Spectral Graph Attention Network with Fast Eigen-approximation

Heng Chang*
TBSI, Tsinghua University
changh17@mails.tsinghua.edu.cn

Yu Rong
Tencent AI Lab
yu.rong@hotmail.com

Tingyang Xu
Tencent AI Lab
Tingyangxu@tencent.com

Wenbing Huang
Tsinghua University
hwenbing@126.com

Somayeh Sojoudi
University of California at Berkeley
sojoudi@berkeley.edu

Junzhou Huang
University of Texas at Arlington
jzhuang@uta.edu

Wenwu Zhu
Tsinghua University
wwzhu@tsinghua.edu.cn

ABSTRACT

Variants of Graph Neural Networks (GNNs) for representation learning have been proposed recently and achieved fruitful results in various fields. Among them, Graph Attention Network (GAT) first employs a self-attention strategy to learn attention weights for each edge in the spatial domain. However, learning the attentions over edges can only focus on the local information of graphs and greatly increases the computational costs. In this paper, we first introduce the attention mechanism in the spectral domain of graphs and present **Spectral Graph Attention Network** (SpGAT) that learns representations for different frequency components regarding weighted filters and graph wavelets bases. In this way, SpGAT can better capture global patterns of graphs in an efficient manner with much fewer learned parameters than that of GAT. Further, to reduce the computational cost of SpGAT brought by the eigen-decomposition, we propose a fast approximation variant SpGAT-Cheby. We thoroughly evaluate the performance of SpGAT and SpGAT-Cheby in semi-supervised node classification tasks and verify the effectiveness of the learned attentions in spectral domain.

CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning settings**; **Neural networks**.

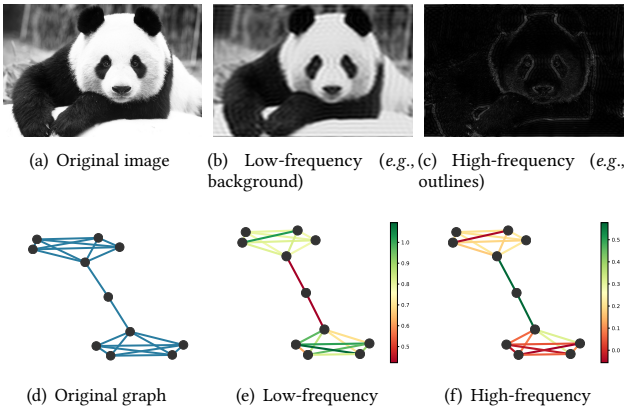
KEYWORDS

graph representation learning, graph spectral analysis, graph neural networks, deep learning

*Heng Chang is supported by 2020 Tencent Rhino-Bird Elite Training Program.

DLG-KDD '21, August 14-18, 2021, Virtual Conference

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456



(a) Original image (b) Low-frequency (*e.g.*, background) (c) High-frequency (*e.g.*, outlines)

(d) Original graph (e) Low-frequency (f) High-frequency

Figure 1: Motivation: Separating the low- and high-frequency signals in both image and graph contributes to the feature learning. Color bars in graphs indicate the measurement of reconstructed edge weights.

ACM Reference Format:
Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Somayeh Sojoudi, Junzhou Huang, and Wenwu Zhu. 2021. Spectral Graph Attention Network with Fast Eigen-approximation. In *DLG-KDD '21: The Sixth International Workshop on Deep Learning on Graphs: Methods and Applications, August 14-18, 2021, Virtual Conference*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Graph Neural Networks (GNNs) [29] aim at imitating the expressive capability of deep neural networks from grid-like data (*e.g.*, images and sequences) to graph structures. The fruitful progress of GNNs in the past decade has made them a crucial kind of tools for a variety of applications, from social networks [15], computer vision [34], text classification [33], to chemistry [17].

Graph Attention Network (GAT) [26], as one central type of GNNs introduces the attention mechanism to further refine the convolution process in generic GCNs [13]. GAT, along with its variants [8, 27, 31, 35, 36], considers the attention in a straightforward way: learning the edge attentions in the spatial domain. In this sense, this attention can capture the local structure of graphs, *i.e.*,

the information from neighbors. However, it is unable to explicitly encode the global structure of graphs. Furthermore, computing the attention weight for every edge in graphs is inefficient, especially for large graphs.

