Spectral Graph Attention Network

Heng Chang
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
Tencent AI Lab
hwenbing@126.com

Somayeh Sojoudi
UC 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 networks (GATs) first employ a self-attention strategy to learn attention weights for each edge in the spatial domain. However, learning the attentions over edges only pays attention to the local information of graphs and greatly increases the number of parameters. In this paper, we first introduce attentions in the spectral domain of graphs. Accordingly, we present Spectral Graph Attention Network (SpGAT) that learn 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. We thoroughly evaluate the performance of SpGAT in the semi-supervised node classification task and verified the effectiveness of the learned attentions in the spectral domain.

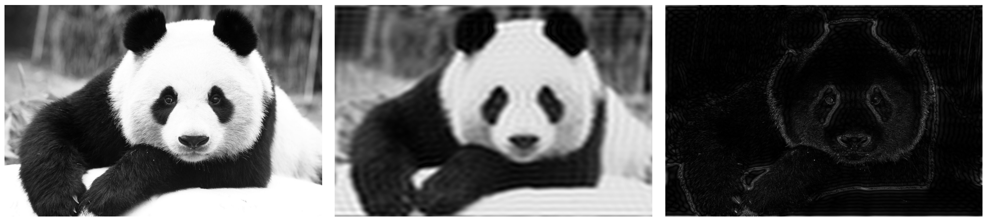
1 Introduction

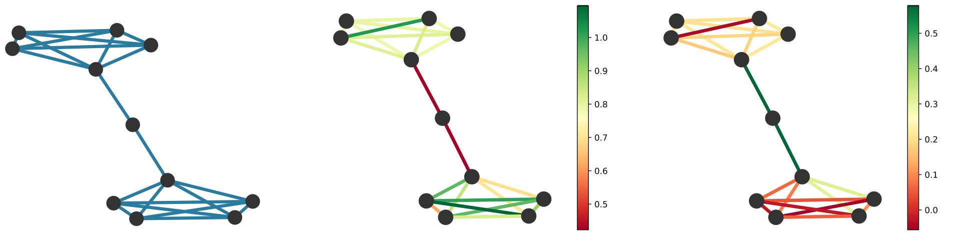
Graph Neural Networks (GNNs) [1] aim at imitating the expressive capability of deep neural networks from grid-like data (*e.g.* images and sequences) to graph structures. The fruitful progress of GNNs in the past decade has made them a crucial kind of tools for a variety of applications, from social networks [2], computer vision [3, 4], text classification [5], to chemistry [6].

Graph Attention Network (GAT) [7], as one central type of GNNs introduces the attention mechanism to further refine the convolution process in generic GCNs [8]. Specifically, during the node aggregation, GAT assigns a self-attention weight to each edge, which can capture the local similarity among neighborhoods, and further boost the expressing power of GNNs as the weight itself is learnable. Many variants have been proposed since GAT [9, 10, 11].

GAT, along with its variants, 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 weights for every edge in the graph is inefficient, especially for large graphs.

Preprint. Under review.


(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. For the graph part, a unweighted barbell graph is reconstructed with filtering out only half low-frequency components ((e)) and half high-frequency components ((f)). The color of edge represents the corresponding weight of edge. Color bars in reconstructed barbell graph (e) and (f) indicate the measurement of reconstructed edge weights.

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. Figure 1(a) ~ 1(c) depict the example of low- and high-frequency components on a panda image. Obviously, the contribution of different frequency components varies with different downstream tasks. To accommodate this phenomenon, [12] proposed Octave Convolution (OctConv) to factorize convolutional feature maps into two groups of different spatial frequencies and process them with different convolutions at their corresponding frequency.

In graph representational learning, this decomposing of low- and high-frequency can be observed more naturally, since graph signal processing (GSP) provides us a way to 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. In contrast, the eigenvectors associated with large eigenvalues carry sharply varying signals across edges [13]. 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, but reserve knowledge between the clusters when constructed with only high-frequency ones. As pointed out by [14, 15], the low- and high-frequency components in the spectral domain may reflect the local and global structural information of graphs in the spatial domain respectively. Moreover, recent works [13, 15] reveal the different importance of low- and high-frequency components of graphs that contributes to the learning of modern GNNs.

Inspired by recent works, we propose to extend the attention mechanism to the spectral domain of graph to 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 these two kernels to capture the their importance respectively. Finally, the pooling function, as well as the 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 the variant SpGAT-Cheby which is more efficient on large graphs. We thoroughly

