Star Topology Convolution for Graph Representation Learning

Star Topology Convolution for Graph Representation Learning

Chong Wu [*1] **Zhenan Feng** [*1] **Jiangbin Zheng** [2] **Houwang Zhang** [1] **Jiwang Cao** [3] **Hong Yan** [1]

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 space. Unlike most existing spectral methods, this method learns subgraphs which have a star topology rather than a fixed graph. Due to the good properties of a star topology, STC is graph/subgraph free. It has fewer parameters in its convolutional filter and is inductive so that it is more flexible and can be applied to large and evolving graphs. Similar to CNNs in Euclidean feature space, the convolutional filter is learnable and localized and maintains a good weight sharing property. To test the method, STC was compared to state-of-the-art spectral methods and spatial methods in a supervised learning setting on five benchmark datasets: Cora, Citeseer, Pubmed, Ogbn-Arxiv, and Ogbn-MAG. The experiment results show that STC outperforms other methods especially on large graphs. In an essential protein identification task, STC also outperforms state-of-the-art essential protein identification methods.

1. Introduction

Convolutional neural networks (CNNs) have been successfully used to solve problems which have a Euclidean feature space (Xu et al., 2019), such as image classification (He et al., 2016) and machine translation (Gehring et al., 2016). However most problems, such as 3D meshes, social networks, telecommunication networks, and biological networks, have a non-Euclidean nature (Veličković et al., 2018), which creates a challenge when introducing CNNs to solve these problems (Xu et al., 2019). Data in the form of a graph is a typical non-Euclidean problem. There are three major problems in generalizing CNNs to graphs: (1) the numbers of directly connected neighbors for different nodes usually differ (Bronstein et al., 2017); (2) the feature dimensions for different nodes may also differ; (3) the edges may have features and their feature dimensions may differ.

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 (Bruna et al., 2014). Then, some simplified versions of Spectral CNN which is based on polynomial expansion have been proposed like Chebyshev network (ChebyNet) (Defferrard et al., 2016) and graph convolutional network (GCN) (Kipf & Welling, 2017). ChebyNet (Defferrard et al., 2016) restricts the kernel of Spectral CNN to a polynomial expansion. GCN (Kipf & Welling, 2017) is a simplification of ChebyNet to make spectral methods spectrum-free, reduces the computational cost of the eigen decomposition of the graph Laplacian matrix and is able to get spatially localized filters (Xu et al., 2019). Because of its good performance and a faster speed of convolution computation, GCN is widely used in solving graph problems. But it has limitations on scaling to large graphs and is difficult to be trained in a minibatch setting (Zeng et al., 2020). To address the problem of scaling GCN to large graphs, layer sampling methods (Chen et al., 2018a; Ying et al., 2018; Chen et al., 2018b; Gao et al., 2018; Huang et al., 2018) and subgraph sampling methods (Chiang et al., 2019; Zeng et al., 2019; Zeng et al., 2020) have been proposed. They are designed for efficient minibatch training for GCN (Zeng et al., 2020). Layer sampling methods are based on a GCN on the full training graph. They just sample nodes or edges from the whole graph to form the minibatches in each layer and then do the forward and backward propagation on the sampled graph. Subgraph sampling methods which perform the subgraph sampling before training the GCN reduce the problem of large computational cost in the node/edge sampling in each layer (Chiang et al., 2019; Zeng et al., 2019; Zeng et al., 2020). However, the filter size of above spectral methods 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 this size has been changed, the trained model using the previous filter will need to be reconstructed and retrained. Although some flexible spatial methods (Hamilton

[*]Equal contribution [1]Department of Electrical Engineering, City University of Hong Kong, Kowloon [2]School of Informatics, Xiamen University, Xiamen, 361005, China [3]Academy of Engineering & Technology, Fudan University, Shanghai 200433, China. Correspondence to: Chong Wu <chongwu2-c@my.cityu.edu.hk>, Zhenan Feng <fengzhenan1996@outlook.com>.

Preprint version: For peer review.

et al., 2017) have been proposed, the aggregator they used is not learnable or convolutional. The problem of designing a filter which is flexible like spatial methods at the same time has an elegant 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 graphs, even of different sizes, will have some common eigenvectors, although they may be rotated to a certain degree. These properties allow the design of a graph/subgraph free (can be input different scales of graphs), learnable, and convolutional filter which can maintain weight sharing for subgraphs with 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 nodes of these subgraphs. Hence, this method can achieve localized filters in both spectral and spatial domains. The computation complexity and memory requirements are largely reduced compared to conventional spectral methods. As this method is inductive, it can be applied to large or evolving graphs. The convolutional process of this method is similar to that in the CNNs commonly used in image science. The contributions of this paper can be summarized as follows,

1. This is the first attempt to use star topology subgraph in graph representation learning.

2. This is a graph convolutional method which has a graph/subgraph free, learnable, and convolutional filter.

3. Based on the good properties of a star topology, this method has a good computation complexity with high memory efficiency.

