Star Topology Convolution for Graph Representation Learning

Star Topology Convolution for Graph Representation Learning

Chong Wu[1], *Graduate Student Member, IEEE,*
Zhenan Feng[1], Jiangbin Zheng[2], Houwang Zhang[2], *Graduate Student Member, IEEE,* Jiawang Cao[2],
and Hong Yan, *Fellow, IEEE*

Abstract—We present a novel graph convolutional method called star topology convolution (STC). This method makes graph convolution more similar to conventional convolutional neural networks (CNNs) in Euclidean feature spaces. STC learns subgraphs which have a star topology rather than learning a fixed graph like most spectral methods. Due to the properties of a star topology, STC is graph-scale free (without a fixed graph size constraint). It has fewer parameters in its convolutional filter and is inductive, so it is more flexible and can be applied to large and evolving graphs. The convolutional filter is learnable and localized, similar to CNNs in Euclidean feature spaces, and maintains a good weight sharing property. To test the method, STC was compared with state-of-the-art graph convolutional methods in a supervised learning setting on six node properties prediction benchmark datasets: Cora, Citeseer, Pubmed, PPI, Ogbn-Arxiv, and Ogbn-MAG. The experimental results showed that STC achieved state-of-the-art performance on all these datasets and maintained good robustness. In an essential protein identification task, STC outperformed state-of-the-art essential protein identification methods.

Index Terms—Big Data, Graph Convolution, Graph Representation Learning, Spectral Convolution, Star Topology.

1 Introduction

Convolutional neural networks (CNNs) have been used to solve problems which have a Euclidean feature space [1], such as image classification [2] and machine translation [3]. However, most problems, such as 3D meshes, social networks, telecommunication networks, and biological networks, have a non-Euclidean nature [4], and data in the form of a graph is a typical non-Euclidean problem. This makes using CNNs to solve these problems a challenge [1]. There are three major problems in generalizing CNNs to graphs: (1) the numbers of directly connected neighbors for nodes usually differ [5]; (2) the feature dimensions for nodes may also differ; (3) the edges may have features with dimensions that differ.

The convolution operation based on the graph Fourier transform, which is called spectral convolutional neural network (Spectral CNN), was first introduced to the graph domain by [6]. Simplified versions of Spectral CNN based on polynomial expansion have been proposed, like Chebyshev network (ChebyNet) [7] and graph convolutional network (GCN) [8]. ChebyNet [7] restricts the kernel of Spectral CNN to a polynomial expansion. GCN [8], a simplification of ChebyNet to bypass the spectral transform, reduces the computational cost of the eigen decomposition of the graph Laplacian matrix and is able to get spatially localized filters [1]. Due to its good performance and faster speed of convolution computation, GCN is widely used to solve graph problems [9]. It has limitations when scaled to large graphs and is difficult to train in a minibatch setting [10].

To address the problem of scaling GCN to large graphs, layer sampling methods [11], [12], [13], [14], [15] and subgraph sampling methods [10], [16], [17] have been proposed. These are designed for efficient minibatch training of GCN [10]. Layer sampling methods are based on a GCN on the full training graph with nodes or edges sampled from the whole graph to form minibatches in each layer with forward and backward propagation on the sampled graph. Subgraph sampling methods perform subgraph sampling before training the GCN to reduce the problem of large computational costs in the node/edge sampling of each layer [10], [16], [17].

However, the filter size is based on the size of the full graph or the size of the sampled subgraph, which is not flexible enough for different sizes of input. If the size changes, the model trained using the previous filter will need to be reconstructed and retrained. Although some flexible spatial methods [18] have been proposed, their aggregators are neither learnable nor convolutional. The problem of designing a flexible filter, like spatial methods, with a learnable and convolutional property, like spectral methods, still remains to be solved.

To address this problem, we propose star topology convolution (STC). All graphs are composed of subgraphs with a star topology, as Figures 1 & 2 show. A star topology has several advantages in its eigen decomposition, as its eigenvalues are all equal except for the first and last elements. This means that the Laplacian matrices of star topology

- *Chong Wu, Zhenan Feng, Houwang Zhang, and Hong Yan are in the Department of Electrical Engineering, City University of Hong Kong, Kowloon, Hong Kong SAR.*
- *Jiangbin Zheng is in the School of Engineering, Westlake University, Hangzhou, 310024, China.*
- *Jiawang Cao is in the Academy of Engineering & Technology, Fudan University, Shanghai, 200433, China.*
- *[1]These authors contributed equally to the theoretical and experimental parts of this paper. [2]These authors contributed equally to the running of experiments in this paper.*
- *The corresponding author is Chong Wu, E-mail: chongwu2-c@my.cityu.edu.hk*

Manuscript submitted in 2021.

graphs, even of different sizes, will have some common eigenvectors, although they may be rotated. These properties allow the design of a graph-scale free (without a fixed graph size constraint), learnable, and convolutional filter which can maintain weight sharing for subgraphs of different sizes and structures.

With such a flexible filter the spectral convolution over a subgraph with a star topology can be defined. Then, a star topology convolution (STC) can be introduced for graph representation learning to perform the spectral convolution on these subgraphs to obtain the spectral node embedding. These can be aggregated to the central node of each subgraph. Hence, this method can create localized filters in both the spectral and spatial domains. The computational complexity and memory requirements are largely reduced compared to conventional spectral methods. To further improve the performance of STC, an edge attention mechanism is defined on the star topology subgraphs, which consumes less memory than GAT. As STC is inductive and has a low memory requirement, it can be applied to large or evolving graphs and its convolutional process is similar to that in CNNs commonly used in image science.

The contributions of this paper can be summarized as:

1. This is the first attempt to use star topology subgraphs for convolution in graph representation learning.

2. This is a graph convolutional method which has a graph-scale free, learnable, and convolutional filter.

3. This method has a comparatively low computational complexity with high memory efficiency, based on the properties of a star topology, even though it uses a self-attention mechanism.

4. This is a graph convolutional method that is more similar to conventional CNNs than most existing spectral methods.

2 Related work

Since the success of CNNs in areas such as computer vision, natural language processing, and speech processing, researchers began to generalize CNNs to the graph domain. The key to generalizing CNNs to graphs is to define a convolution operator on graphs [1]. Two broad methods, spectral and spatial, categorize current graph convolutional neural networks.

Spectral methods use the graph Fourier transform [19] to transfer signals from the spatial domain to the spectral domain and perform convolution on the spectral domain, which can maintain the weight sharing property of CNNs. Some spectral methods have been applied to node classification. Spectral convolutional neural network (Spectral CNN) [6] introduces the graph Fourier transform directly and uses spectral convolution for graph signals. However, the number of learnable parameters for the filter is large, potentially causing severe computational costs [4]. The Chebyshev network (ChebyNet) [7] restricts the kernel of Spectral CNN to a polynomial expansion. Graph convolutional network (GCN) [8] further simplifies ChebyNet to avoid the Fourier transform by reducing the computational cost of the eigen decomposition of the graph Laplacian matrix and obtains spatially and spectrally localized filters [1]. Graph wavelet neural network (GWNN) introduces a graph wavelet transform to replace the Fourier basis of Spectral CNN to achieve spatially and spectrally localized filters while maintaining a good computational performance.