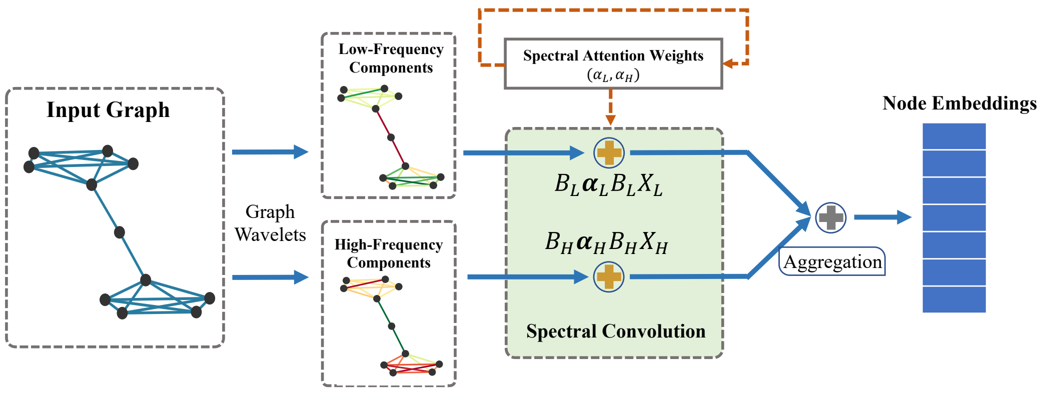

Input Graph
Graph Wavelets
Low-Frequency Components
High-Frequency Components
Spectral Attention Weights (α_L, α_H)
$B_L \boldsymbol{\alpha}_L B_L X_L$
$B_H \boldsymbol{\alpha}_H B_H X_H$
Spectral Convolution
Aggregation
Node Embeddings

Figure 2: The overview of SpGAT.

validate the performance of SpGAT and SpGAT-Cheby on five challenging benchmarks with eleven competitive baselines. SpGAT and SpGAT-Cheby achieve state-of-the-art results on most datasets. The contributions of this paper are summarized as follows:

- To better exploit the local and global structural information of graphs, we propose to extend the attention to the spectral domain rather than the spatial domain and design SpGAT. To the best of our knowledge, SpGAT is the first attempt to adopt the attention mechanism to the spectral domain of graphs.
- Compared with traditional GAT, which needs to compute the attentions for each edge, SpGAT only employs the attention operation on low- and high-frequency components in the spectral domain. We show that SpGAT has the same parameter complexity as VanillaGCN.
- To accelerate the computation of the spectral wavelets, we propose the Chebyshev Polynomial approximation which reduces the computation complexity and achieves at most 7.9x acceleration in benchmark datasets.
- Extensive experiments show the superiority of SpGAT and demonstrate the rationale behind the attention on the spectral domain.

2 Preliminary

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 a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, where $\boldsymbol{A}_{ij} = 1$ indicates an edge (v_i, v_j). We denote $\boldsymbol{D} = \text{diag}(d_1, \cdots, d_n)$ as the node degrees matrix, where d_i represents the node degree of node v_i.

The original version of VanillaGCN is developed by [8]. The feed-forward GCN layer is defined as:

$$\boldsymbol{H}' = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}\boldsymbol{\Theta}^{\mathrm{T}}), \tag{1}$$

where $\boldsymbol{H}' = \{\boldsymbol{h}_1'^{\mathrm{T}}, \cdots, \boldsymbol{h}_n'^{\mathrm{T}}\}$ are the output of hidden vectors from the layer with $\boldsymbol{H}$ as the input features. $\hat{\boldsymbol{A}} = \hat{\boldsymbol{D}}^{-1/2}(\boldsymbol{A} + \boldsymbol{I})\hat{\boldsymbol{D}}^{-1/2}$ refers to the normalized adjacency matrix, where $\hat{\boldsymbol{D}}$ is the corresponding degree matrix of $\boldsymbol{A} + \boldsymbol{I}$. $\sigma(\cdot)$ 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 refers to the feature dimension of input and output respectively.

From the spatial perspective, VanillaGCN is viewed as the feature aggregation among the neighbors of nodes in the spatial domain of graphs. Therefore, we rewrite Eq. (1) to a more general form:

$$\boldsymbol{h}_i' = \sigma(\mathsf{AGG}_{j\in\mathcal{N}_i}(\alpha_{ij}\boldsymbol{\Theta}\boldsymbol{h}_j)), \tag{2}$$

where $\mathcal{N}_i$ refers to the neighborhood set of node i in graph[1]; α_{ij} refers to the aggregation weight of neighbor j for node i; and $\mathsf{AGG}(\cdot)$ refers to the aggregation function that aggregates the output of

[1]Usually, we include v_i in $\mathcal{N}_i$.

each neighbor, such as SUM and MEAN. Usually, VanillaGCN can be viewed as the special case of Eq. (2) where $\alpha_{ij} = \hat{\boldsymbol{A}}_{ij}$, and $\mathsf{AGG}(\cdot) = \mathsf{SUM}(\cdot)$.

Based on Eq. (2), [7] introduces the attention mechanism in graphs and proposes Graph Attention Network (GAT). Concretely, instead of employing the (normalized) adjacency matrix as the aggregation weight, GAT proposes to compute the weight by a self-attention strategy, namely:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k \in \mathcal{N}_i} \exp e_{ik}}, \tag{3}$$

where $e_{ij} = \boldsymbol{\beta}(\boldsymbol{\Theta} h_i, \boldsymbol{\Theta} h_j)$. $\boldsymbol{\beta} \in \mathbb{R}^{2 \times q}$ refers to the attention weight. On one hand, compared with VanillaGCN, it is expensive for GAT to compute the attention weight for every edge in spatial domain because the parameter complexity of K-head GAT is $\mathcal{O}(K \times (p+2) \times q)$, while that of VanillaGCN is $\mathcal{O}(p \times q)$. On the other hand, the self-attention strategy in GAT only consider the local structural information in graphs, *i.e.*, the neighborhoods. It ignores the global structural information in graphs.