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 the computer vision, natural language processing, speech processing, and *etc*, researchers began to generalize CNNs to graph domain. The key of generalizing CNNs to graphs is defining convolution operator on graphs (Xu et al., 2019). Existing graph convolutional neural networks can be classified into two broad categories: (1) spectral methods; (2) spatial methods.

Spectral methods define the convolution operator based on the graph Fourier transform (Hammond et al., 2011) 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 successfully applied to the context of node classification. Spectral convolutional neural network (Spectral CNN) (Bruna et al., 2014) introduces the graph Fourier transform directly and proposes spectral convolution for graph signals. But the number of learnable parameters of the filter is large, potentially causing severe computational cost (Veličković et al., 2018). Chebyshev network (ChebyNet) (Defferrard et al., 2016) restricts the kernel of Spectral CNN to a polynomial expansion. Graph convolutional network (GCN) (Kipf & Welling, 2017) further simplifies ChebyNet to make spectral methods spectrum-free, which reduce the computational cost of the eigen decomposition of the graph Laplacian matrix. GCN is able to get spatially and spectrally localized filters (Xu et al., 2019). Graph wavelet neural network (GWNN) introduces the graph wavelet transform to replace the Fourier basis of Spectral CNN with graph wavelet basis and is also able to achieve spatially and spectrally localized filters while maintaining a good computational performance. But all of above methods have limitations on scaling to large graphs and are difficult to be trained in a minibatch setting (Zeng et al., 2020). To scale the spectral methods to large graphs, some improved versions of GCN have been proposed (Chen et al., 2018a; Ying et al., 2018; Chen et al., 2018b; Gao et al., 2018; Huang et al., 2018). They are designed for efficient minibatch training for GCN. But they are based on a GCN on the whole training graph (Zeng et al., 2020). What's more, they just sample nodes or edges from the whole graph to form the minibatches in each layer iteratively (Zeng et al., 2020), which causes "neighbor explosion" problem and leads to a large computation complexity. To solve this problem, some heuristic based methods which make subgraph sampling as a preprocessing step have been proposed (Chiang et al., 2019; Zeng et al., 2019). But the sampling strategies of these methods introduce the non-identical node sampling probability and bias. To cancel the bias, graph sampling based inductive learning method (GraphSAINT) (Zeng et al., 2020) has been proposed. It develops a normalization technique so that the feature learning does not give larger weights to nodes which are sampled more frequently (Zeng et al., 2020). But above spectral methods still have a problem of generalization: they are not graph/subgraph free. The filter size of spectral methods is determined by the size of full graph or sampled subgraph. If this size is too large, it will result in a large computational and memory cost. And if the size has been changed, the trained model using the previous filter will need to be reconstructed and retrained.

Spatial methods define the convolution on the spatial domain. Mixture model CNN (MoNet) uses a weighted aver-

age of multiple weighting functions defined over the neighborhood of a node as the spatial convolution operator to provide a general framework for designing spatial methods (Monti et al., 2017). Graphs with generative adversarial nets (GraphSGAN) (Ding et al., 2018) facilitates generalization of generative adversarial nets (GANs) to 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 attention mechanism to graph domain. Graph attention network (GAT) adopts a self-attention mechanism to learn the weighting function (Veličković et al., 2018). Dual-primal graph convolutional network (DPGCN) (Monti et al., 2018) generalizes GAT by using convolutions on nodes and edges to achieve a better performance compared to GAT. Graph sample and aggregate method (GraphSage) (Hamilton et al., 2017), a node-based spatial method, learns node embedding rather than graph embedding so that it can be applied to large graphs or evolving graphs. It performs a uniform node sampling with a predefined sampling size to sample neighbors in each layer iteratively. The sampling size will result in bounding the minibatch computation complexity. It is a graph/subgraph free method. But unlike STC, its aggregator is not learnable although they have proposed a long-short term memory (LSTM) (Hochreiter & Schmidhuber, 1997) version which still has some defects like 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. $\boldsymbol{L}$ has an eigen decomposition as follows,

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

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 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 follows,

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

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

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

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 as follows,

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

where, $\boldsymbol{W}$ is a flexible filter for subgraphs with different topologies. 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, at the same time, provide different kernels for different structures (de Haan et al., 2020). 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 have an elegant symmetry in their structure and Laplacian matrix, as Figures 1 & 2 show. We also find that all graphs can be regarded as a composition of subgraphs with a star topology as Figure 2 shows. Then the eigen decomposition of an n-dim star topology (n neighboring nodes and one central node, $n \geq 1$) can use a universal formulation as follows,

$$\boldsymbol{L} = \begin{bmatrix} n & -1 & \cdots & -1 \\ -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & \cdots & 1 \end{bmatrix} = \boldsymbol{U} \Lambda \boldsymbol{U}^T,$$

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

where, all elements on the diagonal of Λ are 1 except the first and last elements, which means that all eigen vectors 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 as follows.

By modifying Equation (5), we can have,

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

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

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

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

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

the solutions of Equation (9) are Λ.

Eigenvectors $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n+1})$ are orthonormal. We can find that 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 as follows,

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

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.

