

3. Each wavelet corresponds to a signal on graph diffused away from a centered node, highly localized in vertex domain. As a result, the graph convolution defined in Equation (3) is localized in vertex domain. We show the localization property in Appendix A. It is the localization property that explains why graph wavelet transform outperforms Fourier transform in defining graph convolution and the associated tasks like graph semi-supervised learning.

4. Graph wavelets are more flexible to adjust the size of node neighborhood in a continuous manner, i.e., varying the scaling parameter s. A small value of s generally corresponds to a smaller neighborhood. Figure 1 (a) and (b) shows two wavelet bases at different scale.



(a) (b)

Figure 1: (a): Fourier basis. (b): wavelet basis at small scale. (c): wavelet basis at larger scale.

2.4 Graph wavelet neural network

Replacing Fourier transform with wavelet transform, we provide the structure of graph wavelet neural network in this section.

Considering a multi-layer convolutional neural networks in our model, structure of m-th layer is as below:

$$\boldsymbol{X}_{m+1,j} = h(\psi_s \sum_{i=1}^{p} \boldsymbol{F}_{m,i,j}\psi_s^{-1}\boldsymbol{X}_{m,i})(j = 1, ..., q). \tag{4}$$

This layer transforms an input tensor $\boldsymbol{X}_m$ with dimensions $n \times p$ into an output tensor $\boldsymbol{X}_{m+1}$ with dimensions $n \times q$. ψ_s is wavelet bases, ψ_s^{-1} is the graph wavelet transform matrix at scale s which

can project spatial signal into spectral domain. $\boldsymbol{F}_{m,i,j}$ is a diagonal filter matrix learned in spectral domain, and h is a non-linear active function.

3 FURTHER DEVELOPMENT OF OUR METHOD

In many research fields, e.g., semi-supervised learning, the graph has a great many nodes but the total number of labels is few. These tasks request lower parameter complexity and time complexity. Further development of our method focus on: reducing parameter complexity and time complexity.

3.1 REDUCING PARAMETER COMPLEXITY

In order to reduce parameter complexity, we detach the feature transformation from convolution and learn a sole convolution kernel among all features. Details are shown in the following.

In Bruna et al. (2014), the parameter complexity of each layer is $O(n \times p \times q)$, where p is the number of features of each vertex in current layer, and q is the number of features of each vertex in next layer. ChebyNet (Defferrard et al., 2016) approximates convolution with polynomial filter, and the m-th layer structure is

$$\boldsymbol{X}_{m+1,j} = h(\sum_{i=1}^{p} \sum_{k=0}^{K-1} \theta_{i,j,k} \boldsymbol{L}^k \boldsymbol{X}_{m,i})(j = 1, ..., q), \tag{5}$$

where $\theta_{i,j,k}$ is a scalar needs to learn, K is a hyper-parameter and $K \ll n$. This method lowers down the parameter complexity to $O(K \times p \times q)$.

In their experimental datasets, e.g., Minist or 20-groups News, each vertex only has one input feature. However, the number of input features in each vertex is large in some research fileds, e.g, node classification in semi-supervised learning. Specifically, GCN (Kipf & Welling, 2017) shows that the accuracy of ChebyNet on node classification becomes lower when K becomes larger. In order to reduce parameters, they simplified ChebyNet and set $K = 2$ to restrict node relation in first-order neighborhood and the parameter complexity declined to $O(p \times q)$. Restricting the range of neighboring nodes in first-order may restrict the modeling capacity, since second-order and high-order neighborhood may also have contributions to target node.

Another solution is detaching the feature transformation from convolution. All the methods above learn convolution kernel for each pair of input features and output features individually. However, CNNs on graph are different from CNNs on image. With the purpose of capturing node relations, convolution filter on graph simply depends on the graph structure. It is unnecessary to learn filter for each input features and output features with the large increase of parameters. From this perspective, convolution layer can be divided into two components: (1) feature dimension transformation component. This part is implemented by $\boldsymbol{XW}$. (2) convolution component. This part is implemented by wavelet transform and inverse wavelet transform discussed above. In this framework, the parameter complexity can be reduced from $O(n \times p \times q)$ to $O(n + p \times q)$.