3 Spectral Graph Attention Network

Other than the neighbor aggregation in the spatial domain, from [8, 16, 17], VanillaGCN can also be understood as the Graph Signal Processing in the spectral domain:

$$g_\theta \star \boldsymbol{x} = \boldsymbol{B} g_\theta \boldsymbol{B}^{\mathrm{T}} \boldsymbol{x}\,, \tag{4}$$

where $\boldsymbol{x}$ is a signal on every node. $\boldsymbol{B} = \{\boldsymbol{b}_1, \cdot, \boldsymbol{b}_n\}$ are the spectral bases extracted from graphs. $g_\theta = \mathrm{diag}(\theta)$ is a diagonal filter parameterized by θ. Given Eq.(4), VanillaGCN can be viewed as the spectral graph convolution based on the Fourier transform on graphs with first-order Chebyshev polynomial approximations [8]. Further, we can separate the spectral graph convolution into two stages [18]:

$$\begin{aligned} \mathsf{feature\ transformation}: \boldsymbol{X} &= \boldsymbol{H}\boldsymbol{\Theta}^{\mathrm{T}}, \\ \mathsf{graph\ convolution}: \boldsymbol{H}' &= \sigma(\boldsymbol{B}\boldsymbol{F}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{X}). \end{aligned} \tag{5}$$

In Eq. (5), $\boldsymbol{F}$ is the diagonal matrix for graph convolution kernel. For instance, the graph convolution kernel for VanillaGCN is $\boldsymbol{F} = \mathrm{diag}(\lambda_1, \cdots, \lambda_n)$, $\{\lambda_i\}_{i=1}^n$ is the eigenvalues of the normalized graph Laplacian matrix $\boldsymbol{L} = \boldsymbol{I} - \hat{\boldsymbol{A}}$ in ascending order, while the spectral bases $\boldsymbol{B}$ for VanillaGCN is the corresponding eigenvectors.

3.1 The Construction of SpGAT Layer

In this section, we start to describe the construction of SpGAT layer. From the Graph Signal Processing perspective, the diagonal values $(f_1, \cdots, f_n)$ on $\boldsymbol{F}$ can be treated as the **frequencies** on the graph. 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 split the spectral bases into two groups and re-write Eq. (5) as follows:

$$\begin{aligned} &\boldsymbol{X_L} = \boldsymbol{H}\boldsymbol{\Theta}_L^{\mathrm{T}}, \boldsymbol{X}_H = \boldsymbol{H}\boldsymbol{\Theta}_H^{\mathrm{T}} \\ &\boldsymbol{H}' = \sigma(\mathsf{AGG}(\boldsymbol{B}_L\boldsymbol{F}_L\boldsymbol{B}_L^{\mathrm{T}}\boldsymbol{X}_L, \boldsymbol{B}_H\boldsymbol{F}_H\boldsymbol{B}_H^{\mathrm{T}}\boldsymbol{X}_H)), \end{aligned} \tag{6}$$

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 decides the splitting boundary of low- and high-frequency. When $\mathsf{AGG}(\cdot) = \mathsf{SUM}(\cdot)$, Eq. (6) is equivalent to the graph convolution stage in Eq. (5).

In Eq. (6), $\boldsymbol{F}_L$ can be viewed as the importance of the low- and high-frequency. Therefore, we introduce the learnable attention weights by exploiting the re-parameterization trick:

$$\boldsymbol{H}' = \sigma(\mathsf{AGG}(\boldsymbol{B}_L\boldsymbol{\alpha}_L\boldsymbol{B}_L^{\mathrm{T}}\boldsymbol{X}_L, \boldsymbol{B}_H\boldsymbol{\alpha}_H\boldsymbol{B}_H^{\mathrm{T}}\boldsymbol{X}_H)). \tag{7}$$

In Eq. (7), $\boldsymbol{\alpha}_L = \mathrm{diag}(\alpha_L, \cdots, \alpha_L)$ and $\boldsymbol{\alpha}_H = \mathrm{diag}(\alpha_H, \cdots, \alpha_H)$ are the two diagonal matrices parameterized by two learnable attention α_L and α_H, respectively. To ensure α_L and α_H are positive and comparable, we normalize them by the softmax function:

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

Theoretically, there are many approaches to re-parameterize $\boldsymbol{\alpha}_L$ and $\boldsymbol{\alpha}_H$, such as self-attention w.r.t the spectral bases $\boldsymbol{b}_i$. 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 graphs, such as under graph-based semi-supervised learning setting. Meanwhile, we validate that such re-parameterization is simple but efficient and effective in practice.

3.2 Choice of Spectral Bases

Another important issue of SpGAT is the choice of the spectral bases. While the Fourier bases have become the common choice in construction of spectral graph convolution, recent works [13, 18] observed the advantages by utilizing spectral wavelets as bases in graph embedding techniques over traditional Fourier ones. Instead of Fourier bases, we choose the graph wavelets as spectral bases in SpGAT.