Referring to Equation (3), we can find that the first row of $\boldsymbol{g}_\theta \boldsymbol{U}^T$ corresponds to the first row of the new feature matrix $\boldsymbol{g}_\theta \boldsymbol{U}^T \boldsymbol{F}$ and also to the first row of $\boldsymbol{F}$. The first row of $\boldsymbol{F}$ corresponds to the central node as $\boldsymbol{L}$ shows. Since we want to obtain the neighboring information 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. We can manually add an artificial node to the original subgraph, then we get a new $\boldsymbol{L} \in \mathbb{R}^{(n+2)*(n+2)}$. We can pad a zero row under the last row of $\boldsymbol{F}$ as zero rows in $\boldsymbol{F}$ won't affect the non-zero rows so that any number of zero rows in $\boldsymbol{F}$ can be added if needed. Then, we can design the

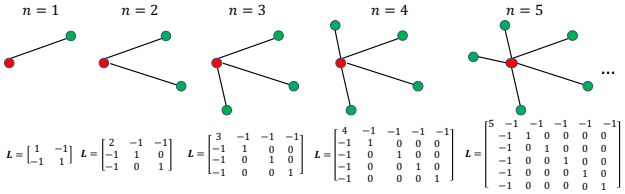

Figure 1. The structural similarity of graphs with a star topology. The red node is the central node.

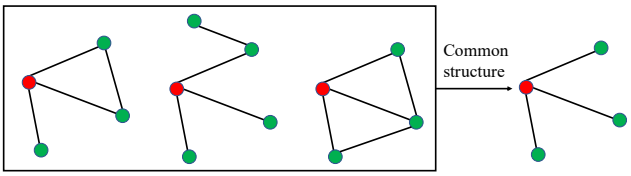

Figure 2. Subgraphs with a star topology exist in different graphs. The red node is the central node.

flexible filter for $\{1, 2, \ldots, k\}$-dim (k can be any non-zero positive integer) star topology subgraphs as follows,

$$\boldsymbol{W}(k) = \boldsymbol{U}\Theta, \tag{11}$$

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

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

Using the properties of a star topology subgraph, we can get the spectral convolution defined on that star topology. The update function for the $(l+1)^{th}$ layer's neighboring information can be defined as follows,

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

where, $y^l_{[\{V_j\}\in\mathbb{V}_i, x]}$ is the output of the l^{th} layer, x and z are the dimension indices, V_i is the central node, p and q are the dimensions of the embedding, and $\mathbb{V}_i$ is the set of directly connected neighbors of V_i. We use the detaching method (Xu et al., 2019) to reduce the computation complexity of

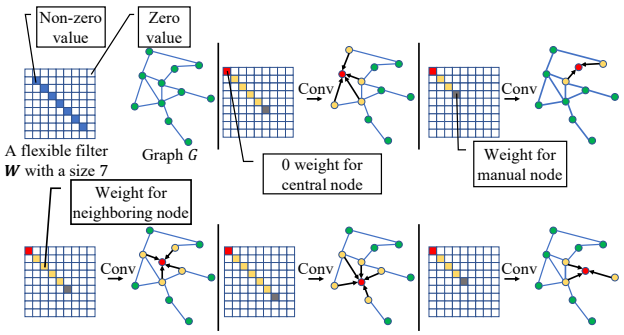

Figure 3. A toy example showing the weight sharing property of the filter on subgraphs with different star topologies. The red node is the central node and the yellow nodes are neighboring nodes to be aggregated. The kernels provided by the filter differ by subgraph. It is a general filter which can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

h^{l+1}. Differently from GWNN (Xu et al., 2019), we first perform spectral convolution as follows,

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

The information on the neighbors obtained from the spectral convolution will be aggregated to the central node using feature transformation as follows,

$$y^{l+1}_{V_i} = \sigma\left(\text{cat}\left(y^l_{[V_i,:]}, \underset{V_j}{\text{mean}}\left(h^{l+1}_{[\{V_j\}\in\mathbb{V}_i,:]}\right)\right)\boldsymbol{S}\right), \tag{15}$$

where, $y^l_{[V_i,:]}$ is the feature vector of the central node V_i, $\text{cat}()$ denotes a concatenation function, σ denotes a nonlinear activation function, and $\boldsymbol{S} \in \mathbb{R}^{2p*q}$ is a scaling parameter matrix for feature transformation. For the minibatch training setting, the batch normalization of $\underset{V_j}{\text{mean}}\left(h^{l+1}_{[\{V_j\}\in\mathbb{V}_i,:]}\right)$ and $y^l_{[V_i,:]}$ for all V_i in the minibatch is done before the concatenation. Then Equation (15) obtains the output y^{l+1}.