In general, the m-th layer of graph wavelet neural network is as below:

$$\boldsymbol{X}_{m,df} = \boldsymbol{X}_m \boldsymbol{W}, \tag{6}$$

$$\boldsymbol{X}_{m+1} = h(\psi_s \boldsymbol{F}_m \psi_s^{-1} \boldsymbol{X}_{m,df}). \tag{7}$$

This layer transforms an input tensor $\boldsymbol{X}_m$ with $n \times p$ dimensions into an output tensor $\boldsymbol{X}_{m+1}$ with $n \times q$ dimensions. $\boldsymbol{W} \in \mathbb{R}^{p \times q}$ is used to learn feature dimension transformation parameters, $\boldsymbol{X}_{m,df}$ is the feature after dimension transform, $\boldsymbol{F}_m$ is the diagonal matrix to learn convolution filter, h is a non-linear active function.

3.2 REDUCING TIME COMPLEXITY

As introduced in Section 2.3, ψ_s^{-1} is a very sparse matrix, the complexity of multiplication between sparse matrix and signal matrix is O($d \times n^2$), d in our experiments is smaller than 5%. However,

calculating ψ_s and ψ_s^{-1} from the eigendecomposition of Laplacian is of high time complexity $O(n^3)$. Hammond et al. (2011) proposed an efficient way to approximate ψ_s and ψ_s^{-1} with Chebyshev polynomials directly. They proved that the computational complexity of graph wavelet transform is $O(m \times |\mathbb{E}|)$, where $|\mathbb{E}|$ is the number of edges, m is the expansion order of Chebyshev polynomials.

4 RELATED WORKS

Graph convolutional neural networks on graphs. The success of CNNs when dealing with images, videos, and speeches motivates researchers to design graph convolutional neural network on graphs. The key of generalizing CNNs to graphs is defining convolution operator on graphs. Existing methods are classified into two categories, i.e., spectral methods and spatial methods.

Spectral methods define convolution via convolution theorem. Spectral CNN (Bruna et al., 2014) is the first attempt at implementing CNNs on graphs, though leveraging graph Fourier transform and defining convolution kernel in spectral domain. Defferrard et al. (2016) introduced a Chebyshev polynomial parametrization for spectral filter, offering us a fast localized spectral filtering method. Kipf & Welling (2017) provided a simplified version of ChebyNet, gaining success in graph semi-supervised learning task.

Spatial methods define convolution as a weighted average function over neighborhood of target vertex. GraphSAGE takes one-hop neighbor as neighborhood and defines the weighting function as various aggregators over neighborhood (Hamilton et al., 2017). Graph attention network (GAT) proposes to learn the weighting function via self-attention mechanism (Velickovic et al., 2017). MoNet offers us a general framework for design spatial methods, taking convolution as the weighted average of multiple weighting functions defined over neighborhood (Monti et al., 2017).

Graph wavelets. Hammond et al. (2011) proposed a method to construct wavelet transform on graphs, Moreover, they designed an efficient way to bypass the eigendecomposition of the Laplacian and approximated wavelets with Chebyshev polynomials. Tremblay & Borgnat (2014) leveraged graph wavelets for multi-scale community mining by the modulating scaling parameter s. Owing to the property of describing information diffusion, Donnat et al. (2018) learned structural node embeddings via wavelets. All their works prove that graph wavelets are not only local and sparse but also meaningful.

5 EXPERIMENTS

5.1 DATASETS AND BASELINES

To evaluate the proposed graph wavelet neural network (GWNN), we conduct experiments on three benchmark datasets, namely, Cora, Citeseer and Pubmed (Sen et al., 2008). In these citation network datasets, nodes represent documents and edges are citation links. Details of these datasets are demonstrated in Table 1. Here, the label rate denotes the proportion of labeled nodes used for training. Following the experimental setup of GCN (Kipf & Welling, 2017) closely, we fetch 20 labeled nodes per class in each dataset to train the model.

Table 1: The Statistics of Datasets

Dataset	Nodes	Edges	Classes	Features	Label Rate
Cora	2,708	5,429	7	1,433	0.052
Citeseer	3,327	4,732	6	3,703	0.036
Pubmed	19,717	44,338	3	500	0.003

We compare with the same baselines as those in Kipf & Welling (2017), including label propagation (LP) (Zhu et al., 2003), semi-supervised embedding (SemiEmb) (Weston et al., 2012), manifold regularization (ManiReg) (Belkin et al., 2006), graph embeddings (DeepWalk) (Perozzi et al., 2014), iterative classification algorithm (ICA) (Lu & Getoor, 2003) and Planetoid (Yang et al., 2016).