Formally, the spectral graph wavelet $\psi_{si}(\lambda)$ is defined as the signal resulting from the modulation in the spectral domain of a signal $\boldsymbol{x}$ centered around the associated node i [19, 20]. Then, given the graph G, the graph wavelet transform is conducted by employing a set of wavelets $\boldsymbol{\Psi}_s = (\psi_{s1}(\lambda_1), \psi_{s2}(\lambda_2), \dots, \psi_{sn}(\lambda_n))$ as bases. Concretely, the spectral graph wavelet transformation is given as:

$$\Psi_s(\lambda) = \boldsymbol{U} g_s(\lambda) \boldsymbol{U}^{\mathrm{T}}, \tag{8}$$

where $\boldsymbol{U}$ is the eigenvectors of normalized graph Laplacian matrix $\boldsymbol{L} = \boldsymbol{I} - \hat{\boldsymbol{A}}$, $g_s(\lambda) = \mathrm{diag}\big(g_s(\lambda_1), g_s(\lambda_2), \dots, g_s(\lambda_n)\big)$ is a scaling matrix with heat kernel scaled by hyperparameter s. The inverse of graph wavelets $\Psi_s^{-1}(\lambda)$ is obtained by simply replacing the $g_s(\lambda)$ in $\Psi_s(\lambda)$ with $g_s(-\lambda)$ corresponding to the heat kernel [13]. Smaller indices in graph wavelets correspond to low-frequency components and vice versa.

The benefits that spectral graph wavelet bases have over Fourier bases mainly fall into three aspects: **1.** Given the sparse real-world networks, the graph wavelet bases are usually much more sparse than Fourier bases, *e.g.*, the density of $\boldsymbol{\Psi}_s$ is 2.8% comparing with 99.1% of U [18]. The sparseness of graph wavelets makes them more computationally efficient for use. **2.** In spectral graph wavelets, the signal $\boldsymbol{\Psi}_s$ resulting from heat kernel filter g_s is typically localized on the graph and in the spectral domain [20]. By adjusting the scaling parameter s, one can easily constrain the range of localized neighborhood. Smaller values of s generally associate with smaller neighborhoods. **3.** Since the information of eigenvalue λ is implicitly contained in wavelets from the process of construction of wavelets, we would not suffer the information loss when do re-parameterization.

Therefore, the architecture of SpGAT layer with graph wavelet bases Ψ_s can be written as:

$$\begin{aligned} \boldsymbol{X} &= \boldsymbol{H}\boldsymbol{\Theta}^{\mathrm{T}} \\ \boldsymbol{H}' &= \sigma(\mathsf{AGG}(\boldsymbol{\Psi}_{sL}\boldsymbol{\alpha}_L\boldsymbol{\Psi}_{sL}^{-1}\boldsymbol{X}, \boldsymbol{\Psi}_{sH}\boldsymbol{\alpha}_H\boldsymbol{\Psi}_{sH}^{-1}\boldsymbol{X}). \end{aligned} \tag{9}$$

In Eq. (9), 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 $\mathcal{O}(2 \times (p \times q + 1))$ to $\mathcal{O}(p \times q + 2)$, which is nearly the same as VanillaGCN. The parameter complexity of SpGAT is much less than that of GAT $\mathcal{O}((p+2) \times q \times K)$ with K-head attention. Comparing with GAT, which captures the local structure of graph from spatial domain, our proposed SpGAT could better tackle global information by combining the low- and high-frequency features explicitly from spectral domain.

4 Fast Approximation of Spectral Wavelets via Chebyshev Polynomials

In SpGAT, directly computing the transformation according to Eq.(8) is intensive for large graphs, since diagonalizing Laplacian L commonly requires $\mathcal{O}(n^3)$ computational complexity. Fortunately, we can employ the Chebyshev polynomials to fast approximate the spectral graph wavelet without eigen-decomposition[19].

Theorem 1. *Let s be the fixed scaling parameter in the heat filter kernel $g_s(\lambda) = e^{-\lambda s}$ and M be the degree of the Chebyshev polynomial approximations for the scaled wavelet (Larger value of M*

yields more accurate approximations but higher computational cost in opposite), the graph wavelet is given by

$$\Psi_s(\lambda) = \frac{1}{2}c_{0,s} + \sum_{i=1}^{M} c_{i,s} T_i(\tilde{\boldsymbol{L}}),$$
$$c_{i,s} = 2e^{-s} J_i(-s) \tag{10}$$

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

Theorem 1 can be deviated from Section 6 in [19]. To further accelerate the computation, we build a look-up table for the Bessel function $J_i(-s)$ to avoid addtional integral operations. With this Chebyshev polynomial approximation, the computational cost of spectral graph wavelets is decreased to $\mathcal{O}(M\|E\| + M \times n)$. Due the real world graphs are usually sparse, this computational difference can be very significant. We denote SpGAT with Chebyshev polynomial approximation as SpGAT-Cheby.

5 Related Works

Spectral convolutional networks on graphs. Existing methods of defining a convolutional operation on graphs can be broadly divided into two categories: spectral based and spatial based methods [21]. We focus on the spectral graph convolutions in this paper. Spectral CNN [22] first attempts to generalize CNNs to graphs based on the spectrum of the graph Laplacian and defines the convolutional kernel in the spectral domain. [23] further employs windowed Fourier transformation to define a local spectral CNN approach. ChebyNet [24] introduces a fast localized convolutional filter on graphs via Chebyshev polynomial approximation. Vanilla GCN [8] further extends the spectral graph convolutions considering networks of significantly larger scale by several simplifications. [25] learns graph-based features on images that are inherently invariant to isometric transformations. Cayleynets [26] alternatively introduce Cayley polynomials allowing to efficiently compute spectral filters on graphs. FastGCN [27] and ASGCN [28] further accelerate the training of Vanilla GCN via sampling approaches. Lanczos algorithm is utilized in LanczosNet [6] to construct low-rank approximations of the graph Laplacian for convolution. SGC [14] further reduces the complexity of Vanilla GCN by successively removing the non-linearities and collapsing weights between consecutive layers. Despite their effective performance, all these convolution theorem based methods lack the strategy to explicitly treat low- and high-frequency components with different importance.