In computer vision, a natural image can be decomposed into a low spatial frequency component containing the smoothly changing structure, *e.g.*, background, and a high spatial frequency component describing the rapidly changing fine details, *e.g.*, outlines [4]. Figure 1(a) ~ 1(c) depict the example of low- and high-frequency components on a panda image. Obviously, the contribution of different frequencies varies with respect to different downstream tasks.

Similar pattern can be observed more naturally in graphs. According to graph signal processing (GSP), we can directly divide the low- and high-frequency components based on the ascending ordered eigenvalues of Laplacian in graphs. The eigenvectors associated with small eigenvalues carry smoothly varying signals, encouraging neighbor nodes to share similar values (local information). In contrast, the eigenvectors associated with large eigenvalues carry sharply varying signals across edges (global information) [7, 18]. As demonstrated in Figure 1(d) ~ 1(f), a barbell graph tends to retain the information inside the clusters when it is reconstructed with only low-frequency components(1(e)), but reserve knowledge between the clusters when constructed with only high-frequency ones (1(f)). Moreover, recent works [7, 18] also reveal the different contributions of low- and high-frequency components in graphs to the learning of modern GNNs.

In this paper, to model the importance of low- and high-frequency components in graphs, we propose to extend the attention mechanism to the spectral domain. In this way, we can explicitly encode the structural information of graphs from a global perspective. Accordingly, we present Spectral Graph Attention Network (SpGAT). In SpGAT, we choose the graph wavelets as the spectral bases and decompose them into low- and high-frequency components with respect to their indices. Then we construct two distinct convolutional kernels according to the low- and high-frequency components and apply the attention mechanism on both kernels to capture their importance respectively. Finally, an pooling function as well as an activation function are applied to produce the output. Figure 2 provides an overview of the design of SpGAT. Furthermore, we employ the Chebyshev polynomial approximation to compute the spectral wavelets of graphs and propose an variant SpGAT-Cheby, which is more efficient on large graphs. We thoroughly validate the performance of SpGAT and SpGAT-Cheby on five benchmarks with fourteen competitive baselines. SpGAT and SpGAT-Cheby achieve state-of-the-art results on all of the datasets.

2 THE PROPOSED FRAMEWORK

We denote $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as an undirected graph, where $|\mathcal{V}| = n$ is the set of n nodes, and $\mathcal{E}$ is the set of edges, where $(v_i, v_j) \in \mathcal{E}$. The adjacency matrix is defined as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, where $\boldsymbol{A}_{ij} = 1$ indicates an edge (v_i, v_j). We denote $\boldsymbol{D}$ as the degree matrix. $\hat{\boldsymbol{A}} = \hat{\boldsymbol{D}}^{-1/2}(\boldsymbol{A}+\boldsymbol{I})\hat{\boldsymbol{D}}^{-1/2}$ refers to the normalized adjacency matrix with self-loop, where $\boldsymbol{I}$ is the identity matrix and $\hat{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}$.

From the spatial perspective, GNNs can usually be viewed as the feature aggregation among the neighbors of nodes in the spatial domain of graphs. Therefore, we write the feed-forward layer of

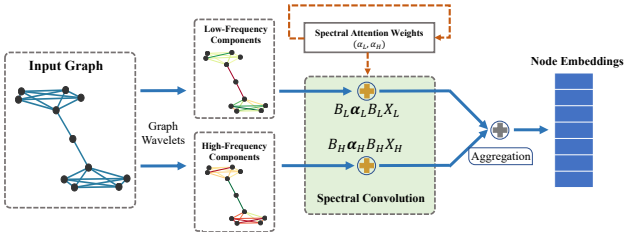


Figure 2: Overview of proposed Spectral Graph Attention Network(SpGAT). Two distinct convolutional kernels according to the low- and high-frequency components are constructed and the attention mechanism is employed on both kernels to capture the importance, respectively.