Furthermore, along with the development of deep learning on graph, graph convolutional networks is proved to be effective in semi-supervised learning. Thus, we also compare against Spectral

CNN (Bruna et al., 2014), ChebyNet (Defferrard et al., 2016), GCN (Kipf & Welling, 2017) and Monet (Monti et al., 2017).

5.2 Experimental Settings

We train a two-layer graph wavelet neural network with 16 hidden units, and prediction accuracy is evaluated on a test set of 1000 labeled samples. The partition of datasets is the same as GCN (Kipf & Welling, 2017) with an additional validation set of 500 labeled samples to determine hyper-parameters.

Weights are initialized following Glorot & Bengio (2010). We adopt the Adam optimizer (Kingma & Ba, 2014) for parameter optimization with an initial learning rate $lr = 0.01$. For each dataset, if $|\psi_{s[i,j]}|$ and $|\psi^{-1}_{s[i,j]}|$ is smaller than a given threshold t, we set it zero to accelerate the computation. The optimal hyper-parameters, e.g., scaling parameter s and threshold t, are different for each dataset to suit different network. For Cora, $s = 1.0$ and $t = 1e - 4$. For Citeseer, $s = 0.7$ and $t = 1e - 5$. For Pubmed, $s = 0.5$ and $t = 1e - 7$. To avoid overfitting, dropout (Srivastava et al., 2014) is applied and the value is set to 0.5. Meanwhile, the training will be terminated if the validation loss does not decrease for 100 consecutive epochs.

5.3 Analysis on detaching feature transformation from convolution

Since the parameters of undetached version of GWNN is $O(n \times p \times q)$, we can hardly implement this version in the case of networks with large number of nodes n and huge number of input channels p. As a result, we validate the effectiveness of detaching on ChebyNet introduced in Section 3.1, whose parameter complexity is $O(K \times p \times q)$. Instead of learning the individual filter for each pair of input feature and output feature, we detach feature transformation from convolution to reduce the number of parameters significantly. Table 2 shows the performance and parameter number of each method. Here, we report the maximum performance for filters of order $K = 2, 3, 4$.

Table 2: Results of Detaching

	Method	**Cora**	**Citeseer**	**Pubmed**
Prediction Accuracy	ChebyNet	81.2%	**69.8%**	74.4%
	Detaching-ChebyNet	**81.6%**	68.5%	**78.6%**
Parameter Num	ChebyNet	46,080	178,080	24,336
	Detaching-ChebyNet	23,048	59,428	8,118

Reducing the number of parameters may restrict the modeling capacity to some degree. However, as demonstrated in Table 2, with fewer parameters, we improve the accuracy on Pubmed by a large margin. This may due to that the label rate of Pubmed is only 0.003. By detaching feature transformation from convolution, the parameter complexity significantly reduced, avoiding overfitting in semi-supervised learning and thus greatly improving prediction accuracy.

Table 3: Results of Node Classification

Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg	59.5%	60.1%	70.7%
SemiEmb	59.0%	59.6%	71.7%
LP	68.0%	45.3%	63.0%
DeepWalk	67.2%	43.2%	65.3%
ICA	75.1%	69.1%	73.9%
Planetoid	75.7%	64.7%	77.2%
Spectral CNN	73.3%	58.9%	73.9%
ChebyNet	81.2%	69.8%	74.4%
GCN	81.5%	70.3%	79.0%
MoNet	81.7±0.5%	—	78.8±0.3%
GWNN	**82.8%**	**71.7%**	**79.1%**

5.4 PERFORMANCE OF GWNN

We now validate the effectiveness of GWNN with detach technique on node classification. Experimental results are reported in Table 3. As is shown, GWNN improves the classification accuracy on all the three datasets. Through replacing Fourier transform with wavelet transform, the proposed GWNN is comfortably ahead of Spectral CNN. The large improvement could be explained from two perspectives: (1) Convolution in Spectral CNN is non-local in vertex domain, and as a result, the range of feature diffusion is not restricted to neighboring nodes. Thus, the performance gets limited; (2) the scaling parameter s of wavelet transform can be used to adjust the diffusion range to suit different applications and different networks. GWNN also outperform ChebyNet, due to that GWNN holds the fllexibility of convolution kernel.