Figure 4 shows the comparison of a two-layer conventional CNN and a two-layer STC. The formulation of the two-layer STC is as follows,

$$\begin{aligned} &1^{st}\ layer: y^1_{V_i} = \\ &\quad \textbf{ReLU}\left(\text{cat}\left(\boldsymbol{F}_{[V_i,:]}, \underset{V_j}{\text{mean}}\left(h^1_{[\{V_j\}\in\mathbb{V}_i,:]}\right)\right)\boldsymbol{S}_1\right), \\ &2^{nd}\ layer: y^2_{V_i} = \\ &\quad \textbf{ReLU}\left(\text{cat}\left(y^1_{[V_i,:]}, \underset{V_j}{\text{mean}}\left(h^2_{[\{V_j\}\in\mathbb{V}_i,:]}\right)\right)\boldsymbol{S}_2\right), \end{aligned} \tag{16}$$

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 achieve spatially localized filters. Similar to the conventional CNN, our method can also obtain global information through the introduction of more layers. After the spatial search, our method conducts a spectral convolution on the subgraphs obtained. In a conventional CNN, the shape of different areas selected by a window function/kernel is usually the same. Take a 3*3 kernel for example, the different areas selected are all square and have the same size (9 pixels). In the graph, star topology subgraphs also have symmetry in their spatial structures which allows STC to do a similar convolution process as in a conventional CNN. If a new filter is used to replace the $\boldsymbol{W}$ here, a new inductive spectral convolutional method can be designed. Hence, our method provides a framework to design inductive spectral convolutional methods. If $\boldsymbol{W}$ becomes spatial aggregator like mean aggregator, our method becomes the GraphSage. Hence, our method also can be seen as a GraphSage with a learnable and convolutional aggregator.

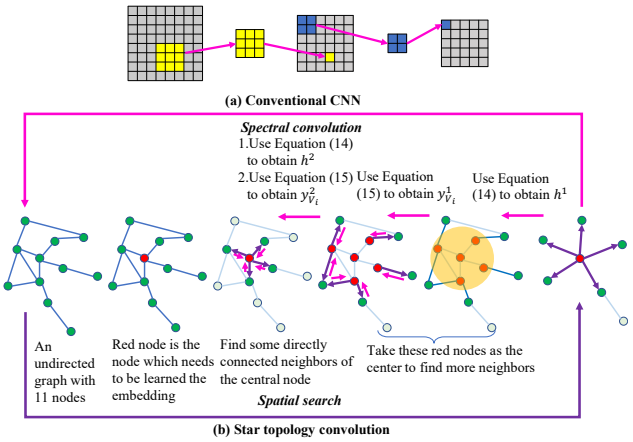

Figure 4. Comparison of a conventional CNN and a star topology convolution (STC). In the convolution process, CNN uses a grid filter to select some spatially localized areas from the training image for convolution. Similar to CNN, STC uses a universal filter $\boldsymbol{W}$ to select some spatially localized star topology subgraphs from the training graph for convolution. It first adopts a spatial search which is similar to GraphSage (Hamilton et al., 2017) to find some neighbors, the depth of neural network determines the order of neighbors used in the convolution, here is an example of a two-layer STC, hence only the first and the second order of neighbors are selected. Then it performs the spectral convolution on the selected subgraphs.

3.6. Node sampling

Since the filter size k is unknown, we need to confirm it in the actual implementation. A simple way to confirm the filter size is to just assign it as K, where K is the maximum

number of connections of a node in the graph. In our implementation, we use the node sampling strategy of GraphSage (Hamilton et al., 2017) to achieve a function of dropout (Hinton et al., 2012) to improve the robustness of our method. For a node V_i which has $|\mathbb{V}_i|$ directly connected neighbors, when $|\mathbb{V}_i| > k$, only k neighbors will be randomly selected in each training iteration as follows,

$$\begin{aligned} y^l_{[:,x]} &= y^l_{[\boldsymbol{ind},x]}, \\ \boldsymbol{ind} &= \text{randperm}(\{V_j\} \in \mathbb{V}_i, k),\ \{V_i, V_j\} \in \mathbb{E}, \end{aligned} \tag{17}$$

where, ***ind*** is the mask, the size of which is determined by the filter size k. For nodes with $|\mathbb{V}_i| \leq k$, all neighbors will be selected, and these nodes can be considered as nodes that do not have enough information. Hence, it is necessary to take advantage of all their information. Through controlling the filter size, our method can achieve a 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. Variant: Subgraph sampling

Subgraph sampling strategies have been proven to be an efficient minibatch training method to improve the performance of GCN (Chiang et al., 2019; Zeng et al., 2019; Zeng et al., 2020). We also introduce a subgraph sampling version of STC which is called STC-SubSam. In each training epoch, we use a node-based subgraph sampler to sample a set of connected subgraphs which takes the training nodes as the roots. Then, STC will be trained on these subgraphs rather than the full graph. This step generates more diverse training data for the training and enlarges the randomness of the training process. It will improve the robustness of the trained STC. Different to most existing spectral methods, STC is a graph/subgraph free method which means that it can use different scales of graphs as the input.