GNNs in a general form:

$$\boldsymbol{h}'_i = \sigma(\mathsf{AGG}_{j\in\mathcal{N}_i}(\alpha_{ij}\boldsymbol{\Theta}\boldsymbol{h}_j)), \tag{1}$$

where $\mathcal{N}_i$ refers to the neighborhood set of node i in graph. $\boldsymbol{H}' = \{\boldsymbol{h}'^{\mathrm{T}}_1, \cdots, \boldsymbol{h}'^{\mathrm{T}}_n\}$ are the output of hidden vectors from the layer with $\boldsymbol{H} = \{\boldsymbol{h}^{\mathrm{T}}_1, \cdots, \boldsymbol{h}^{\mathrm{T}}_n\}$ as the input features. $\sigma(\cdot)$ refers to the activation function, such as ReLU. $\boldsymbol{\Theta}^{\mathrm{T}} \in \mathbb{R}^{p\times q}$ refers to the learning parameters of the layer, where p and q refer to the feature dimensions of input and output, respectively. α_{ij} refers to the aggregation weight of neighbor j for node i. $\mathsf{AGG}(\cdot)$ refers to the aggregation function that aggregates the output of each neighbor, such as SUM and MEAN. For examples, VanillaGCN can be viewed as the special case of Eq. (1) where $\alpha_{ij} = \hat{\boldsymbol{A}}_{ij}$, and $\mathsf{AGG}(\cdot) = \mathsf{SUM}(\cdot)$. Meanwhile, GAT proposes to compute the weight α_{ij} by a self-attention strategy and uses SUM as aggregation function.

Other than the neighbor aggregation in the spatial domain [13], VanillaGCN can also be understood from the perspective of GSP in the spectral domain:

$$g_\theta \star \boldsymbol{x} = \boldsymbol{B}g_\theta\boldsymbol{B}^{\mathrm{T}}\boldsymbol{x}, \tag{2}$$

where $\boldsymbol{x}$ is a signal on every node. $\boldsymbol{B} = \{\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n\}$ are the spectral bases extracted from the graph. $g_\theta = \mathrm{diag}(\theta)$ is a diagonal filter parameterized by θ. Given Eq. (2), VanillaGCN can be viewed as the spectral graph convolution based on the Fourier transformation on graphs with the first-order Chebyshev polynomial approximations [13]. Further, we can separate the spectral graph convolution into two stages [30]:

$$\begin{aligned} \text{feature transformation} &: \boldsymbol{X} = \boldsymbol{H}\boldsymbol{\Theta}^{\mathrm{T}}, \\ \text{graph convolution} &: \boldsymbol{H}' = \sigma(\boldsymbol{B}\boldsymbol{F}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{X}). \end{aligned} \tag{3}$$

In Eq. (3), $\boldsymbol{F}$ is a diagonal matrix for the kernel of graph convolution. For instance, the convolutional kernel for VanillaGCN is $\boldsymbol{F} = \mathrm{diag}(\lambda_1, \cdots, \lambda_n)$, where $\{\lambda_i\}_{i=1}^n$ are the eigenvalues of the normalized Laplacian $\boldsymbol{L} = \boldsymbol{I} - \hat{\boldsymbol{A}}$ in ascending order, while the spectral bases $\boldsymbol{B}$ for VanillaGCN are the corresponding eigenvectors.

2.1 The Construction of SpGAT Layer

In this section, we start to describe the construction of SpGAT layer. From the perspective of GSP, the diagonal values $(f_1, \cdots, f_n)$ on $\boldsymbol{F}$ can be treated as the **frequencies** on graphs when they equal to the eigenvalues. We denote the diagonal values with small / large indices as the low / high frequencies, respectively. Meanwhile,

the corresponding spectral bases in $\boldsymbol{B}$ are low- and high-frequency components. As discussed in Section 1, the low- and high-frequency components carry different structural information in graphs. In this vein, we first propose to split the spectral bases into two groups and re-write Eq. (3) as follows:

$$\begin{aligned} &\boldsymbol{X_L} = \boldsymbol{H}\boldsymbol{\Theta}_L^{\mathrm{T}}, \boldsymbol{X}_H = \boldsymbol{H}\boldsymbol{\Theta}_H^{\mathrm{T}}, \\ &\boldsymbol{H'} = \sigma\left(\mathsf{AGG}(\boldsymbol{B}_L\boldsymbol{F}_L\boldsymbol{B}_L^{\mathrm{T}}\boldsymbol{X}_L, \boldsymbol{B}_H\boldsymbol{F}_H\boldsymbol{B}_H^{\mathrm{T}}\boldsymbol{X}_H)\right), \end{aligned} \tag{4}$$