All these methods are difficult to scale to large graphs and train in a minibatch setting [10]. To scale spectral methods to large graphs, versions of GCN [11], [12], [13], [14], [15] have been designed for efficient minibatch training, but they are based on the whole training graph [10] and iteratively sample nodes or edges from the whole graph to form the minibatches in each layer [10]. This may cause "neighbor explosion" and lead to a large computational complexity. Some heuristic based methods, which make subgraph sampling as a preprocessing step [16], [17], attempt to solve this, but may introduce non-identical node sampling probabilities and bias. To cancel the bias, a graph sampling based inductive learning method, (GraphSAINT) [10], developed a normalization technique so that feature learning does not give larger weights to nodes which are sampled more frequently. These spectral methods still have a problem with generalization: they have a fixed graph size constraint (not graph-scale free). The filter size of spectral methods is determined by the size of the full or sampled sub-graph and larger sizes will result in large computational and memory costs. If the size changes, the model will need to be reconstructed and retrained.

Spatial methods define the convolution on graph geometry. Mixture model CNN (MoNet) uses a weighted average of functions defined over the neighborhood of a node as the spatial convolution operator to provide a general framework for the design of spatial methods [20]. The approach, Graphs with generative adversarial nets (GraphSGAN) [21], facilitates generalization of generative adversarial nets (GANs) to the graph domain through low-density areas by generating fake samples between subgraphs to improve the performance of semi-supervised learning on graphs. Some spatial methods focus on improving model capacity by introducing an attention mechanism to the graph domain, such as the Graph attention network (GAT), which adopts a self-attention mechanism to learn the weighting function [4]. Developments of GAT, such as Dual-primal graph convolutional network (DPGCN) [22] generalized GAT by using convolutions on nodes and edges, giving a better performance and the Hyperbolic graph attention network learns robust node representations of graphs in hyperbolic spaces [23]. The graph sample and aggregate method (GraphSage) [18], a node-based spatial method, learns node, rather than graph, embeddings so it is graph-scale free and can be applied to large or evolving graphs. It performs a uniform node sampling, with a predefined sampling size, to neighbors in each layer iteratively. The sampling size gives an upper bound to the minibatch computational complexity. Unlike STC, proposed here, its aggregator is not learnable. A long-short term memory (LSTM) [24] version has been proposed, but is inherently symmetric.

3 Methods

3.1 Preliminary

Given an undirected graph $G = \{\mathbb{V}, \mathbb{E}, \boldsymbol{A}\}$, where $\mathbb{V} = (V_1, V_2, \ldots, V_n)$ is a node set ($|\mathbb{V}| = n$), $\mathbb{E}$ is an edge set, and $\boldsymbol{A}$ is an adjacency matrix ($\boldsymbol{A}_{[i,j]} = \boldsymbol{A}_{[j,i]}$); its graph

Laplacian matrix $\boldsymbol{L}$ can be defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D} = \mathrm{diag}(\sum_{i \neq j} \boldsymbol{A}_{[i,j]})$ is a degree matrix, and $\boldsymbol{L}$ is a symmetric positive-semidefinite matrix. The eigen decomposition of $\boldsymbol{L}$ is:

$$\boldsymbol{L} = \boldsymbol{U} \Lambda \boldsymbol{U}^{\mathrm{T}}, \tag{1}$$

where, $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n)$ are the eigenvectors, which are orthonormal, and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the diagonal matrix of the corresponding eigenvalues, which are real and non-negative and can be interpreted as the frequencies of the graph. These eigenvectors compose the basis of the feature space in the spectral domain.

3.2 Spectral convolution

For a signal $\boldsymbol{F} = (\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_n)$ on the nodes of graph G, its graph Fourier transform is defined as:

$$\hat{\boldsymbol{F}} = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{F}, \tag{2}$$

Given another signal $\boldsymbol{g}$, the convolution of $\boldsymbol{g}$ and $\boldsymbol{F}$ can be defined as:

$$\boldsymbol{F} \star \boldsymbol{g} = \boldsymbol{U} \boldsymbol{g}_\theta \boldsymbol{U}^{\mathrm{T}} \boldsymbol{F}, \tag{3}$$

where, $\boldsymbol{g}_\theta$ is the graph Fourier transform of $\boldsymbol{g}$. Equation (3) is the spectral convolution, which is similar to the convolution theorem defined in Euclidean feature space, and $\boldsymbol{g}_\theta$ can be regarded as the convolution filter which provides a set of kernel functions.

3.3 Inductive spectral convolution

To apply spectral convolution on large or evolving graphs, we need to introduce an inductive version which can learn the local and global structural properties of each node. Inductive methods focus on obtaining inductive node embedding, rather than whole graph embedding, which provides good flexibility. Hence, the inductive spectral convolution can be defined as a node-based subgraph spectral convolution by:

$$\boldsymbol{F}_{[V_i,:]} \star \boldsymbol{g} = \sum_{V_j \in \mathbb{V}_i} \boldsymbol{W} \hat{\boldsymbol{F}}_{[V_j,:]}, \tag{4}$$

where, $\boldsymbol{W}$ is a flexible filter for subgraphs with different topology, and $\mathbb{V}_i$ is the set of directly connected neighbors of V_i. The key to this work is to find a universal $\boldsymbol{W}$ to make Equation (4) hold.

3.4 Properties of a star topology

$\boldsymbol{W}$ is related to the topology of the subgraphs. A good $\boldsymbol{W}$ should maintain the weight sharing property and, at the same time, provide different kernels for different structures [25]. We need to find a common structure, lying in different subgraphs, which should be identical, or at least symmetric, to help us design filters as in a conventional CNN. We find that star topology graphs with different sizes, n, are symmetric in their structure and Laplacian matrix, as Figures 1 & 2 show. All graphs can be regarded as a composition of subgraphs with a star topology (Figure 2). The eigen decomposition of

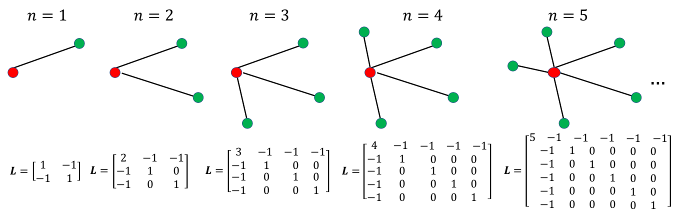


Fig. 1. The structural similarity of graphs with a star topology. The red node is the central node.

an n-dimensional star topology (n neighboring nodes and one central node, $n \geq 1$) can use a universal formulation:

$$\boldsymbol{L} = \begin{bmatrix} n & -1 & \cdots & -1 \\ -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & \cdots & 1 \end{bmatrix} = \boldsymbol{U} \Lambda \boldsymbol{U}^{\mathrm{T}},$$
$$\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n+1}),\ \Lambda = \begin{bmatrix} n+1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{5}$$

where, all elements on the diagonal of Λ are 1, except the first and last elements. That is all eigenvectors have the same weight, except the first and last ones. The largest eigenvalue corresponds to the central node and its connections. All of the value 1 eigenvalues correspond to neighboring nodes and their connections. Since the rank of $\boldsymbol{L} \in \mathbb{R}^{(n+1)*(n+1)}$ is n, the last eigenvalue is 0. The eigenvalues in Equation (5) can be obtained by:

$$\boldsymbol{L}\boldsymbol{U} = \Lambda \boldsymbol{U}, \tag{6}$$

$$\boldsymbol{L}\boldsymbol{U} = \Lambda \mathbf{I} \boldsymbol{U}, \tag{7}$$

$$(\Lambda \mathbf{I} - \boldsymbol{L})\boldsymbol{U} = 0, \tag{8}$$

where, $\mathbf{I}$ is an identical matrix. According to the theorem of linear systems of equations, Equation (8) having non-zero solutions is equivalent to $\det(\Lambda \mathbf{I} - \boldsymbol{L}) = 0$. Using Gaussian elimination on $\det(\Lambda \mathbf{I} - \boldsymbol{L}) = 0$,

$$(\lambda - n - 1)(\lambda - 1)^{n-1} \lambda = 0, \tag{9}$$

the solutions of Equation (9) are Λ.