3.8. Complexity analysis

The computation complexity of STC depends on the filter size and the size of scaling matrix. The computation complexity of the $(l+1)^{th}$ layer is $p * q + k$, where $p * q$ is the size of scaling matrix and k is the filter size. k is much smaller than $p * q$, and, in our experiments, k is set to 18 and $p * q \geq 128 * 128$. Compared to other spectral methods like Spectral CNN (Bruna et al., 2014) (with a complexity of $p * q * |\mathbb{V}|$, where $|\mathbb{V}|$ is the total number of nodes in the full graph) and GWNN (Xu et al., 2019) (with a complexity of $p * q + |\mathbb{V}|$), the complexity of our method is much smaller and is able to be used in large or evolving graphs. Compared to subgraph sampling based GCNs, STC-SubSam does not need to perform subgraph sampling on the test set. As to the memory requirement, our method is an inductive node-based convolutional method and it learns node embedding instead of graph/subgraph embedding, hence, it does not need to store information of the full graph/subgraph in each layer and just needs to allocate the memory for the filter and the information of the central node and its neighbors.

4. Experiments and results

4.1. Experiment settings

To test our method, we compared STC and STC-SubSam with several state-of-the-art spectral methods: Spectral CNN (Bruna et al., 2014), GCN (Kipf & Welling, 2017), Spectral CNN detaching (SCNND) (Xu et al., 2019), GWNN (Xu et al., 2019) and GraphSAINT (Zeng et al., 2020), and spatial methods: GraphSage (Hamilton et al., 2017) and MoNet (Monti et al., 2017) in a supervised learning setting on five citation benchmark datasets: Cora (Sen et al., 2008), Citeseer (Sen et al., 2008), Pubmed (Sen et al., 2008), Ogbn-Arxiv (Hu et al., 2020), and Ogbn-MAG (Hu et al., 2020). What's more, we introduced STC and STC-SubSam in an essential protein identification task and compared STC and STC-SubSam with several state-of-the-art essential protein identification methods: deep neural network based method (DNN) (Zeng et al., 2018), support vector machine (SVM) (Zeng et al., 2018), Adaboost (Zeng et al., 2018), decision tree (DT) (Zeng et al., 2018), random forest (RF) (Zeng et al., 2018), multi-objective optimization based method (IMAMOBH) (Wu et al., 2020), degree centrality (DC) (Jeong et al., 2001), betweeness centrality (BC) (Joy et al., 2005), closeness centrality (CC) (Wuchty & Stadler, 2003), eigenvector centrality (EC) (Bonacich, 1987), edge clustering coefficient centrality (NC) (Wang et al., 2012), and local average connectivity (LAC) (Li et al., 2011). Essential protein identification can be regarded as a binary node classification task. The protein-protein interaction (PPI) data of Saccharomyces cerevisiae (Yeast) were downloaded from the DIP database (Xenarios et al., 2002) updated to Oct.10, 2010. There are 4512 proteins and 22091 interactions, without self and repeated interactions, in this dataset. We selected Yeast because its PPI data and gene essentiality data are the most complete and reliable among various species. The essential protein list was selected from MIPS (Mewes et al., 2006), SGD (Cherry et al., 1998), DEG (Zhang & Lin, 2008), and SGDP (SGD). There are 1285 essential proteins in the list, 1031 of which are in our PPI network. We took these 1031 proteins as essential proteins while the other 3481 proteins were considered as non-essential ones. The gene expression data used as the features were collected from GEO (Edgar et al., 2002), which sampled 36 time points during successive Yeast metabolic cycles. Table 3 in supplementary material shows the statistics of six datasets. The division of training/validation/test sets for Cora, Citeseer, and Pubmed is made according to (Rong et al., 2020). The division of training/validation/test sets of Protein dataset is set to 3012:500:1000. The division of

training/validation/test sets of Ogbn-Arxiv and Ogbn-MAG is made according to OGB paper (Hu et al., 2020). The Protein dataset including PPI data, labels, and features used in this paper is available in supplementary material.

Table 1. Comparison results of all graph convolutional methods in terms of test accuracy on Citeseer, Cora, and Pubmed, and comparison of the test accuracy of identification of essential proteins between ours and state-of-the-arts. "GS200" denotes GraphSAINT using 200 training epochs. The best values are in bold face and the second best values are in italic.

Methods	Citeseer	Cora	Pubmed	Protein
STC	0.7232	**0.8694**	**0.9020**	**0.7883**
STC-SubSam	0.7193	0.8583	*0.9017*	0.7857
GraphSAINT	0.6470	0.7580	0.8863	0.7467
GS200	0.6302	0.7025	0.8868	0.7378
GraphSage	**0.7366**	*0.8677*	0.8753	0.7770
MoNet	0.7007	0.7858	0.8395	0.7650
Spectral CNN	0.6840	0.8365	0.7920	0.6493
GCN	*0.7320*	0.8675	0.8990	*0.7880*
SCNND	0.6371	0.8072	0.8192	0.7241
GWNN	0.6958	0.7260	0.8940	0.7790
DNN	-	-	-	0.7640
IMAMOBH	-	-	-	0.7409
DT	-	-	-	0.6810
Adaboost	-	-	-	0.7620
RF	-	-	-	0.7650
SVM	-	-	-	0.7680
BC	-	-	-	0.7004
CC	-	-	-	0.7035
DC	-	-	-	0.7216
EC	-	-	-	0.7052
LAC	-	-	-	0.7425
NC	-	-	-	0.7345