5.5 ANALYSIS ON SPARSITY

Besides improvement on prediction accuracy, wavelet transform with localized and sparse transform matrix holds sparsity in both spatial domain and spectral domain. Here, we will show the sparsity in Cora.

The sparsity of transform matrix. There are 2,708 nodes in Cora. Thus, the wavelet transform matrix ψ_s and the Fourier transform matrix $\boldsymbol{U}^\top$ both belong to $\mathbb{R}^{2,708\times 2,708}$. The first two rows in Table 4 demonstrate that ψ_s is much sparser than $\boldsymbol{U}^\top$. With lower density, wavelet basis can not only accelerate the computation, but also describe the neighboring topology centered at each node.

The sparsity of projected signal. As mentioned above, each node in Cora represents a document and has a sparse bag-of-words feature. The input feature $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is a binary matrix, and $\boldsymbol{X}_{i,j} = 1$ when the i-th document contains the j-th word in the bag of words, otherwise, it equals 0. Here, $X_{:,j}$ denotes the j-th column of $\boldsymbol{X}$, and it represents the feature vector of a word. Considering a specific signal $X_{:,984}$, we project the spatial signal into spectral domain with domain transform, and get the projected vector of each basis matrix. Here, $\boldsymbol{p} = \psi_s^{-1} X_{:,984}$ denotes the projected vector of wavelet basis, $\boldsymbol{q} = \boldsymbol{U}^\top X_{:,984}$ denotes the projected vector of Fourier basis, and $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^{2,708}$. The last row in Table 4 lists the numbers of non-zero elements in $\boldsymbol{p}$ and $\boldsymbol{q}$. As is shown, with wavelet transform, the projected signal is much sparser.

Table 4: Statistics of wavelet transform and Fourier transform

	Statistical Property	**wavelet transform**	**Fourier transform**
Transform Matrix	Density	2.8%	99.1%
	Num of Non-zero Elements	205,774	7,274,383
Projected Signal	Density	10.9%	100%
	Num of Non-zero Elements	297	2,708

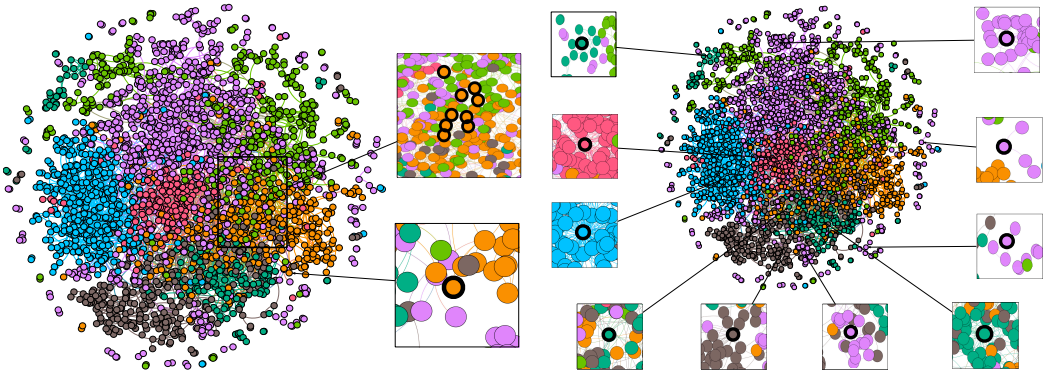
5.6 ANALYSIS ON INTERPRETABILITY

GWNN also provides interpretable domain transformation, we will show the interpretability with specific examples in Cora as follows.

Each spectral wavelet basis is centered at a node. In other words, each feature, i.e. word in the bag of words, has a value in the projected vector with each basis. Here each basis corresponds to a document. The value can be regarded as the relation between the word and the document. Thus, each value in $\boldsymbol{p}$ can be interpreted as the relation between $Word_{984}$ and a document. However, it is hard to explain $\boldsymbol{q}$ due to its high density and non-local of $\boldsymbol{U}^\top$. In order to elaborate the interpretability of wavelet transform, we analyze the projected values of different feature as following.