Eigenvectors $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n+1})$ are orthonormal. Different size star topology graphs have common parts in their eigenvectors for which the corresponding eigenvalue is 1. For an m-dim star topology graph $G_m \in \mathbb{R}^{(m+1)*(m+1)}$ and an n-dim star topology graph $G_n \in \mathbb{R}^{(n+1)*(n+1)}$, where $m \leq n$, the eigenvectors corresponding to eigenvalues of 1 in G_m can be expressed by any $m-1$ eigenvectors with corresponding eigenvalues of 1 in G_n using the vector rotation formula:

$$\boldsymbol{U}_{G_m[2:m]} = \boldsymbol{R} \boldsymbol{U}_{G_n[\boldsymbol{ind}]},\ \boldsymbol{ind} = \mathrm{randperm}(\{2:n\}, m-1), \tag{10}$$

where, $\boldsymbol{R}$ is an identical rotation matrix, and $\boldsymbol{ind}$ is the mask of $m-1$ randomly selected indices. Hence, these eigenvectors can be regarded as equivalent and are suitable to be the universal basis for different scales of subgraphs.

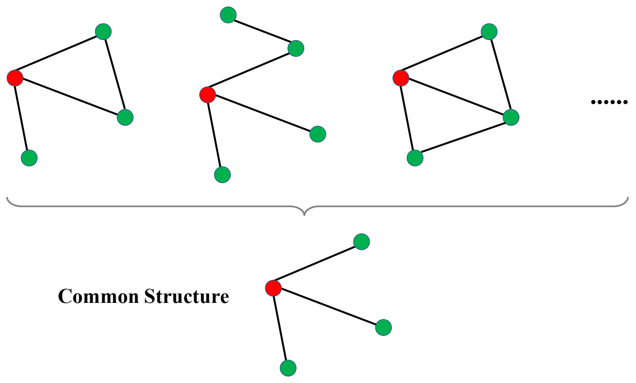


Fig. 2. Subgraphs with a star topology exist in different graphs. The red node is the central node.

Referring to Equation (3), the first row of $\boldsymbol{g}_\theta \boldsymbol{U}^{\mathrm{T}}$ corresponds to the first row of the new feature matrix $\boldsymbol{g}_\theta \boldsymbol{U}^{\mathrm{T}} \boldsymbol{F}$ and also to the first row of $\boldsymbol{F}$. The first row of $\boldsymbol{F}$ corresponds to the central node as in $\boldsymbol{L}$. Since we want to obtain information on neighbors of the central node, the first row of $\boldsymbol{F}$ can be padded to zero. The last eigenvalue is zero which means that the last eigenvector is the least important. By manually adding an artificial node to the original subgraph, we get a new $\boldsymbol{L} \in \mathbb{R}^{(n+2)*(n+2)}$. Adding a zero row under the last row of $\boldsymbol{F}$ won't affect the non-zero rows, so any number of zero rows in $\boldsymbol{F}$ can be added if needed. Then, we can design the flexible filter for $\{1, 2, \ldots, k\}$-dimensional (k can be any non-zero positive integer) star topology subgraphs as:

$$\boldsymbol{W} = \boldsymbol{U}\Theta, \ \boldsymbol{W} \in \mathbb{R}^{(k+2)*(k+2)}, \tag{11}$$

$$\Theta = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \theta_1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \theta_k & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \tag{12}$$

where, Θ is a diagonal matrix which has k learnable parameters. The filter can provide different kernels on different star topology subgraphs and its weight sharing property is shown in Figure 3.

3.5 Star topology convolution

The spectral convolution defined on a star topology can be obtained using the properties of the topology. The update function for the $l+1$-th layer's neighboring information is defined as:

$$h^{l+1}_{[V_j,z]} = \sum_{x=1}^{p} \boldsymbol{W}^{l+1}_{x,z} \boldsymbol{U}^{\mathrm{T}} y^{l}_{[V_j,x]}, \ (z = 1, 2, \ldots, q), \ V_j \in \mathbb{V}_i, \tag{13}$$

where, $y^l_{[V_j,x]}$ is the output of the l-th layer for node V_j, x and z are the dimension indices, p and q are the dimensions of the embedding, and $\mathbb{V}_i$ is the set of directly connected neighbors of the central node V_i. To reduce the computational complexity of h^{l+1}, we first perform spectral convolution:

$$h^{l+1}_{[V_j,x]} = \boldsymbol{W}^{l+1} \boldsymbol{U}^{\mathrm{T}} y^{l}_{[V_j,x]}, \ (x = 1, 2, \ldots, p), \ V_j \in \mathbb{V}_i, \tag{14}$$

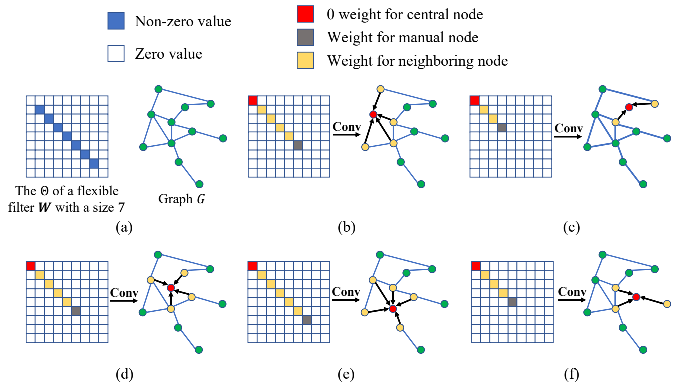

Non-zero value
Zero value
0 weight for central node
Weight for manual node
Weight for neighboring node
The Θ of a flexible filter W with a size 7
Graph G
Conv
(a)
(b)
(c)
(d)
(e)
(f)

Fig. 3. A toy example showing the weight sharing property of the filter on subgraphs with different star topology. The red node is the central node and the yellow nodes are neighboring nodes to be aggregated. The kernels provided by the filter differ by subgraph. It is a general filter which can deal with different subgraphs, that is, the filter is graph-scale free.

Information on the neighbors obtained from the spectral convolution will be aggregated to the central node using feature transformation.

$$h^{l+1}_{[V_i,:]} = \sum_{V_j \in \mathbb{V}_i} \mathsf{s}_{V_j} h^{l+1}_{[V_j,:]}, \tag{15}$$

where, s_{V_j} denotes the weight of edge (V_i, V_j), $h^{l+1}_{[V_i,:]}$ is the convolution result of the neighboring information of the central node V_i. Then $h^{l+1}_{[V_i,:]}$ will be concatenated with the last layer output feature vector $y^l_{[V_i,:]}$ of the central node V_i. After a scaling transformation, the new feature vector will be activated using a nonlinear function as the current layer output of node V_i as:

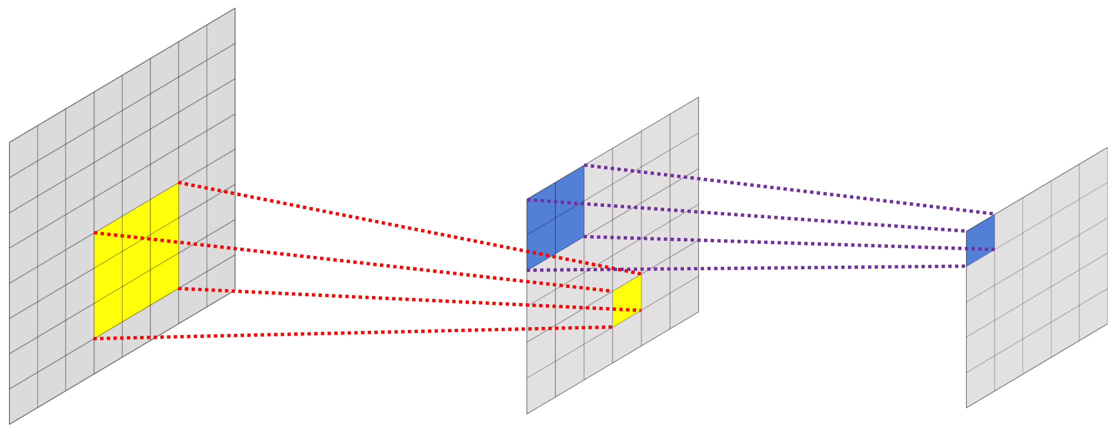
$$y^{l+1}_{[V_i,:]} = \sigma\left(\mathrm{cat}\left(y^l_{[V_i,:]}, h^{l+1}_{[V_i,:]}\right) \boldsymbol{S}^{l+1}\right), \tag{16}$$

where, $y^l_{[V_i,:]}$ is the feature vector of the central node V_i obtained in l-th layer, $\mathrm{cat}()$ denotes a concatenation function, σ denotes a nonlinear activation function, and $\boldsymbol{S}^{l+1} \in \mathbb{R}^{2p*q}$ is a scaling parameter matrix for feature transformation.

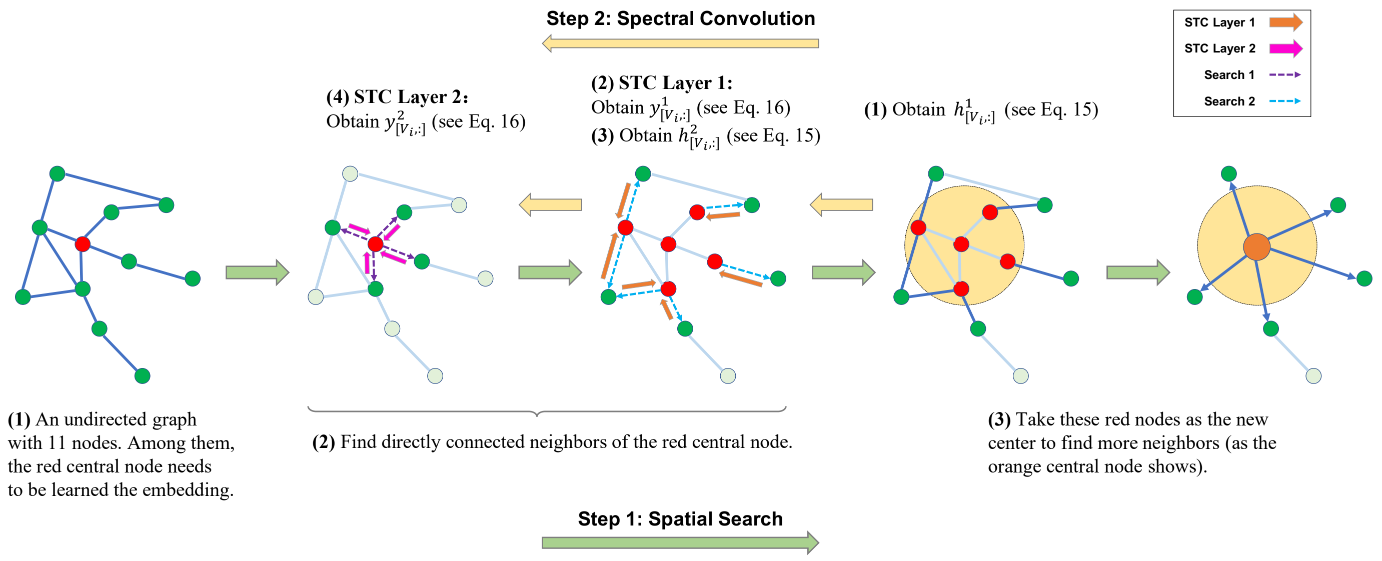
Figure 4 shows the comparison of a two-layer conventional CNN and a two-layer STC. The two-layer STC is formulated as:

$$\begin{aligned} &1^{st}\ layer : y^1_{[V_i,:]} = \\ &\quad \mathbf{ReLU}\left(\mathrm{cat}\left(\boldsymbol{F}_{[V_i,:]}, h^1_{[V_i,:]}\right) \boldsymbol{S}^1\right), \\ &2^{nd}\ layer : y^2_{[V_i,:]} = \\ &\quad \mathbf{ReLU}\left(\mathrm{cat}\left(y^1_{[V_i,:]}, h^2_{[V_i,:]}\right) \boldsymbol{S}^2\right), \end{aligned} \tag{17}$$

where, $\boldsymbol{S}^1 \in \mathbb{R}^{2p*q}$ and $\boldsymbol{S}^2 \in \mathbb{R}^{2q*d}$ are two scaling parameter matrices, and d is the embedding dimension. STC divides the convolution process of a conventional CNN into two cascading steps: (1) spatial search; (2) spectral convolution. Spatial search is used to find spatially localized areas. Hence, our method can directly create spatially localized filters. As with a conventional CNN, STC obtains global information through the introduction of more layers. After the spatial search, a spectral convolution on the subgraphs is obtained. In a conventional CNN, the shape of different areas selected by a window function/kernel is usually the same. In a



(a) CNN


Step 2: Spectral Convolution
STC Layer 1
STC Layer 2
Search 1
Search 2
(4) STC Layer 2: Obtain $y^2_{[V_i,:]}$ (see Eq. 16)
(2) STC Layer 1: Obtain $y^1_{[V_i,:]}$ (see Eq. 16)
(3) Obtain $h^2_{[V_i,:]}$ (see Eq. 15)
(1) Obtain $h^1_{[V_i,:]}$ (see Eq. 15)
Step 1: Spatial Search

(1) An undirected graph with 11 nodes. Among them, the red central node needs to be learned the embedding.

(2) Find directly connected neighbors of the red central node.

(3) Take these red nodes as the new center to find more neighbors (as the orange central node shows).

(b) STC

Fig. 4. Comparison of a conventional CNN and a star topology convolution (STC). In the convolution process, a CNN uses a grid filter to select spatially localized areas from the training image for convolution. Similarly, STC uses a universal filter ***W*** to select spatially localized star topology subgraphs from the training graph for convolution. It adopts a spatial search, similar to GraphSage [18], to find neighbors. The depth of the neural network determines the order of neighbors used in the convolution. Shown is an example of a two-layer STC, hence only first and second order neighbors are selected. Spectral convolution is performed on the selected subgraphs.

3*3 kernel, for example, the different areas selected are all square and have the same size (9 pixels). In a graph, star topology subgraphs have symmetry in their spatial structures which allows STC to have a convolution process similar to a conventional CNN. If a new filter is used to replace ***W***, a new inductive spectral convolutional method can be designed. Hence, STC provides a framework to design inductive spectral convolutional methods.

3.6 Node sampling

Since the filter size k is unknown, it must be confirmed in an implementation. This can be done simply by assigning it as K, where K is the maximum number of connections of a node in the graph. The node sampling strategy of GraphSage [18], giving a dropout function [26], is used to improve the robustness of STC. For a node V_i with $|\mathbb{V}_i|$ directly connected neighbors, when $|\mathbb{V}_i| > k$, k neighbors will be randomly selected in each training iteration.

$$y^l_{[:,:]} = y^l_{[\boldsymbol{ind},:]}, \ \boldsymbol{ind} = \mathrm{randperm}(\mathbb{V}_i, k), \tag{18}$$

where, ***ind*** is the mask, with its size determined by the size of the filter, k. For nodes with $|\mathbb{V}_i| \leq k$ directly connected neighbors, all neighbors will be selected, and these nodes can be considered to not have enough information. Hence, it is necessary to take advantage of all their information. Through controlling the filter size, STC can achieve good robustness. The effect of the filter size and how to select it will be discussed in the ablation study in subsection 4.3.