where $\boldsymbol{B}_L = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_d)$ and $\boldsymbol{B}_H = (\boldsymbol{b}_{d+1}, \cdots, \boldsymbol{b}_n)$ are the low- and high-frequency components, respectively. Here d is a hyper-parameter that determines the splitting boundary of low- and high-frequency. When $\mathsf{AGG}(\cdot) = \mathsf{SUM}(\cdot)$, Eq. (4) is equivalent to the graph convolution stage in Eq. (3).

In Eq. (4), $\boldsymbol{F}_L$ and $\boldsymbol{F}_H$ can also be viewed as the importances of the low- and high-frequency. Therefore, we introduce the learnable attention weights by exploiting the *re-parameterization* trick:

$$\boldsymbol{H'} = \sigma\left(\mathsf{AGG}(\boldsymbol{B}_L\boldsymbol{\alpha}_L\boldsymbol{B}_L^{\mathrm{T}}\boldsymbol{X}_L, \boldsymbol{B}_H\boldsymbol{\alpha}_H\boldsymbol{B}_H^{\mathrm{T}}\boldsymbol{X}_H)\right). \tag{5}$$

In Eq. (5), $\boldsymbol{\alpha}_L = \mathrm{diag}(\alpha_L, \cdots, \alpha_L)$ and $\boldsymbol{\alpha}_H = \mathrm{diag}(\alpha_H, \cdots, \alpha_H)$ are parameterized by two learnable weights α_L and α_H, respectively. To ensure α_L and α_H are positive and comparable, we normalize them by the $\mathrm{softmax}$ function in an attention manner:

$$\alpha_* = \mathrm{softmax}(\alpha_*) = \frac{\exp(\alpha_*)}{\sum_* \exp(\alpha_*)}, \quad * = L, H.$$

Theoretically, there are many approaches to re-parameterize $\boldsymbol{\alpha}_L$ and $\boldsymbol{\alpha}_H$, such as self-attention w.r.t the spectral basis $\boldsymbol{b}_i$. However, these kinds of re-parameterization can not reflect the nature of low- and high-frequency components. On the other hand, they may introduce too many additional learnable parameters, especially for large graphs. These parameters might prohibit the efficient training due to the limited amount of training data in graph learning, especially under the graph-based semi-supervised setting.

2.2 Choice of Spectral Bases

Another important design is the choice of the spectral basis. Instead of Fourier bases, we choose graph wavelets as spectral bases in SpGAT following the observation on the advantages of spectral wavelets in recent works [7, 30]. Formally, the wavelet on a graph $\psi_{si}(\lambda)$ is defined as the signal resulting from the modulation in the spectral domain of a signal $\boldsymbol{x}$ centered around the associated node i [11, 24]. Then, given a graph G, the graph wavelet transformation is conducted by employing a set of wavelets $\boldsymbol{\Psi}_s = (\psi_{s1}(\lambda_1), \psi_{s2}(\lambda_2), \ldots, \psi_{sn}(\lambda_n))$ as bases:

$$\boldsymbol{\Psi}_s(\lambda) = \boldsymbol{U}g_s(\lambda)\boldsymbol{U}^{\mathrm{T}}, \tag{6}$$

where $\boldsymbol{U}$ is the eigenvectors of the normalized Laplacian $\boldsymbol{L} = \boldsymbol{I} - \hat{\boldsymbol{A}}$. $g_s(\lambda) = \mathrm{diag}\big(g_s(\lambda_1), g_s(\lambda_2), \ldots, g_s(\lambda_n)\big)$ is a scaling matrix with heat kernel scaled by hyperparameter s. The inverse of graph wavelets $\boldsymbol{\Psi}_s^{-1}(\lambda)$ is obtained by simply replacing $g_s(\lambda)$ with $g_s(-\lambda)$ [7]. Smaller indices in graph wavelets correspond to low-frequency components and vice versa. Overall, the architecture of SpGAT layer with graph wavelet $\boldsymbol{\Psi}_s$ as bases can be written as:

$$\boldsymbol{X} = \boldsymbol{H}\boldsymbol{\Theta}^{\mathrm{T}}, \quad \boldsymbol{H'} = \sigma\left(\mathsf{AGG}(\boldsymbol{\Psi}_{sL}\boldsymbol{\alpha}_L\boldsymbol{\Psi}_{sL}^{-1}\boldsymbol{X}, \boldsymbol{\Psi}_{sH}\boldsymbol{\alpha}_H\boldsymbol{\Psi}_{sH}^{-1}\boldsymbol{X})\right). \tag{7}$$

2.3 Parameter Complexity of SpGAT

In Eq. (7), aiming to further reduce the parameter complexity, we share the parameters in feature transformation stage for $\boldsymbol{X}_L$ and $\boldsymbol{X}_H$, *i.e.*, $\boldsymbol{\Theta}_L = \boldsymbol{\Theta}_H$. In this way, we reduce the parameter complexity from $O(2 \times (p \times q + 1))$ to $O(p \times q + 2)$, which is nearly the same as VanillaGCN, which is $O(p \times q)$. The parameter complexity of SpGAT is much less than that of GAT with K-head attention, which is $O((p + 2) \times q \times K)$. Comparing with GAT, which captures the local structure of graphs from spatial domain, our proposed SpGAT could better tackle global information by combining the low- and high-frequency features explicitly from spectral domain.

3 FAST APPROXIMATION OF SPGAT

In SpGAT, directly computing the transformation according to Eq. (6) is intensive for large graphs, since diagonalizing Laplacian $\boldsymbol{L}$ commonly requires $O(n^3)$ computational complexity. Fortunately, we can employ the Chebyshev polynomials to fast approximate the spectral graph wavelets without eigen-decomposition[11].

Theorem 1. *Let s be the scaling parameter in the heat kernel $g_s(\lambda) = e^{-\lambda s}$, and M be the degree of the Chebyshev polynomial approximation for the scaled wavelet (larger value of M yields more accurate approximation but higher computational cost in the opposite), then the graph wavelet is given by*

$$\boldsymbol{\Psi}_s(\lambda) = \frac{1}{2}c_{0,s} + \sum_{i=1}^{M} c_{i,s}T_i(\tilde{\boldsymbol{L}}), \; c_{i,s} = 2e^s J_i(s), \tag{8}$$

where $\tilde{\boldsymbol{L}} = \frac{2}{\lambda_{max}}\boldsymbol{L} - \boldsymbol{I}$, $T_i(\tilde{\boldsymbol{L}})$ is the i_{th} order Chebyshev polynomial, and $J_i(s)$ is the modified Bessel function of the first kind.

Theorem 1 can be derived from Section 6 in [11]. It should be noted that though [30] discusses the possibility to bring the method from [11] into approximating wavelets but with integral operations, we make the first attempt to integrate Theorem 1 into practice. Moreover, to accelerate the computation, we build a look-up table for the Bessel function $J_i(s)$ to avoid additional integral operations.

With this Chebyshev polynomial approximation, the computational cost of the spectral graph wavelet is decreased to $O(M\|\mathcal{E}\| + Mn)$, where $\|\mathcal{E}\|$ is the total number of edges. Due to the real world graphs are usually sparse, this computational reduction can be very significant. We denote SpGAT with Chebyshev polynomial approximation as SpGAT-Cheby. As for SpGAT-Cheby, instead of using eigen-decomposition, we directly employ Eq.(8) to speed up the computation of the spectral wavelets $\boldsymbol{\Psi}_s(\lambda)$. After that, the approximated $\boldsymbol{\Psi}_s(\lambda)$ are seamlessly fed into the original SpGAT.

4 EXPERIMENTS

4.1 Experimental Setup

Joining the practice of previous works, we mainly focus on five node classification benchmarks under semi-supervised setting with different graph size, feature type and public splitting, including three citation networks: Citeseer, Cora and Pubmed [22], a coauthor network: Coauthor CS, and a co-purchase network: Amazon Photo [19]. Statistical overview of all datasets can be found in [23].