All methods used their default settings in all experiments in this paper (GraphSage used in this paper is the GraphSage-mean). All methods were trained six runs (each run with a fixed number of epochs) independently per dataset without using a validation process to take their average F1-micro score on the test set as their test accuracy. All methods used the same random seed: 2. The architecture of all methods is kept the same: a two-layer encoder with a fully-connected layer as the decoder is used for all experiments in this paper as shown in Table 3 in supplementary material. The embedding dimensions of the two layers of the encoder for all methods are kept the same as shown in Table 3. The optimizer for STC is stochastic gradient descent and the initial learning rate is set to 0.7 for all experiments in this paper. The optimizers of state-of-the-art methods are used their default settings. The minibatch sizes for all datasets of the methods using minibatch training are kept the same as shown in Table 3. The number of epochs of all none subgraph sampling methods is kept the same as shown in Table 3. The number of epochs for subgraph sampling methods (STC-SubSam and GraphSAINT) is also kept the same as shown in Table 3. Subgraph sampling methods used the same node-based subgraph sampler with the same parameters. The parameters of the subgraph sampler are as the last two rows of Table 3 show. STC has a hyper-parameter: the size of filter. In the comparison with baseline methods, the filter size of the first STC layer and the filter size of the second STC layer are shown in Table 3. And the hyper-parameter of STC-SubSam is kept the same as STC. To validate the effect of this hyper-parameter on the performance of STC, we introduced an ablation study. The Pytorch version of STC used in this paper is available in supplementary material. All experiments except the ablation study and GPU memory cost test 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 1.7.1, 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 1.7.1, and 1 NVIDIA GeForce GTX 1660 Super GPU.

Table 2. Comparison results of 4 inductive methods (STC, STC-SubSam, GraphSAINT, and GraphSage) in terms of test accuracy on Ogbn-Arxiv and Ogbn-MAG using PC1 and comparison of 7 spectral methods (STC, STC-SubSam, GraphSAINT, Spectral CNN, GCN, SCNND, and GWNN) in terms of GPU memory cost on Pubmed in the model initialization process using PC2 (1*GTX 1660 Super). "GS200" and "GS2000" denote GraphSAINT using 200 and 2000 training epochs respectively. The best values are in bold face and the second best values are in italic.

Methods	Arxiv	MAG	GPU MEM cost
STC	*0.6034*	*0.2863*	*376.8* Mb
STC-SubSam	**0.6501**	**0.3099**	398.0 Mb
GraphSAINT	0.5902	0.2674	**337.6** Mb
GS200	0.6014	-	**337.6** Mb
GS2000	-	0.2844	**337.6** Mb
GraphSage	0.5428	0.2813	-
Spectral CNN	-	-	Out of memory
GCN	-	-	398.4 Mb
SCNND	-	-	1949.6 Mb
GWNN	-	-	Out of memory

4.2. Performance analysis

Table 1 shows the test accuracy obtained by all graph convolutional methods on Cora, Citeseer, and Pubmed. STC achieves the highest accuracy on Cora and Pubmed. GraphSage achieves the highest accuracy on Citeseer and the second highest accuracy on Cora. STC-SubSam achieves the second highest accuracy on Pubmed. The difference between the performance of GraphSage and the performance

of STC/STC-SubSam on small graphs is not significant. STC/STC-SubSam achieves a state-of-the-art performance on small graphs. Compared to our methods, GraphSAINT performs not good at small graphs. Table 1 also shows the comparison of our methods with some state-of-the-art essential protein identification methods. STC significantly outperforms these methods, especially some advanced methods like DNN, LAC, and IMAMOBH. GCN and STC-SubSam have a test accuracy only slightly lower than STC in this dataset which shows that graph convolutional methods have some advantages in solving traditional graph problems. Table 2 shows the test accuracy obtained by four inductive methods (STC, STC-SubSam, GraphSAINT, and GraphSage) on Ogbn-Arxiv and Ogbn-MAG. STC and STC-SubSam achieve the second highest accuracy and the highest accuracy on both two graphs respectively. STC-SubSam significantly outperforms GraphSAINT and GraphSage on large graphs. STC-SubSam achieves a higher accuracy on large graphs than STC. Since subgraph sampling based methods: STC-SubSam and GraphSAINT only used 10 training epochs, to ensure the fairness, we also introduced a GraphSAINT using 200 epochs (GS200) on Citeseer, Cora, Pubmed, Protein, and Arxiv, and a GraphSAINT using 2000 epochs (GS2000) on MAG for comparison. Through introducing more training epochs, GraphSAINT achieves a better performance on large graphs as Table 2 shows. But it achieves a worse performance on small graphs which can be explained by overfitting as Table 1 shows. Our methods still outperform GS200 and GS2000 on large graphs. STC-SubSam was trained on the sampled subgraphs and tested on the full graph which means that STC-SubSam have learned the "global vision" from some subgraphs of the full graph. It can happen because STC is a graph/subgraph free spectral convolutional method which has no requirement of the input size. Table 2 also shows the comparison of GPU memory cost of 7 spectral methods in the model initialization process on Pubmed using PC2. STC/STC-SubSam achieves a state-of-the-art performance in terms of GPU memory cost.