3.7 Edge attention

We introduce a self-attention mechanism (edge attention) [4] to STC to further improve its performance and robustness. For each star topology subgraph, STC performs convolution and edge attention independently (Figure 5). Edge attention evaluates the importance of each edge of the central node in the star topology subgraph. Then the convolution result h, edge attention result h, and central node's feature y are concatenated to form the output y (a new feature vector for

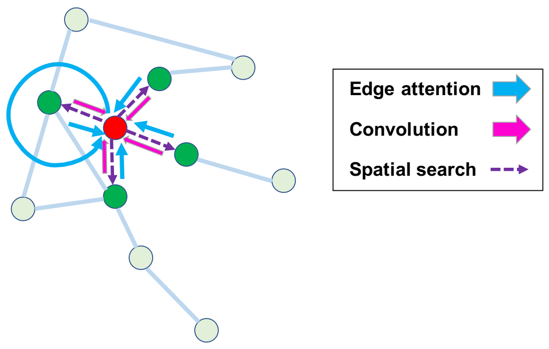


Fig. 5. A toy example showing the convolution and edge attention in STC. Here a star topology subgraph consists of deep green nodes. Convolution and edge attention are performed independently on the star topology subgraph.

the central node) in each layer. Through introduction of a self-attention mechanism to STC, Equation 16 becomes:

$$y^{l+1}_{[V_i,:]} = \sigma\left(\text{cat}\left(y^{l}_{[V_i,:]}, h^{l+1}_{[V_i,:]}, \mathsf{h}^{l+1}_{[V_i,:]}\right) \boldsymbol{S}^{l+1}\right), \tag{19}$$

where $\mathsf{h}^{l+1}_{[V_i,:]}$ is the high-level feature vector obtained by edge attention:

$$\mathsf{h}^{l}_{[V_j,:]} = \text{cat}\left(y^{l}_{[V_i,:]}, y^{l}_{[V_j,:]}\right) \boldsymbol{S}^{l}_{e},\ V_j \in \hat{\mathbb{V}}_i, \tag{20}$$

$$\mathsf{a}^{l+1}_{[V_i,V_j]} = \frac{\exp\left(\sigma\left(\mathsf{h}^{l}_{[V_j,:]}\boldsymbol{a}^{l+1}\right)\right)}{\sum_{V_j\in\hat{\mathbb{V}}_i}\exp\left(\sigma\left(\mathsf{h}^{l}_{[V_j,:]}\boldsymbol{a}^{l+1}\right)\right)}, \tag{21}$$

$$\mathsf{h}^{l+1}_{[V_i,:]} = \sum_{V_j\in\hat{\mathbb{V}}_i} \mathsf{a}^{l+1}_{[V_i,V_j]}\mathsf{h}^{l}_{[V_j,:]}, \tag{22}$$

where, $\boldsymbol{a}^{l+1} \in \mathbb{R}^{2q*1}$ is a scaling parameter vector for concatenated feature, $\hat{\mathbb{V}}_i = \mathbb{V}_i \cup V_i$, $\boldsymbol{S}^{l}_{e} \in \mathbb{R}^{2p*q}$ is a scaling parameter matrix, and $\mathsf{a}^{l+1}_{[V_i,V_j]}$ is the attention weight of edge (V_i, V_j) in the $l+1$-th layer.

3.8 Comparison with state-of-the-art methods

In vanilla GCN, the $l+1$-th layer of GCN can be defined as:

$$\boldsymbol{Y}^{l+1} = \sigma(\boldsymbol{A}\boldsymbol{Y}^{l}\mathsf{W}^{l+1}), \tag{23}$$

where $\boldsymbol{Y}^{l+1}$ is the hidden unit output of the $l+1$-th layer, $\boldsymbol{A}$ is the adjacency matrix which is used to introduce the edge information to the learning process, W^{l+1} is the convolution filter, which is used to do scaling transformation, in the $l+1$-th layer. Compared to GCN, STC uses a spectral filter $\boldsymbol{W}^{l+1}$ to replace $\boldsymbol{A}$ and W^{l+1}. Unlike $\boldsymbol{A}$, $\boldsymbol{W}^{l+1}$ implicitly includes the edge information and STC has an edge attention mechanism.

For the commonly used method GraphSage (mean aggregator), its $l+1$-th aggregator can be defined as:

$$h^{l+1}_{[V_i,:]} = \underset{V_j\in\mathbb{V}_i}{\text{mean}}\left(y^{l}_{[V_j,:]}\right). \tag{24}$$

The embedding of the central node, V_i, in the $l+1$-th layer of GraphSage can be obtained using Equation 16. GraphSage sets all weights to 1 to replace the spectral convolution filter in STC, which will lose some information in the learning process. GraphSage uses a simple mean to aggregate neighboring information to the central node V_i, while STC uses a learnable weighted average function.

GAT performs edge attention on the whole graph by:

$$\boldsymbol{Y}^{l+1} = \sigma\left(\mathsf{A}\boldsymbol{Y}^{l}\mathsf{W}^{l+1}\right), \tag{25}$$

where, A is the attention weight matrix. STC performs edge attention on the star topology subgraphs, making STC more suitable than GAT for large graphs.

3.9 Complexity analysis

The computational complexity of STC can be considered in two parts: (1) the convolution process; (2) the edge attention process. The convolution process of STC depends on the filter size and the size of the scaling parameter matrix, so its computational complexity in the $l+1$-th layer is $2p*q+2k$, where $2p*q$ is the size of scaling parameter matrix and k is the filter size. k is much smaller than $p*q$ in most cases. Compared to other spectral methods, like Spectral CNN [6] (with a complexity of $p*q*|\mathbb{V}|$, where $|\mathbb{V}|$ is the total number of nodes in the full graph) and GWNN [1] (with a complexity of $p*q+|\mathbb{V}|$), the complexity of STC is much smaller allowing it to be used for large or evolving graphs. STC performs edge attention on star topology subgraphs. If the filter size is k, then the maximum size of a star topology subgraph is $k+1$, so its computational complexity is $2q*(p+1)+(k+1)$. The computational complexity of GAT is $2q*(p+1)+(|\mathbb{E}|+|\mathbb{V}|)$, where $|\mathbb{E}|$ is the total number of unique edges in the full graph. $k+1$ is much smaller than $|\mathbb{E}|+|\mathbb{V}|$.

4 Experiments and Results

4.1 Experiment settings

Benchmarks: Six benchmark datasets: Cora [27], Citeseer [27], Pubmed [27], PPI [18], [28], Ogbn-Arxiv [29], and Ogbn-MAG [29] were selected to validate the ability of STC to predict the properties of nodes. Cora, Citeseer, Pubmed, Ogbn-Arxiv, and Ogbn-MAG are five citation datasets. Ogbn-Arxiv and Ogbn-MAG are large datasets, and Ogbn-MAG is a heterogeneous graph. PPI is a dataset used to test the ability of graph convolutional methods to generalize across graphs. STC was tested in an essential protein identification task and compared with several state-of-the-art essential protein identification methods: deep neural network based method (DNN) [30], multi-objective optimization based method (IMAMOBH) [31], degree centrality (DC) [32], betweeness centrality (BC) [33], closeness centrality (CC) [34], eigenvector centrality (EC) [35], edge clustering coefficient centrality (NC) [36], and local average connectivity (LAC) [37]. Essential protein identification can be regarded as a binary node classification task. The protein-protein interaction data of Saccharomyces cerevisiae (Yeast) were downloaded from the DIP database [38] updated to Oct.10, 2010. There are 4512 proteins and 22091 protein-protein interactions, excluding self and repeated interactions, in this dataset. Yeast was selected because its protein-protein interaction data and gene essentiality data are the most complete and reliable among various species. The essential protein list was selected from MIPS [39], SGD [40], DEG [41], and SGDP [42]. There are 1285 essential proteins in the list, 1031 of which are in the protein-protein interaction network. These 1031 proteins were considered essential and the other 3481 proteins were considered non-essential. The gene expression data used as the features were collected from GEO [43], sampling 36 time points during successive Yeast metabolic cycles. The division of training/validation/test sets for Cora, Citeseer, and Pubmed was made according to [44]; for PPI according to [18]; for Ogbn-Arxiv and Ogbn-MAG according to [29]; for the essential protein dataset it was set to 3012:500:1000. The statistics for Cora, Citeseer, and Pubmed can be seen in [1], [44]; for PPI in [4], [18]; for Ogbn-Arxiv and Ogbn-MAG in [29] and for the essential protein dataset in Table 1.