Considering two features $Word_{984}$ and $Word_{1177}$, we select the top-10 active bases, which have the 10 largest projected values of each feature. As illustrated in Figure 2, to present clearly, we magnify the local structure of corresponding nodes and marked them with bold rims. The central network in each subgraph denotes the dataset Cora, and each node represents a document. And 7 different colors represent 7 classes. These nodes are clustered by OpenOrd (Martin et al., 2011) based on the adjacency matrix.

Figure 2a shows the top-10 active bases of $Word_{984}$. In Cora, this word only appears 8 times, and all the documents containing $Word_{984}$ belong to the class " Case-Based ". Consistently, all top-10 nodes activated by $Word_{984}$ are concentrated and belong to the class " Case-Based ". And, the frequencies of $Word_{1177}$ appearing in different classes are similar, indicating that $Word_{1177}$ is a universal word. In concordance with our expectation, the top-10 active bases of $Word_{1177}$ are discrete and belong to different classes in Figure 2b.



(a) (b)

Figure 2: Top-10 active bases of two words in Cora. The central network of each subgraph represents the dataset Cora, which is split into 7 classes. Each node represents a document, and its color indicates its label. The nodes that represent the top-10 active bases are marked with bold rims. $Word_{984}$ only appears in documents of the class " Case-Based " in Cora. And all its 10 active bases also belong to the class " Case-Based ". The frequencies of $Word_{1177}$ appearing in different classes are similar in Cora. Consistent with inspection, the top-10 active bases of $Word_{1177}$ also belong to different classes.

The sparsity and locality of wavelet transform lower down the computation complexity. Also, The projected values of wavelet transform can be explained as the correlation between features and nodes. However, this is not straightforward for graph Fourier basis. With the dense and non-local Fourier basis, it is hard to explain the dense projected vector.

6 Conclusion and Future Work

Replacing graph Fourier transform with wavelet transform, We proposed GWNN. Graph wavelet transform has three desirable properties: (1) bases and transform matrix are both local and sparse; (2) scaling parameter can modulate the range of feature diffusion; (3) Convolution is localized in vertex domain while retaining flexibility of kernel. These advantages make the whole learning process become more interpretable and efficient. Additionally, we get better experimental results than previous methods. On the other hand, our work focuses on dealing with semi-supervised learning problems, which request lower parameter complexity. We put forward the architecture that detaching the feature transformation from convolution.

However, though we have reduced the parameter complexity by a large margin, it also depends on the number of node. We will investigate how to reduce complexity of parameter further in future.

References

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7 (Nov):2399–2434, 2006.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 496–503, 2003.

Shawn Martin, W Michael Brown, Richard Klavans, and Kevin W Boyack. Openord: an open-source toolbox for large graph layout. In *Visualization and Data Analysis 2011*, volume 7868, pp. 786806. International Society for Optics and Photonics, 2011.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, pp. 3, 2017.

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

Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K Hammond. Gspbox: A toolbox for signal processing on graphs. *arXiv preprint arXiv:1408.5781*, 2014.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Nicolas Tremblay and Pierre Borgnat. Graph wavelets for multiscale community mining. *IEEE Transactions on Signal Processing*, 62(20):5227–5239, 2014.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pp. 639–655. Springer, 2012.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48, 2016.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

APPENDIX A LOCALIZED GRAPH CONVOLUTION VIA WAVELET TRANSFORM

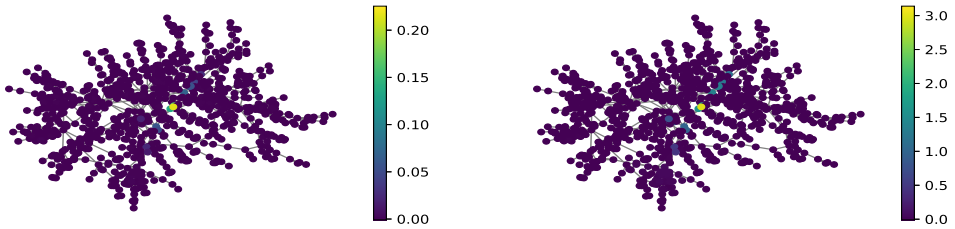
We use a diagonal matrix Θ to represent learned kernel transformed by wavelets $\psi_s^{-1}\boldsymbol{y}$, and replace the Hadamard product with matrix muplication. Then Equation 3 is a below:

$$\boldsymbol{x} *_{\mathcal{G}} \boldsymbol{y} = \psi_s \Theta \psi_s^{-1} \boldsymbol{x}. \tag{8}$$