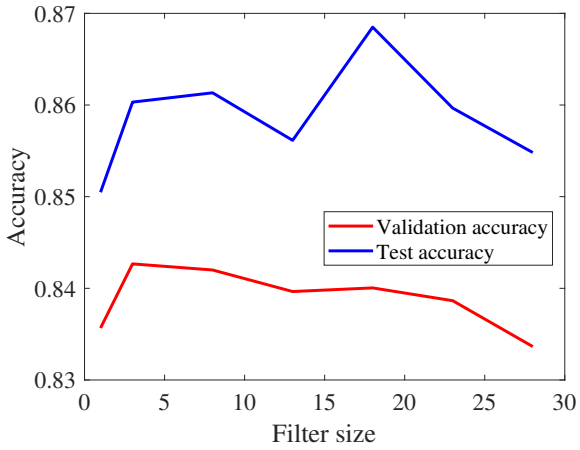

Figure 5. Mean accuracy of 6 runs at different filter sizes on Cora.

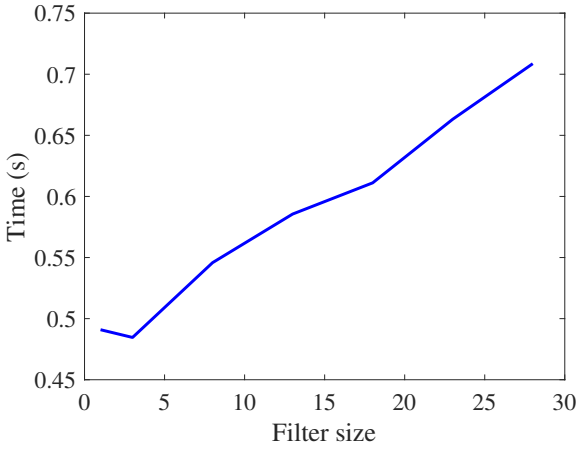

Figure 6. Average batch time with different filter size on Cora.

4.3. Effect of filter size

To validate the effect of filter size on the performance of STC, we set different sizes of the filter in the first STC layer and kept the size of the filter in the second STC layer unchanged and validated on Cora. Figure 5 shows the test accuracy and validation accuracy obtained by our prediction model with different filter sizes in its first STC layer. Validation accuracy and test accuracy improve with increasing filter size to a peak value and then reduce in a general trend. When the filter size is too small, there is insufficient information to train a good model. When the filter size is too large, redundant information will be introduced to the trained model, causing overfitting and leading to a degradation in performance by the model. Figure 6 shows the average batch time obtained by our prediction model with different filter sizes in its first STC layer on Cora. Running time is linear with respect to filter size.

5. Conclusion

In this paper, we present a novel graph convolutional method, called star topology convolution (STC), which is a graph/subgraph free inductive spectral convolutional method. It learns node embedding on subgraphs within a star topology. In experiments, our method outperforms state-of-the-art graph convolutional methods especially on the large graphs. In the task of identification of essential proteins, our method also outperforms state-of-the-art essential protein identification methods.

6. 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

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

Bonacich, P. Power and centrality: A family of measures. *American Journal of Sociology*, 92:1170–1182, 03 1987. doi: 10.1086/228631.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, 2017.

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

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=rytstxWAW.

Chen, J., Zhu, J., and Song, L. Stochastic training of graph convolutional networks with variance reduction. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 941–949. PMLR, 2018b. URL http://proceedings.mlr.press/v80/chen18p.html.

Cherry, J. M., Adler, C., Ball, C., Chervitz, S. A., Dwight, S. S., Hester, E. T., Jia, Y., Juvik, G., Roe, T., Schroeder, M., Weng, S., and Botstein, D. SGD: Saccharomyces Genome Database . *Nucleic Acids Research*, 26(1):73–79, 01 1998. ISSN 0305-1048. doi: 10.1093/nar/26.1.73. URL https://doi.org/10.1093/nar/26.1.73.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. 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*, KDD '19, pp. 257–266, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330925.

de Haan, P., Cohen, T., and Welling, M. Natural graph networks. *arXiv preprint arXiv:2007.08349*, 07 2020. URL https://arxiv.org/abs/2007.08349v1.

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

Ding, M., Tang, J., and Zhang, J. Semi-supervised learning on graphs with generative adversarial nets. *CoRR*, abs/1809.00130, 2018. URL http://arxiv.org/abs/1809.00130.

Edgar, R., Domrachev, M., and Lash, A. E. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1):207–210, 01 2002. ISSN 0305-1048. doi: 10.1093/nar/30.1.207. URL https://doi.org/10.1093/nar/30.1.207.

Gao, H., Wang, Z., and Ji, S. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pp. 1416–1424, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3219947. URL https://doi.org/10.1145/3219819.3219947.

Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. N. A convolutional encoder model for neural machine translation. *CoRR*, abs/1611.02344, 2016. URL http://arxiv.org/abs/1611.02344.

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

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129 – 150, 2011. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2010.04.005.