We thoroughly evaluate the performance of SpGAT with fourteen representative baselines:

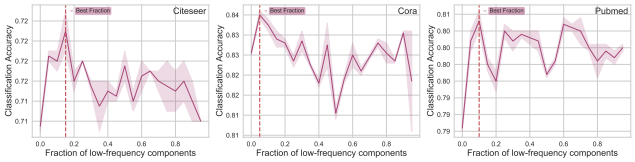


Figure 3: The performance of SpGAT w.r.t the proportion of low-frequency d on citation datasets. The best fraction is marked with the dashed vertical line.

- **Traditional graph embedding methods**: DeepWalk [21] and Planetoid [32];
- **Spectral-based GNNs**: ChebyNet [5], VanillaGCN [13],SGC [28], GWNN [30], ARMA [3], and GZoom(DGI) [6];
- **Spatial-based GNNs**: GGNN [16], GraphSAGE [10], GAT [26], HyperGraph [2], HighOrder [20], and APPNP [14].

For all experiments, a 2-layer neural network is constructed using TensorFlow [1] with 64 hidden units. We train our model utilizing the Adam optimizer [12] with an initial learning rate $lr = 0.01$. Early stopping is used with a window size of 100. Most training processes are stopped in less than 200 steps as expected. We initialize the weights following [9], employ 5×10^{-4} L2 regularization and dropout the input and hidden layers to prevent overfitting [25]. For constructing wavelets, we set $s = 1$, $t = 1 \times 10^{-4}$ for SpGAT, and $M = 1$, $s = 2$ and $t = 1 \times 10^{-4}$ for SpGAT-Cheby on all datasets. In addition, we employ the grid search to determine the best d of low-frequency components and the impact of this parameter would be discussed in Section 4.3. Two variants with MEAN-pooling and MAX-pooling are implemented to demonstrate the effectiveness of aggregation function in SpGAT and SpGAT-Cheby. Without other specification, we use MAX-pooling in both models.

4.2 Semi-supervised Node Classification

Table 1 summaries the results on all datasets. For all baselines, we reuse the results from their public literature. From Table 1, we have these findings: (1) Clearly, the attention-based GNNs (GAT, SpGAT and SpGAT-Cheby) achieve relatively better performance across all datasets. It validates that the attention mechanism can capture the important patterns from either spatial or spectral perspective. (2) Specifically, SpGAT and SpGAT-Cheby achieve the best performance across all datasets. Particularly on Coauthor CS, the best accuracy by SpGAT-Cheby-MAX is 92.5% and it is better than the previous best (90.7%), which is regarded as a remarkable boost considering the challenge on this benchmark. (3) Compared with MEAN aggregation, MAX aggregation seems to be a better choice for both models. This may due to that MAX aggregation can preserve the significant signals learned by SpGAT. (4) It is worthy to note that to achieve such results, both SpGAT and SpGAT-Cheby only employ the attention on low- and high-frequency of graphs in spectral domain, while GAT needs to learn the attention weights on every edge in spatial domain. It verifies that SpGAT is more efficient than GAT, since the global information of graphs can be better captured from spectral domain while with less parameters.

4.3 Ablation Studies

4.3.1 The Impact of Proportion of Low-frequency Components d. To evaluate the impact of the hyperparameter d, we fix the other hyperparameters and vary d from 0 to 100% linearly to run SpGAT on citation datasets. Figure 3 depicts the mean (in bold line) and variance (in light area) of every d. As shown in Figure 3, the mean value curve of three datasets exhibits similar pattern, that is, the best performance is achieved when d is small. The best proportions of low-frequency components are 15%, 5% and 10% for Citeseer, Cora and Pubmed, respectively. In the other words, consistently, only a relatively small fraction of components needs to be treated as low-frequency components in SpGAT. This finding is consistent with the argument from [18], which has discussed that a small fraction of low-frequency components already contains sufficient information for the reconstruction of signals. Thus it might be a good choice to select a small d for our model when generalizing to other datasets. A theoretically heuristic method to determine d could be interesting and will be left for future exploration.

Table 1: Experimental results (in percentage) on semi-supervised node classification.