We set $\psi_s = (\psi_{s1}, \psi_{s2}, ..., \psi_{sn})$, $\psi_s^{-1} = (\psi_{s1}^*, \psi_{s2}^*, ..., \psi_{sn}^*)$, $\Theta = diag(\{\theta_k\}_{k=1}^n)$. Since ψ_s is a symmetric matrix, Equation 8 is :

$$\boldsymbol{x} *_{\mathcal{G}} \boldsymbol{y} = \sum_{k=1}^{n} \theta_k \psi_{sk} (\psi_{sk}^*)^\top \boldsymbol{x}. \tag{9}$$

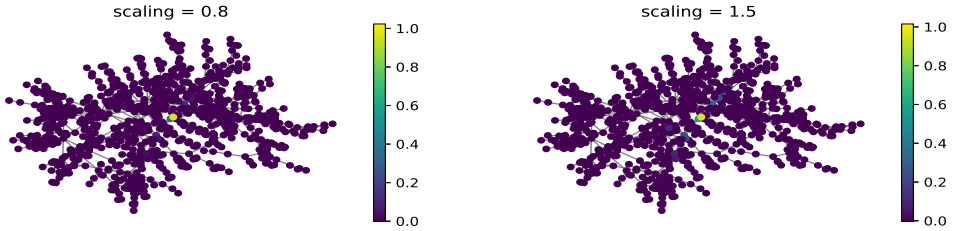
As proved by Hammond et al. (2011), wavelets provide locality in spectral domain, which means, ψ_s and ψ_s^{-1} are both local in small scale. Figure 3 shows the locality of ψ_{s1} and ψ_{s1}^*, i.e., the first column in ψ_s and ψ_s^{-1} when $s = 3$. Each column in ψ_s and ψ_s^{-1} describes the neighboring topology of target node, which means that ψ_s and ψ_s^{-1} are local. The locality of ψ_{sk} and ψ_{sk}^* leads to the locality of the resulting matrix of multiplication between the column vector ψ_{sk} and row vector $(\psi_{sk}^*)^\top$. We set $\boldsymbol{M}_k = \psi_{sk}(\psi_{sk}^*)^\top$ only when $(\psi_{sk}^*)^\top(i) > 0$ and $\psi_{sk}(j) > 0$. And then $M_{k[i,j]} > 0$. In other words, if $\boldsymbol{M}_{k[i,j]} > 0$, vertex i and vertex j have correlation through vertex k.



(a) (b)

Figure 3: (a): ψ_{s1} (b) ψ_{s1}^*. Both are local describing the neighboring topology around target node.

Since each $\boldsymbol{M}_k$ is local, for any convolution kernel Θ, $\psi_s \Theta \psi_s^{-1}$ is local, and it means that convolution is localized in vertex domain. By replacing Θ with an identity matrix in Equation 9, we get $\boldsymbol{x} *_{\mathcal{G}} \boldsymbol{y} = \sum_{k=1}^{n} \boldsymbol{M}_k \boldsymbol{x}$. We define $\boldsymbol{H} = \sum_{k=1}^{n} \boldsymbol{M}_k$, and Figure 4 shows $H_{1,:}$ in different scaling, i.e., correlation between the first node and other nodes during convolution. The locality of $\boldsymbol{H}$ suggests that graph convolution is localized in vertex domain. Moreover, with scaling parameter s becoming larger, the range of feature diffusion becomes larger.



(a) (b)

Figure 4: (a): Related nodes at small scale. (b): Related nodes at larger scale. Non-zero value of node represents correlation between this node and target node during convolution. Locality of $\boldsymbol{H}$ suggests that graph convolution is localized in vertex domain. Moreover, with scaling parameter s becoming larger, the range of feature diffusion becomes larger.