Spectral graph wavelets. Theoretically, the lifting scheme is proposed for the construction of wavelets that can be adapted to irregular graphs in [29]. [19] defines wavelet transforms appropriate for graphs and describes a fast algorithm for computation via fast Chebyshev polynomial approximation. For applications, [30] utilizes graph wavelets for multi-scale community mining and obtains a local view of the graph from each node. [13] introduces the property of graph wavelets that describes information diffusion and learns structural node embeddings accordingly. GWNN [18] first attempts to construct graph neural networks with graph wavelets. These works emphasize the local and sparse property of graph wavelets for graph signal processing both theoretically and practically.

Space/spectrum-aware feature representation. In computer vision, [12] first defines space-aware feature representations based on scale-space theory and reduces spatial redundancy of vanilla CNN models by proposing the Octave Convolution (OctConv) model. [31] further leverages octave convolutions for designing stabilizing GANs. To our knowledge, this is the first time that spectrum-aware feature representations are considered in irregular graph domain and established with graph convolutional neural networks.

6 Experiments

6.1 Datasets

Joining the practice of previous works, we focus on five node classification benchmark datasets under semi-supervised setting with different graph size and feature type. (1) Three citation networks:

Table 1: The overview of dataset statistics.

Dataset	Nodes	Edges	Classes	Features	Label rate
Citeseer	3,327	4,732	6	3,703	0.036
Cora	2,708	5,429	7	1,433	0.052
Pubmed	19,717	44,338	3	500	0.003
Coauthor CS	18,333	81,894	15	6,805	0.016
Amazon Photo	7,487	11,9043	8	745	0.021

Table 2: Experimental results (in percent) on semi-supervised node classification.

Model	Citeseer	Cora	Pubmed	Coauthor CS	Amazon Photo
DeepWalk [37]	43.2	67.2	65.3	–	–
Planetoid [34]	64.7	75.7	77.2	–	–
GGNN [38]	55.5 ± 2.8	47.3 ± 6.1	66.1 ± 4.4	86.6 ± 1.4	74.1 ± 12.3
ChebyNet [24]	70.1 ± 0.8	79.5 ± 1.2	74.4 ± 1.1	90.5 ± 0.4	89.6 ± 1.6
VanillaGCN [8]	70.1 ± 0.7	81.5 ± 0.4	79.0 ± 0.5	89.8 ± 0.3	90.6 ± 0.7
GraphSAGE [39]	65.9 ± 0.9	73.7 ± 1.8	78.5 ± 0.6	90.1 ± 0.4	90.1 ± 1.4
FeaStNet [40]	69.3 ± 1.1	80.4 ± 0.7	76.6 ± 0.6	88.4 ± 0.2	90.5 ± 0.6
GAT [7]	$\mathbf{72.5} \pm 0.7$	83.0 ± 0.7	79.0 ± 0.3	85.5 ± 1.9	89.7 ± 1.7
HyperGraph [41]	67.9 ± 0.5	80.5 ± 0.6	77.4 ± 0.2	86.9 ± 0.5	87.5 ± 0.7
GWNN [18]	70.7 ± 0.4	82.3 ± 0.5	79.0 ± 0.2	90.3 ± 0.2	88.5 ± 0.3
ARMA [42]	65.3 ± 4.1	74.9 ± 10.6	68.5 ± 11.4	90.6 ± 1.1	86.4 ± 3.0
HighOrder [9]	64.2 ± 1.0	76.6 ± 1.2	75.0 ± 2.6	84.2 ± 1.0	26.1 ± 12.4
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	$\mathbf{91.3} \pm 0.3$	$\mathbf{92.8} \pm 0.2$
SpGAT-MEAN	71.6 ± 0.2	82.6 ± 0.3	80.1 ± 0.2	91.0 ± 0.3	91.8 ± 0.2
SpGAT-MAX	72.1 ± 0.2	$\mathbf{83.5} \pm 0.2$	$\mathbf{80.5} \pm 0.3$	91.1 ± 0.3	91.4 ± 0.2

Citeseer, Cora and Pubmed [32], which aims to classify the research topics of papers. (2) A coauthor network: Coauthor CS which aims to predict the most active fields of study for each author from the KDD Cup 2016 challenge[2]. (3) A co-purchase network: Amazon Photo [33] which aims to predict the category of products from Amazon. For the citation networks, we follow the public split setting provided by [34], that is, 20 labeled nodes per class in each dataset for training and 500 / 1000 labeled samples for valiation / test respectively. For the other two datasets, we follow the splitting setting from [35, 36]. Statistical overview of all datasets is given in Table 1. Label rate denotes the ratio of labeled nodes fetched in training process.