TABLE 1
The statistics of the essential protein dataset.

	Essential protein
Nodes	4512
Edges	22091
Classes	2
Features	36
Training set	3012
Test set	1000
Validation set	500

Baselines: To test STC, it was compared with several state-of-the-art methods: GWNN [1], GAT (sparse) [4], GCN [8], GraphSAINT (mean aggr) [10], GraphSAINT (concat aggr) [10], GraphSage (mean aggr) [18], MoNet [20], Plain Linear + C&S [45], GraphZoom (Node2vec) [46], Node2vec [47], Label Propagation [48], MLP [29], MLP+FLAG [49], CoLinkDistMLP [50], and MetaPath2vec [51]. All methods used their default settings in all experiments in this paper. We followed the protocol of [29] to run all experiments. All methods used a fixed number of layers (2) in all experiments, were run with enough epochs to converge, and used a validation process to select the model with the highest F1-micro score on the validation set for testing. The hidden dimensions of the two layers were kept the same for all methods at 256.

The optimizer for STC was Adam [52] and the learning rate was set to 0.001 for all experiments in this paper. STC adopted a fixed number for the filter size (18) for all datasets, except Ogbn-MAG (Ogbn-MAG was used to do the ablation study of filter size), and a fixed dropout rate (0.5) for all datasets, except Cora (0.8). To clarify the effect of layer depth and the number of parameters in STC on its performance, two variants of STC were tested: STC (conv only), which only adopts the convolution process, and STC (one layer).

All experiments except the ablation study of the filter size and GPU memory cost comparison were conducted on a personal computer (PC1) with Ubuntu OS 18.04, Intel (R) Xeon (R) Silver 4210 2.20 GHz 10-Core CPU, 188 GB RAM, CUDA version 10.2, torch version 1.8.1, and 4 NVIDIA TITAN RTX GPUs. Other experiments were run on a personal computer (PC2) with Windows OS 10 Home, AMD Ryzen 5 3600 3.59 Ghz 6-Core CPU, 16 GB RAM, CUDA version 10.2, torch version 1.8.1, and 1 NVIDIA GeForce GTX 1660 Super GPU.

4.2 Performance analysis

Small datasets: Table 2 shows the test accuracy obtained by STC and state-of-the-art graph convolutional methods on small citation datasets. STC and GCN achieved the highest accuracy on Citeseer. STC also outperformed the other methods on Pubmed. GWNN achieved the highest test accuracy on Cora with the performance of STC just slightly inferior. STC achieved state-of-the-art performance on small datasets. Table 2 shows the number of trainable parameters for each method on these small citation datasets. STC has more parameters, but a comparable number, than other methods at the same level. STC (conv only), which does not have the edge attention mechanism in STC, has similar numbers of parameters to most methods, and performed close to STC on small datasets (better on Pubmed). Increasing the number of parameters has a minor effect on the performance of STC on small datasets. Layer depth affected the performance of STC on small datasets with STC (one layer) having a degraded performance.

Essential protein identification: Table 3 shows the performance of STC and state-of-the-art methods on an essential protein identification task. STC and its variants outperformed other graph convolutional methods and essential protein identification methods. Graph convolutional methods outperformed traditional essential protein identification methods, even DNN, showing that graph convolutional methods have an advantage in solving traditional graph problems. On this dataset, STC (one layer) achieved the best performance, followed by STC (conv only). STC (one layer) appears to have sufficient representation for this problem and a more complex structure may overrepresent this problem.

Medium-sized datasets: Table 4 shows the test accuracy obtained by STC and state-of-the-art graph convolutional methods on a medium-sized dataset. STC achieved the highest accuracy in this dataset and substantially better performance than STC (conv only) and STC (one layer). A more complex structure or more parameters appear to improve the performance on this dataset.

Large datasets: Tables 5-6 compare STC and its variants with leading methods on the large graph datasets Ogbn-Arxiv and Ogbn-MAG. STC outperformed the other methods and, of its variants, only STC (conv only) was close to STC. STC and its variants are homogeneous graph convolutional methods but STC still performed best on the heterogeneous graph (Ogbn-MAG) compared to other homogeneous graph convolutional methods (GraphSage (mean aggr), GCN, CoLinkDistMLP, and MLP) in Table 6 and even better than the heterogeneous graph convolutional method, MetaPath2vec.

In summary, STC achieved state-of-the-art performance on small, middle, and large graphs, demonstrating its robustness. A complex structure (more parameters) helped STC improve its performance on some datasets. A greater depth of layers generally gave better performance. Table 7 compares the ability of STC and state-of-the-art methods to be run on large datasets (Ogbn-Arxiv and Ogbn-MAG) using PC2 (one NVIDIA GeForce GTX 1660 Super GPU (6GB)). GraphSage (mean aggr) and STC can be trained on both datasets but GraphSAINT (mean aggr) and GraphSAINT (concat aggr) exceed available memory on Ogbn-MAG while GWNN and GAT (sparse) do this on both datasets.

4.3 Effect of filter size

The effect of different filter sizes in each STC layer were examined on Ogbn-MAG. Validation and test accuracy (Figure 6) improved with increasing filter size to a peak value (at 13) and then steadily reduced. A small filter size might

TABLE 2
Comparison of test accuracy and number of trainable parameters of STC and state-of-the-art methods on small citation datasets. Best values are in bold, second best values are in italic.

	Citeseer	Num of param.	Cora	Num of param.	Pubmed	Num of param.
STC	**75.50±0.23**	3174224	*86.88±0.37*	1431120	*90.50±0.22*	713552
STC (conv only)	74.55±0.83	2028624	86.50±0.54	866640	**90.77±0.38**	387920
STC (one layer)	72.88±0.35	2911528	82.43±0.59	1168424	88.98±0.57	450856
GraphSage (mean aggr)	75.23±0.60	2028544	86.08±0.44	866560	89.95±0.34	387840
MoNet	69.55±1.36	2848810	80.50±0.78	1106219	84.80±0.62	386617
GAT (sparse)	74.80±0.00	950028	86.20±0.00	369166	87.50±0.00	129286
GCN	**75.50±0.00**	949766	86.20±0.00	368903	89.00±0.00	129027
GWNN	74.93±0.50	956158	**87.85±0.29**	374056	83.85±0.26	168202
GraphSAINT (mean aggr)	69.52±0.28	2097938	81.98±0.67	935957	88.17±0.42	457225
GraphSAINT (concat aggr)	69.27±1.37	2294546	81.52±1.11	1132565	87.80±0.47	653833

TABLE 3
Comparison of test accuracy and number of trainable parameters of STC and state-of-the-art methods in an essential protein identification task. Best values are in bold, second best values are in italic.