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

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Improving neural networks by

preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. URL http://arxiv.org/abs/1207.0580.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

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

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

Jeong, H., Mason, S., Barabasi, A.-L., and Oltvai, Z. Lethality and centrality in protein networks. *Nature*, 411:41–42, 06 2001. doi: 10.1038/35075138.

Joy, M., Brock, A., Ingber, D., and Huang, S. High-betweenness proteins in the yeast protein interaction network. *Journal of biomedicine & biotechnology*, 2005: 96–103, 07 2005. doi: 10.1155/JBB.2005.96.

Kipf, T. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl&noteId=SJU4ayYgl.

Li, M., Wang, J., Chen, X., Wang, H., and Pan, Y. A local average connectivity-based method for identifying essential proteins from the network level. *Computational Biology and Chemistry*, 35(3):143 – 150, 2011. ISSN 1476-9271. doi: https://doi.org/10.1016/j.compbiolchem.2011.04.002.

Mewes, H. W., Frishman, D., Mayer, K. F. X., Münsterkötter, M., Noubibou, O., Pagel, P., Rattei, T., Oesterheld, M., Ruepp, A., and Stümpflen, V. MIPS: Analysis and annotation of proteins from whole genomes in 2005. *Nucleic Acids Research*, 34(suppl_1):D169–D172, 01 2006. ISSN 0305-1048. doi: 10.1093/nar/gkj148. URL https://doi.org/10.1093/nar/gkj148.

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

Monti, F., Shchur, O., Bojchevski, A., Litany, O., Günnemann, S., and Bronstein, M. M. Dual-primal graph convolutional networks. *CoRR*, abs/1806.00770, 2018. URL http://arxiv.org/abs/1806.00770.

Rong, Y., Huang, W., Xu, T., and Huang, J. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkx1qkrKPr.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. Collective classification in network data articles. *AI Magazine*, 29:93–106, 09 2008. doi: 10.1609/aimag.v29i3.2157.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Wang, J., Li, M., Wang, H., and Pan, Y. Identification of essential proteins based on edge clustering coefficient. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):1070–1080, 2012.

Wu, C., Zhang, H., Zhang, L., and Zheng, H. 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)*, pp. 1329–1333, 2020.

Wuchty, S. and Stadler, P. Centers of complex networks. *Journal of theoretical biology*, 223:45–53, 08 2003. doi: 10.1016/S0022-5193(03)00071-7.

Xenarios, I., Salwínski, L., Duan, X. J., Higney, P., Kim, S.-M., and Eisenberg, D. DIP, the Database of Interacting Proteins: A research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30(1): 303–305, 01 2002. ISSN 0305-1048. doi: 10.1093/nar/30.1.303. URL https://doi.org/10.1093/nar/30.1.303.

Xu, B., Shen, H., Cao, Q., Qiu, Y., and Cheng, X. Graph wavelet neural network. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1ewdiR5tQ.

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

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Accurate, efficient and scalable graph embedding. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 462–471, 2019. doi: 10.1109/IPDPS.2019.00056.

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

Zeng, M., Li, M., Fei, Z., Wu, F., Li, Y., and Pan, Y. 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)*, pp. 583–588, 2018.

Zhang, R. and Lin, Y. DEG 5.0, a database of essential genes in both prokaryotes and eukaryotes. *Nucleic Acids Research*, 37(suppl_1):D455–D458, 10 2008. ISSN 0305-1048. doi: 10.1093/nar/gkn858. URL https://doi.org/10.1093/nar/gkn858.

7. Supplementary material

Code and datasets are available in https://github.com/Espera-fzn/Star_Topology_Convolution/.

Table 3. The statistics of six benchmark datasets and the training settings for each method. In the row of architecture, 1 means convolution layer; 0 means fully-connected layer. In the row of subgraph sampling training epoch,"STC" denotes STC-SubSam and "GS" denotes GraphSAINT.

	Citeseer	Cora	Pubmed	Protein	Ogbn-Arxiv	Ogbn-MAG
Nodes	3327	2708	19717	4512	169343	1939743
Edges	4732	5429	44338	22091	1166243	21111007
Classes	6	7	3	2	40	349
Features	3703	1433	500	36	128	128
Embedding dimension for the 1st layer	128	128	128	128	128	256
Embedding dimension for the 2nd layer	128	128	128	128	128	256
Training set	1827	1208	18217	3012	90941	629571
Test set	1000	1000	1000	1000	48603	41939
Validation set	500	500	500	500	29799	64879
Minibatch size	100	100	512	100	1000	200
Architecture	1-1-0	1-1-0	1-1-0	1-1-0	1-1-0	1-1-0
None subgraph sampling training epoch	200	200	200	200	200	2000
Filter size of the 1st STC layer	18	18	18	18	18	18
Filter size of the 2nd STC layer	18	18	18	18	18	18
Subgraph sampling training epoch	STC: 10; GS: 10 for all, 200 for Citeseer, Cora, Pubmed, Protein, and Arxiv, 2000 for MAG					
Sample coverage	20	20	20	20	10	1
Subgraph size	100	100	512	100	1000	200