6.2 Baselines

We thoroughly evaluate the performance of SpGAT with 11 representative baselines. Among them, DeepWalk [37] and Planetoid [34] are the traditional graph embedding methods. ChebyNet [24], VanillaGCN [8], GWNN [18], ARMA [42] are the spectral-based GNNs. GGNN [38], GraphSAGE [43], GAT [7], FeaStNet [40], HyperGraph [41] and HighOrder [9] are the spatial-based GNNs. In addition, we also implement the the variant of SpGAT with Chebyshev Polynomial approximation, which is denoted as SpGAT-Cheby.

6.3 Experimental setup

For all experiments, a 2-layer network of our model is constructed using TensorFlow [44] with 64 hidden units. We train our model utilizing the Adam optimizer [45] with an initial learning rate $lr = 0.01$. We train the model using early stopping with a window size of 100. Most training process

[2]https://kddcup2016.azurewebsites.net

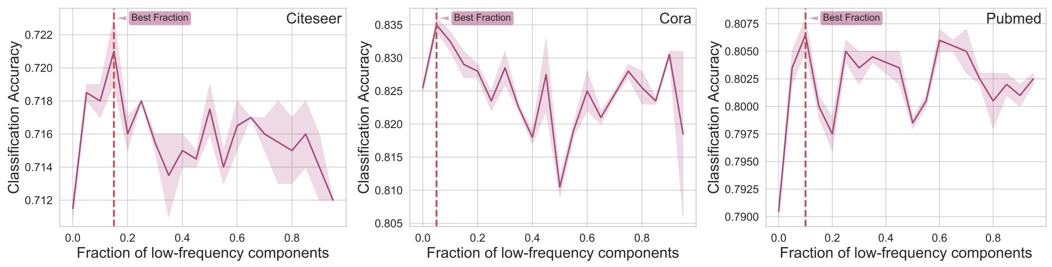


Figure 3: The performance of learned SpGAT w.r.t the proportion of low-frequency components d. The best fraction is marked with the red vertical line.

are stopped in less than 200 steps as expected. We initialize the weights matrix following [46], employ 5×10^{-4} L2 regularization on weights and dropout input and hidden layers to prevent overfitting [47].

For constructing wavelets $\psi_s(\lambda)$, we follow the suggestion from [18] for each dataset, *i.e.*, $s = 0.7$ $t = 1 \times 10^{-5}$ for Citeseer, $s = 0.5$ $t = 1 \times 10^{-7}$ for Pubmed and $s = 1.0$ $t = 1 \times 10^{-4}$ for Cora, Coauthor CS and Amazon Photo. In addition, we employ the grid search to determine the best d of low-frequency components in SpGAT and the impact of this parameter would be discussed in Section 6.5.1. Without specification, we use the MAX aggregation function in SpGAT.

6.4 Performance on Semi-supervised Node Classification

Table 2 summaries the results on all datasets. For all baselines, we reuse result the reported in literatures [7, 36]. From Table 2, we have these findings. (1) Clearly, the attention-based GNNs (GAT, SpGAT and SpGAT-Cheby) achieve the best performance among all datasets. It validates that the attention mechanism can capture the important pattern from either spatial and spectral perspective. (2) Specifically, SpGAT achieves best performance on four datasets; particularly on Pubmed, the best accuracy by SpGAT-MAX is 80.5% and it is better than the previous best(79.0%), which is regarded as a remarkable boost considering the challenge on this benchmark. Meanwhile, compared with baselines, SpGAT-Cheby can also achieve the better performance on three datasets and even gain the best performance on two of them. (4). Compared with MEAN aggregation, MAX aggregation seems a better choice for SpGAT. This may due to MAX aggregation can preserve the significant signals learned by SpGAT. (5) It's worth to note that to achieve such results, both SpGAT and SpGAT-Cheby only employ the attention on low- and high-frequency filter 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 spectral domain contains the meaningful signals and can capture the global information of graphs.

6.5 Ablation Studies

6.5.1 The impact of low-frequency components proportion 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 Citeseer, Cora and Pubmed, respectively. Figure 4 depicts the mean (in bold line) and variance (in light area) of every d on three datasets. As shown in Figure 4, the mean value curve of three datasets exhibits the similar pattern, that is, the best performance are achieved when d is small. The best proportion of low-frequency components are 15%, 5%, and 10% for Citeseer, Cora and Pubmed, respectively. In the other words, consistently, only a small fraction of components needs to be treated as the low-frequency components in SpGAT.

6.5.2 The ablation study on low- and high-frequency Components

To further elaborate the importance of low- and high-frequency components in SpGAT, we conduct the ablation study on the classification results by testing only with low- or high-frequency components w.r.t the best proportion in Section 6.5.1. Specially, we manually set α_L or α_H to 0 during testing stage to observe how the learned low- and high-frequency components in graphs affect the classification accuracy. As shown in Table 3, both low- and high-frequency components are essential for the

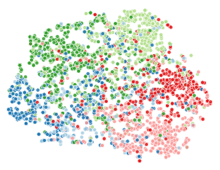
Table 3: The results of ablation study on low- and high-frequency compoents.