	Essential protein	Num of param.
STC	77.55±0.28	356896
STC (conv only)	*77.58±0.31*	150048
STC (one layer)	**77.70±0.39**	94224
GCN	77.50±0.00	9986
GAT (sparse)	77.30±0.00	10244
GraphSage (mean aggr)	77.10±0.39	150016
MoNet	76.27±0.92	29478
DNN	75.80±0.00	281
LAC	75.63±0.00	0
IMAMOBH	75.33±0.13	0
NC	74.69±0.00	0
DC	73.36±0.00	0
EC	71.94±0.00	0
CC	71.51±0.00	0
BC	71.39±0.00	0

TABLE 4
Comparison of test accuracy and number of trainable parameters of STC and state-of-the-art methods on PPI. Best values are in bold, second best values are in italic.

	PPI	Num of param.
STC	**76.01±0.09**	398160
STC (conv only)	69.16±0.17	187728
STC (one layer)	64.48±0.15	135464
GraphSage (mean aggr)	*74.55±0.41*	187648
GCN	64.69±0.00	44153
GraphSAINT (mean aggr)	66.51±0.25	257387
GraphSAINT (concat aggr)	67.64±0.33	453995

TABLE 5
Comparison of test accuracy and number of trainable parameters of STC and state-of-the-art methods on Arxiv. Best values are in bold, second best values are in italic.

	Arxiv	Num of param.
STC	**71.85±0.07**	437328
STC (conv only)	71.45±0.08	206928
STC (one layer)	67.64±0.05	174632
GCN	*71.74±0.29*	110120
GraphSage (mean aggr)	71.49±0.27	218664
Plain Linear + C&S	71.26±0.01	5160
GraphZoom (Node2vec)	71.18±0.18	8963624
Node2vec	70.07±0.13	21818792
Label Propagation	68.32±0.00	0
GraphSAINT (mean aggr)	58.86±0.51	276344
GraphSAINT (concat aggr)	58.11±0.71	472952
CoLinkDistMLP	56.38±0.16	120912
MLP+FLAG	56.02±0.19	110120
MLP	55.50±0.23	110120

TABLE 6
Comparison of test accuracy and number of trainable parameters of STC and state-of-the-art methods on MAG. Best values are in bold, second best values are in italic.

	MAG	Num of param.
STC	**35.59±0.17**	516412
STC (conv only)	34.96±0.35	286012
STC (one layer)	33.45±0.41	253726
MetaPath2vec	*35.44±0.36*	94479069
GraphSage (mean aggr)	31.53±0.15	285952
GCN	30.43±0.25	122717
CoLinkDistMLP	27.61±0.18	278202
MLP	26.92±0.26	188509

TABLE 7
Comparison of STC and state-of-the-art methods using one NVIDIA GeForce GTX 1660 Super GPU (6GB memory) on two large datasets.

	Arxiv	MAG
STC	Available	Available
GraphSage (mean aggr)	Available	Available
GraphSAINT (mean aggr)	Available	Out of memory
GraphSAINT (concat aggr)	Available	Out of memory
GWNN	Out of memory	Out of memory
GAT (sparse)	Out of memory	Out of memory

give insufficient information to train a good model, while larger filter sizes might introduce redundant information to the trained model, leading to overfitting and a degradation in performance. Figure 7 shows that the average batch time obtained by STC with different filter sizes on Ogbn-MAG were linear with respect to filter size. Table 8 shows that the number of parameters required for different filter sizes on Ogbn-MAG differed only slightly. The filter size could be tuned to achieve the best performance / batch time.

TABLE 8
Number of parameters with different filter sizes on MAG.

Filter size	8	13	18	28	38
Num of param.	516392	516412	516432	516472	516512

5 Conclusions

In this paper, we presented a novel graph convolutional method, called star topology convolution (STC), which is a graph-scale free inductive spectral convolutional method. It learns node embedding on subgraphs within a star topology. In experiments, our method outperformed state-of-the-art

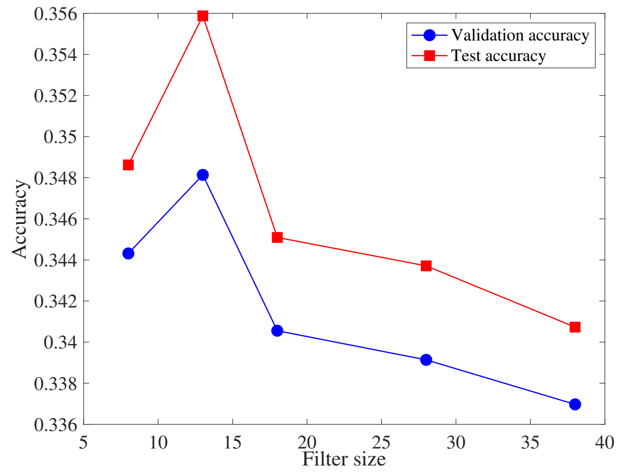


Fig. 6. Mean accuracy of 10 runs at different filter sizes on MAG.

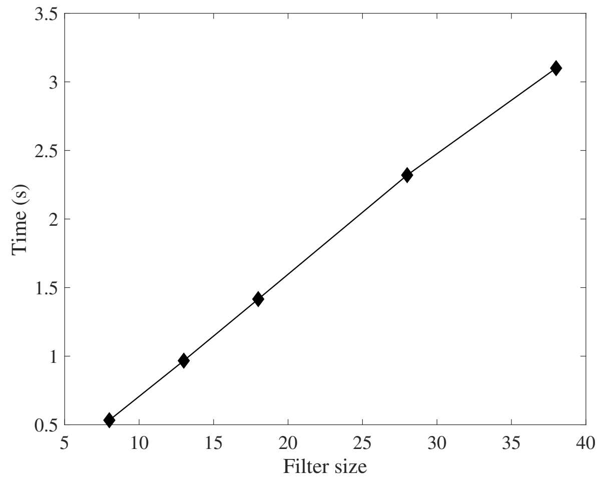


Fig. 7. Average batch time with different filter sizes on MAG.

graph convolutional methods, especially on large graphs, and showed better robustness and was more generalizable. STC also outperformed state-of-the-art methods in identifying essential proteins.

Acknowledgments

This work is supported by the Hong Kong Research Grants Council (Project 11200818), the Hong Kong Innovation and Technology Commission, and City University of Hong Kong (Project 9610460).

References

[1] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1ewdiR5tQ

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[3] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," *CoRR*, vol. abs/1611.02344, 2016. [Online]. Available: http://arxiv.org/abs/1611.02344

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[5] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *CoRR*, vol. abs/1312.6203, 2014.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3844–3852.

[8] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl&noteId=SJU4ayYgl

[9] M. Shi, Y. Tang, X. Zhu, and J. Liu, "Multi-label graph convolutional network representation learning," *IEEE Transactions on Big Data*, pp. 1–1, 2020.

[10] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: Graph sampling based inductive learning method," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BJe8pkHFwS

[11] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rytstxWAW

[12] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 4800–4810.

[13] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 941–949. [Online]. Available: http://proceedings.mlr.press/v80/chen18p.html

[14] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1416–1424. [Online]. Available: https://doi.org/10.1145/3219819.3219947

[15] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 4558–4567.

[16] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 257–266.

[17] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Accurate, efficient and scalable graph embedding," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019, pp. 462–471.

[18] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 1024–1034.

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

[20] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5425–5434.