Model	Citeseer	Cora	Pubmed	Coauthor CS	Amazon Photo
DeepWalk [21]	43.2	67.2	65.3	–	–
Planetoid [32]	64.7	75.7	77.2	–	–
ChebyNet [5]	69.8	81.2	74.4	90.5	89.6
VanillaGCN [13]	70.3	81.5	79.0	89.8	90.6
GWNN [30]	71.7	82.8	79.1	90.3	88.5
ARMA [3]	70.9	83.3	78.4	90.6	86.4
SGC [28]	71.9	81.0	78.9	89.3	90.1
GZoom(DGI) [6]	71.7	83.2	77.1	88.9	89.3
GGNN [16]	64.6	77.6	75.8	86.6	74.1
GraphSAGE [10]	67.2	74.5	76.8	90.1	90.1
GAT [26]	72.5	83.0	79.0	85.5	89.7
HyperGraph [2]	71.2	82.7	78.4	86.9	87.5
HighOrder [20]	64.2	76.6	75.0	84.2	26.1
APPNP [14]	**72.7**	83.1	79.1	90.7	91.8
SpGAT-MEAN	71.6 ± 0.2	82.6 ± 0.3	80.3 ± 0.2	91.0 ± 0.3	91.8 ± 0.3
SpGAT-MAX	$\mathbf{72.1} \pm 0.2$	$\mathbf{83.7} \pm 0.2$	$\mathbf{80.6} \pm 0.3$	91.6 ± 0.3	91.4 ± 0.2
SpGAT-Cheby-MEAN	70.0 ± 0.2	80.7 ± 0.4	78.3 ± 0.3	91.1 ± 0.2	92.4 ± 0.1
SpGAT-Cheby-MAX	71.1 ± 0.4	82.1 ± 0.3	80.2 ± 0.2	$\mathbf{92.1} \pm 0.1$	$\mathbf{92.8} \pm 0.2$

Table 2: Running time (s) comparison for obtaining spectral wavelets $\mathbf{\Psi}_s(\lambda)$ between SpGAT and SpGAT-Cheby.

Models	Eigen-decomposition	Fast approximation
Citeseer	11.23	5.19 (~**2.2**×)
Cora	5.79	2.78 (~**2.1**×)
Pubmed	1185.12	150.79 (~**7.9**×)

4.3.2 Time Efficiency of SpGAT and SpGAT-Cheby. As discussed in Section 3, we propose the fast approximation of spectral wavelets $\mathbf{\Psi}_s(\lambda)$ according to Chebyshev polynomials. To elaborate its efficiency, we compare the time cost of calculating $\mathbf{\Psi}_s(\lambda)$ between via eigen-decomposition (SpGAT) and fast approximation (SpGAT-Cheby). We report the mean time cost of SpGAT and SpGAT-Cheby with second-order Chebyshev polynomials after 10 runs for citation datasets. As shown in Table 2, we can find that this fast approximation can greatly accelerate the training process. Specifically, SpGAT-Cheby runs 7.9× times faster than SpGAT for obtaining $\mathbf{\Psi}_s(\lambda)$ on the relatively large dataset Pubmed. It further validates the scalability of the fast approximation approach.

Meanwhile, we also compare the running time cost of 200 epochs among our proposed methods, VanillaGCN and GAT on Pubmed dataset, and the results can be found in Figure 4. We can observe that SpGAT runs slightly faster than GAT. Furthermore, with the fast approximation technique, SpGAT-Cheby saves nearly half of the time comparing with GAT and achieves comparable efficiency

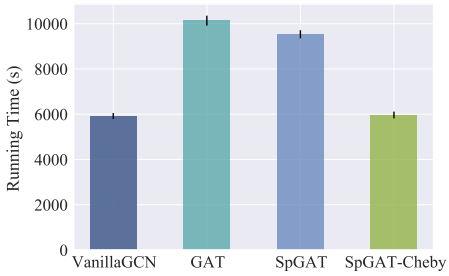


Figure 4: Running time (s) comparison on Pubmed.

against VanillaGCN. It further confirms the efficiency of the proposed approximation on spectral wavelets.