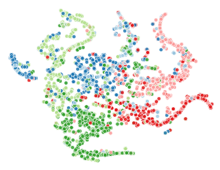
Methods	Citeseer (15%)	Cora (5%)	Pubmed (10%)
with low-frequency	57.70	66.80	76.70
with high-frequency	70.90	82.40	80.40
SpGAT	**71.60**	**83.50**	**80.50**

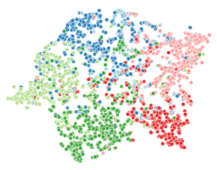
model. Meanwhile, we can find that SpGAT with very small proportion (5% - 15%) of low-frequency components can achieve the comparable results to those obtained by full SpGAT. It reads that the low-frequency components contain more information that can contribute to the feature representation learned from the model.

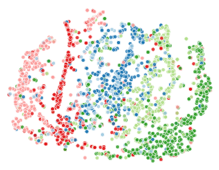
6.5.3 The learned attention on low- and high-frequency components

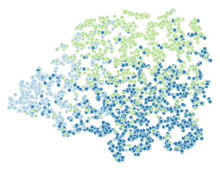
To investigate the results in Section 6.5.2, we further show the learned attentions of SpGAT w.r.t the best proportion for Cietseer, Cora and Pubmed which are demonstrated in Table 4. Interestingly, despite the small proportion, the attention weight of low-frequency components learned by SpGAT is much larger than that of high-frequency components in each layer consistently. Hence, SpGAT is successfully to capture the importance of low- and high-frequency components of graphs in the spectral domain. Moreover, as pointed out by [13, 15], the low-frequency components in graphs usually indicate smooth varying signals which can reflect the locality property in graphs. It implies that the local structural information is important for these datasets. This may explain why *GAT* also gains good performance on these datasets.

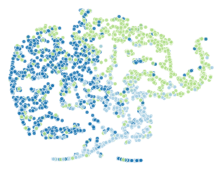

(a) VanillaGCN on Citeseer

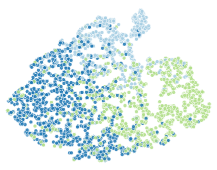

(b) GWNN on Citeseer


(c) GAT on Citeseer


(d) SpGAT on Citeseer


(e) VanillaGCN on Pubmed


(f) GWNN on Pubmed


(g) GAT on Pubmed

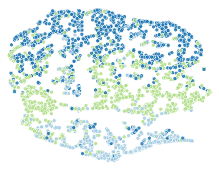

(h) SpGAT on Pubmed

Figure 4: The t-SNE visualization of SpGAT comparing with other baselines. Each color corresponds to a different class that the embeddings belongs to.

6.6 Time Efficiency of Chebyshev Polynomials Approximation

As discussed in Section 4, we propose the fast approximation of spectral wavelets ψ_s according to Chebyshev polynomials. To elaborate its efficiency, we compare the time cost of calculating ψ_s 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 Core, Citeseer and Pubmed respectively. As shown in Table 5, we can find that this fast approximation can greatly accelerate the training process. Specifically, SpGAT-Cheby run 7.9x times faster than that of SpGAT. It validate the efficiency of the proposed fast approximation approaches.

Table 4: Learned attention weights α_L and α_H of SpGAT for low- and high-frequency w.r.t the best proportion of low frequency components d (number followed after the name of datasets).

Dataset	Citeseer (15%)		Cora (5%)		Pubmed (10%)	
Attention filter weights	α_L	α_H	α_L	α_H	α_L	α_H
Learned value (first layer)	**0.838**	0.162	**0.722**	0.278	**0.860**	0.140
Learned value (second layer)	**0.935**	0.065	**0.929**	0.071	**0.928**	0.072

6.7 t-SNE Visualization of Learned Embeddings

To evalute the effectiveness of the learned features of SpGAT qualitatively, we depict the t-SNE visualization [48] of learned embeddings of SpGAT on Citeseer and Pubmed in Figure 4 comparing with VanillaGCN, GWNN and GAT. The representation exhibits discernible clustering in the projected 2D space. In Figure 4, the color indicates the class label in datasets. Compared with the other methods, the intersections of different classes in SpGAT are more separated. It verifies the discriminative power of SpGAT across the classes.

Table 5: Running time (s) comparison for obtaining the spectral wavelets ψ_s between SpGAT and SpGAT-Cheby on Cora, Citeseer and Pubmed.

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×)

7 Conclusion

In this paper, we propose SpGAT, a novel spectral-based graph convolutional neural network to learn the representation of graph with respect to different frequency components in the spectral domain. By introduce the distinct trainable attention weight for low- and high-frequency component, SpGAT can effectively capture both local and global information in graphs and enhance the performance of GNNs. Furthermore, a fast Chebyshev polynomial approximation is proposed to accelerate the spectral wavelet calculation. 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 will shed light on building more efficient architectures for learning with graphs.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596* (2019) .

[2] J. Li, Y. Rong, H. Cheng, H. Meng, W. Huang, and J. Huang, "Semi-supervised graph classification: A hierarchical graph perspective," in *The World Wide Web Conference (WWW), San Francisco, CA, USA*, pp. 972–982. 2019.

[3] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, and E. P. Xing, "Rethinking knowledge graph propagation for zero-shot learning," *arXiv preprint arXiv:1805.11724* (2018) . `https://academic.microsoft.com/paper/2807352623`.

[4] R. Zeng, W. Huang, M. Tan, Y. Rong, P. Zhao, J. Huang, and C. Gan, "Graph convolutional networks for temporal action localization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7094–7103. 2019.

[5] L. Yao, C. Mao, and Y. Luo, "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`.