[21] M. Ding, J. Tang, and J. Zhang, "Semi-supervised learning on graphs with generative adversarial nets," *CoRR*, vol. abs/1809.00130, 2018. [Online]. Available: http://arxiv.org/abs/1809.00130

[22] F. Monti, O. Shchur, A. Bojchevski, O. Litany, S. Günnemann, and M. M. Bronstein, "Dual-primal graph convolutional networks," *CoRR*, vol. abs/1806.00770, 2018. [Online]. Available: http://arxiv.org/abs/1806.00770

[23] Y. Zhang, X. Wang, C. Shi, X. Jiang, and Y. F. Ye, "Hyperbolic graph attention network," *IEEE Transactions on Big Data*, pp. 1–1, 2021.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] P. de Haan, T. Cohen, and M. Welling, "Natural graph networks," *arXiv preprint arXiv:2007.08349*, 07 2020. [Online]. Available: https://arxiv.org/abs/2007.08349v1

[26] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[27] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data articles," *AI Magazine*, vol. 29, pp. 93–106, 09 2008.

[28] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics (Oxford, England)*, vol. 33, no. 14, p. i190—i198, July 2017. [Online]. Available: https://europepmc.org/articles/PMC5870717

[29] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[30] M. Zeng, M. Li, Z. Fei, F. Wu, Y. Li, and Y. Pan, "A deep learning framework for identifying essential proteins based on protein-protein interaction network and gene expression data," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2018, pp. 583–588.

[31] C. Wu, H. Zhang, L. Zhang, and H. Zheng, "Identification of essential proteins using a novel multi-objective optimization method," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 1329–1333.

[32] H. Jeong, S. Mason, A.-L. Barabasi, and Z. Oltvai, "Lethality and centrality in protein networks," *Nature*, vol. 411, pp. 41–42, 06 2001.

[33] M. Joy, A. Brock, D. Ingber, and S. Huang, "High-betweenness proteins in the yeast protein interaction network," *Journal of biomedicine & biotechnology*, vol. 2005, pp. 96–103, 07 2005.

[34] S. Wuchty and P. Stadler, "Centers of complex networks," *Journal of theoretical biology*, vol. 223, pp. 45–53, 08 2003.

[35] P. Bonacich, "Power and centrality: A family of measures," *American Journal of Sociology*, vol. 92, pp. 1170–1182, 03 1987.

[36] J. Wang, M. Li, H. Wang, and Y. Pan, "Identification of essential proteins based on edge clustering coefficient," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 4, pp. 1070–1080, 2012.

[37] M. Li, J. Wang, X. Chen, H. Wang, and Y. Pan, "A local average connectivity-based method for identifying essential proteins from the network level," *Computational Biology and Chemistry*, vol. 35, no. 3, pp. 143 – 150, 2011.

[38] I. Xenarios, L. Salwínski, X. J. Duan, P. Higney, S.-M. Kim, and D. Eisenberg, "DIP, the Database of Interacting Proteins: A research tool for studying cellular networks of protein interactions," *Nucleic Acids Research*, vol. 30, no. 1, pp. 303–305, 01 2002. [Online]. Available: https://doi.org/10.1093/nar/30.1.303

[39] H. W. Mewes, D. Frishman, K. F. X. Mayer, M. Münsterkötter, O. Noubibou, P. Pagel, T. Rattei, M. Oesterheld, A. Ruepp, and V. Stümpflen, "MIPS: Analysis and annotation of proteins from whole genomes in 2005," *Nucleic Acids Research*, vol. 34, no. suppl_1, pp. D169–D172, 01 2006. [Online]. Available: https://doi.org/10.1093/nar/gkj148

[40] J. M. Cherry, C. Adler, C. Ball, S. A. Chervitz, S. S. Dwight, E. T. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein, "SGD: Saccharomyces Genome Database ," *Nucleic Acids Research*, vol. 26, no. 1, pp. 73–79, 01 1998. [Online]. Available: https://doi.org/10.1093/nar/26.1.73

[41] R. Zhang and Y. Lin, "DEG 5.0, a database of essential genes in both prokaryotes and eukaryotes," *Nucleic Acids Research*, vol. 37, no. suppl_1, pp. D455–D458, 10 2008. [Online]. Available: https://doi.org/10.1093/nar/gkn858

[42] "Saccharomyces genome deletion project," [http://yeastdeletion.stanford.edu//].1:32 2020/5/26 Accessed 20 June 2012.

[43] R. Edgar, M. Domrachev, and A. E. Lash, "Gene Expression Omnibus: NCBI gene expression and hybridization array data repository," *Nucleic Acids Research*, vol. 30, no. 1, pp. 207–210, 01 2002. [Online]. Available: https://doi.org/10.1093/nar/30.1.207

[44] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=Hkx1qkrKPr

[45] Q. Huang, H. He, A. Singh, S.-N. Lim, and A. R. Benson, "Combining label propagation and simple models out-performs graph neural networks," *arXiv preprint arXiv:2010.13993*, 2020.

[46] C. Deng, Z. Zhao, Y. Wang, Z. Zhang, and Z. Feng, "GraphZoom: A multi-level spectral approach for accurate and scalable graph embedding," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1lGO0EKDH

[47] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *CoRR*, vol. abs/1607.00653, 2016. [Online]. Available: http://arxiv.org/abs/1607.00653

[48] X. Zhu, "Semi-supervised learning literature survey," *world*, vol. 10, p. 10, 2005.

[49] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, and T. Goldstein, "FLAG: Adversarial data augmentation for graph neural networks," *arXiv preprint arXiv:2010.09891*, 2020.

[50] Y. Luo, A. Chen, K. Yan, and L. Tian, "Distilling self-knowledge from contrastive links to classify graph nodes without passing messages," *arXiv preprint arXiv:2106.08541*, 2021.

[51] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 135–144.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Chong Wu (S'19) was born in Zhejiang, China. He received a B.E. degree in automation from the School of Automation, China University of Geosciences, Wuhan, China, in 2018. He is currently pursuing a Ph.D. degree in electrical engineering with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong.

His current research interests include graph representation learning, image/video processing, and artificial intelligence.

Zhenan Feng received a B.E. degree and a M.E. degree from the School of Automation, China University of Geosciences, Wuhan, China. She is currently pursuing a Ph.D. degree in electrical engineering with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong.

Her current research interests include graph representation learning, image/video processing, motors and controls, design and optimization of electrical systems, and artificial intelligence.

Jiangbin Zheng was born in Taizhou, Zhejiang, China, in 1996. He received a M.E. degree in intelligent science and technology from the School of Informatics, Xiamen University in 2021. He is currently pursuing a Ph.D. degree with the School of Engineering, Westlake University, Hangzhou, China.

His current research interests include cross-modal sign language translation, machine translation and graph neural network.

Houwang Zhang (S'19) received a B.E. degree in industrial design from the School of Mechanical Engineering and Electronic Information, China University of Geosciences, Wuhan, China, in 2018. And then he received a M.E. degree in control science and engineering from the School of Automation, China University of Geosciences, Wuhan, China, in 2021. He is currently pursuing a Ph.D. degree in electrical engineering with the Department of Electrical Engineering, City University of Hong Kong, Hong Kong.

His current research interests include graph representation learning, image/video processing, and bioinformatics.

Jiawang Cao received a B.E. degree in automation from the School of Automation, China University of Geosciences, Wuhan, China, in 2019. He is currently pursuing a master's degree with the Academy of Engineering & Technology, Fudan University, Shanghai, China.

His current research interests include computer vision and deep learning.

Hong Yan received a Ph.D. degree from Yale University. He was a Professor of imaging science with the University of Sydney. He is currently the Chair Professor of computer engineering with the City University of Hong Kong. He was elected an IAPR Fellow for contributions to document image analysis and an IEEE Fellow for contributions to image recognition techniques and applications. He received the 2016 Norbert Wiener Award from the IEEE SMC Society for contributions to image and biomolecular pattern recognition techniques.

His current research interests include bioinformatics, image processing, and pattern recognition.