5 CONCLUSION

In this paper, we propose SpGAT, a novel spectral-based graph convolutional neural network to learn the representation of graphs with respect to different frequency components in the spectral domain. By introducing the distinct trainable attention weights for low- and high-frequency components, SpGAT can effectively capture both local and global information in graphs and enhance the performance of GNNs. Furthermore, a fast variant SpGAT-Cheby based on Chebyshev polynomial approximation is proposed to accelerate the spectral graph wavelets calculation and benefit the scalability. To the best of our knowledge, this is the first attempt to adopt the attention mechanism to the spectral domain of graphs. It is expected that SpGAT and SpGAT-Cheby could shed light on building more efficient architectures for the area of graph learning.

REFERENCES

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org* 1, 2 (2015).

[2] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2019. Hypergraph Convolution and Hypergraph Attention. *arXiv preprint arXiv:1901.08150* (2019). https://academic.microsoft.com/paper/2911251106

[3] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2019. Graph Neural Networks with convolutional ARMA filters. *arXiv preprint arXiv:1901.01343* (2019). https://academic.microsoft.com/paper/2912636151

[4] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. 2019. Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks with Octave Convolution. *arXiv preprint arXiv:1904.05049* (2019).

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *neural information processing systems* (2016), 3844–3852.

[6] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. 2020. GraphZoom: A Multi-level Spectral Approach for Accurate and Scalable Graph Embedding. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1lGO0EKDH

[7] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning Structural Node Embeddings via Diffusion Wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1320–1329.

[8] Hongyang Gao and Shuiwang Ji. 2019. Graph representation learning via hard and channel-wise attention networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 741–749.

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

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *neural information processing systems* (2017), 1024–1034.

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

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *international conference on learning representations* (2017).

[14] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations (ICLR)*.

[15] Jia Li, Yu Rong, Hong Cheng, Helen Meng, Wen-bing Huang, and Junzhou Huang. 2019. Semi-Supervised Graph Classification: A Hierarchical Graph Perspective. In *The World Wide Web Conference (WWW), San Francisco, CA, USA*. 972–982.

[16] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks.. In *ICLR (Poster)*. https://academic.microsoft.com/paper/2950898568

[17] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S. Zemel. 2019. LanczosNet: Multi-Scale Deep Graph Convolutional Networks. In *ICLR 2019 : 7th International Conference on Learning Representations*.

[18] Takanori Maehara. 2019. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters. *arXiv preprint arXiv:1905.09550* (2019).

[19] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 43–52.

[20] Christopher Morris, Martin Ritzert, Matthias Fey, William Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. *AAAI 2019 : Thirty-Third AAAI Conference on Artificial Intelligence* 33, 1 (2019), 4602–4609. https://academic.microsoft.com/paper/2962810718

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

[22] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *Ai Magazine* 29, 3 (2008), 93–106.

[23] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[24] David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. 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 (2013), 83–98.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR 2018 : International Conference on Learning Representations 2018*.

[27] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.

[28] Felix Wu, Amauri H. Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML 2019 : Thirty-sixth International Conference on Machine Learning*. 6861–6871.

[29] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).

[30] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2019. Graph Wavelet Neural Network. *international conference on learning representations* (2019).

[31] Yiding Yang, Xinchao Wang, Mingli Song, Junsong Yuan, and Dacheng Tao. 2019. SPAGAN: Shortest Path Graph Attention Network.. In *IJCAI*. 4099–4105.

[32] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML 2016*. 40–48.

[33] Liang Yao, Chengsheng Mao, and Yuchen Luo. 2019. Graph Convolutional Networks for Text Classification. *AAAI 2019 : Thirty-Third AAAI Conference on Artificial Intelligence* 33 (2019), 7370–7377. https://academic.microsoft.com/paper/2962946486

[34] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. 2019. Graph Convolutional Networks for Temporal Action Localization. In *Proceedings of the IEEE International Conference on Computer Vision*. 7094–7103.

[35] Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. 2019. Adaptive structural fingerprints for graph attention networks. In *International Conference on Learning Representations*.

[36] Shuo Zhang and Lei Xie. 2019. Improving attention mechanism in graph neural networks via cardinality preservation. *arXiv preprint arXiv:1907.02204* (2019).