[6] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "Lanczosnet: Multi-scale deep graph convolutional networks," in *ICLR 2019 : 7th International Conference on Learning Representations*. 2019.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR 2018 : International Conference on Learning Representations 2018*. 2018.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *international conference on learning representations* (2017) .

[9] C. Morris, M. Ritzert, M. Fey, W. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," *AAAI 2019 : Thirty-Third AAAI Conference on Artificial Intelligence* **33** no. 1, (2019) 4602–4609. `https://academic.microsoft.com/paper/2962810718`.

[10] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, pp. 2022–2032. 2019.

[11] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, and G. Chen, "Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems," in *The World Wide Web Conference*, pp. 2091–2102. 2019.

[12] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, S. Yan, and J. Feng, "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution." *arXiv preprint arXiv:1904.05049* (2019) .

[13] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1320–1329. 2018.

[14] F. Wu, A. H. Souza, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *ICML 2019 : Thirty-sixth International Conference on Machine Learning*, pp. 6861–6871. 2019.

[15] T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," *arXiv preprint arXiv:1905.09550* (2019) .

[16] M. Jin, H. Chang, W. Zhu, and S. Sojoudi, "Power up! robust graph convolutional network against evasion attacks based on graph powering," *arXiv preprint arXiv:1905.10029* (2019) .

[17] H. Chang, Y. Rong, T. Xu, W. Huang, H. Zhang, P. Cui, W. Zhu, and J. Huang, "A restricted black-box adversarial framework towards attacking graph embedding models," in *AAAI 2020 : The Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2020.

[18] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," *international conference on learning representations* (2019) .

[19] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis* **30** no. 2, (2011) 129–150.

[20] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine* **30** no. 3, (2013) 83–98.

[21] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *arXiv preprint arXiv:1812.04202* (2018) .

[22] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, pp. http–openreview. 2014.

[23] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks," in *Computer Graphics Forum*, vol. 34, pp. 13–23, Wiley Online Library. 2015.

[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *neural information processing systems* (2016) 3844–3852.

[25] R. Khasanova and P. Frossard, “Graph-based isometry invariant representation learning,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1847–1856, JMLR. org. 2017.

[26] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, “Cayleynets: Graph convolutional neural networks with complex rational spectral filters,” *IEEE Transactions on Signal Processing* **67** no. 1, (2018) 97–109.

[27] J. Chen, T. Ma, and C. Xiao, “Fastgcn: Fast learning with graph convolutional networks via importance sampling,” in *ICLR 2018 : International Conference on Learning Representations 2018*. 2018. `https://academic.microsoft.com/paper/2963695795`.

[28] W. Huang, T. Zhang, Y. Rong, and J. Huang, “Adaptive sampling towards fast graph representation learning,” in *Advances in Neural Information Processing Systems*, pp. 4563–4572. 2018.

[29] W. Sweldens, “The lifting scheme: A construction of second generation wavelets,” *SIAM journal on mathematical analysis* **29** no. 2, (1998) 511–546.

[30] N. Tremblay and P. Borgnat, “Graph wavelets for multiscale community mining,” *IEEE Transactions on Signal Processing* **62** no. 20, (2014) 5227–5239.

[31] R. Durall, F.-J. Pfreundt, and J. Keuper, “Stabilizing gans with octave convolutions,” *arXiv preprint arXiv:1905.12534* (2019) .

[32] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, “Collective classification in network data,” *Ai Magazine* **29** no. 3, (2008) 93–106.

[33] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, “Image-based recommendations on styles and substitutes,” in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52. 2015.

[34] Z. Yang, W. W. Cohen, and R. Salakhutdinov, “Revisiting semi-supervised learning with graph embeddings,” in *ICML 2016*, pp. 40–48. 2016.

[35] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, “Pitfalls of graph neural network evaluation,” *arXiv preprint arXiv:1811.05868* (2018) .

[36] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, “Measuring and relieving the over-smoothing problem for graph neural networks from the topological view,” *arXiv preprint arXiv:1909.03211* (2019) .

[37] B. Perozzi, R. Al-Rfou, and S. 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.

[38] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, “Gated graph sequence neural networks.” in *ICLR (Poster)*. 2016. `https://academic.microsoft.com/paper/2950898568`.

[39] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *neural information processing systems* (2017) 1024–1034.

[40] N. Verma, E. Boyer, and J. Verbeek, “Feastnet: Feature-steered graph convolutions for 3d shape analysis,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2598–2606. 2018. `https://academic.microsoft.com/paper/2791092480`.

[41] S. Bai, F. Zhang, and P. H. S. Torr, “Hypergraph convolution and hypergraph attention.” *arXiv preprint arXiv:1901.08150* (2019) . `https://academic.microsoft.com/paper/2911251106`.

[42] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, “Graph neural networks with convolutional arma filters.” *arXiv preprint arXiv:1901.01343* (2019) . `https://academic.microsoft.com/paper/2912636151`.

[43] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications.” *IEEE Data(base) Engineering Bulletin* **40** (2017) 52–74.

[44] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous systems, 2015,” *Software available from tensorflow. org* **1** no. 2, (2015) .

[45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980* (2014) .

[46] X. Glorot and Y. 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.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research* **15** no. 1, (2014) 1929–1958.

[48] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research* **9** no. Nov, (2008) 2579–2605.