Graph Convolutional Networks using Heat Kernel for Semi-supervised Learning

Bingbing Xu, **Huawei Shen**, **Qi Cao**, **Keting Cen** and **Xueqi Cheng**

CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences
School of Computer Science and Technology,
University of Chinese Academy of Sciences, Beijing, China
{xubingbing, shenhuawei, caoqi, cenketing, cxq}@ict.ac.cn

Abstract

Graph convolutional networks gain remarkable success in semi-supervised learning on graph-structured data. The key to graph-based semi-supervised learning is capturing the smoothness of labels or features over nodes exerted by graph structure. Previous methods, spectral methods and spatial methods, devote to defining graph convolution as a weighted average over neighboring nodes, and then learn graph convolution kernels to leverage the smoothness to improve the performance of graph-based semi-supervised learning. One open challenge is how to determine appropriate neighborhood that reflects relevant information of smoothness manifested in graph structure. In this paper, we propose GraphHeat, leveraging heat kernel to enhance low-frequency filters and enforce smoothness in the signal variation on the graph. GraphHeat leverages the local structure of target node under heat diffusion to determine its neighboring nodes flexibly, without the constraint of order suffered by previous methods. GraphHeat achieves state-of-the-art results in the task of graph-based semi-supervised classification across three benchmark datasets: Cora, Citeseer and Pubmed.

1 Introduction

Convolutional neural networks (CNNs) [LeCun *et al.*, 1998] have been successfully used in various machine learning problems, such as image classification [He *et al.*, 2016] and speech recognition [Hinton *et al.*, 2012], where there is an underlying Euclidean structure. However, in many research areas, data are naturally located in a non-Euclidean space, with graph or network being one typical case. The success of CNNs motivates researchers to design convolutional neural network on graphs. Existing methods fall into two categories, spatial methods and spectral methods, according to the way that convolution is defined.

Spatial methods define convolution directly in the vertex domain, following the practice of the conventional CNN. For each node, convolution is defined as a weighted average function over its neighboring nodes, with the weighting function characterizing the influence exerting to the tar-

get node by its neighboring nodes. GraphSAGE [Hamilton *et al.*, 2017] defines the weighting function as various aggregators over neighboring nodes. Graph attention network (GAT) proposes learning the weighting function via self-attention mechanism [Velickovic *et al.*, 2017]. MoNet [Monti *et al.*, 2017] offers us a general framework for designing spatial methods. It takes convolution as a weighted average of multiple weighting functions defined over neighboring nodes. One open challenge for spatial methods is how to determine appropriate neighborhood for target node when defining graph convolution.

Spectral methods define graph convolution via convolution theorem. As the pioneering work of spectral methods, Spectral CNN [Bruna *et al.*, 2014] leverages graph Fourier transform to convert signals defined in vertex domain into spectral domain, and defines the convolution kernel as a set of learnable coefficients associated with Fourier bases, i.e., the eigenvectors of Laplacian matrix. Given that the magnitude of eigenvalue reflects the smoothness over graph of the associated eigenvector, only the eigenvectors associated with smaller eigenvalues are used. Unfortunately, this method relies on the eigendecomposition of Laplacian matrix, resulting in high computational complexity. ChebyNet [Defferrard *et al.*, 2016] introduces a polynomial parametrization to convolution kernel, i.e., convolution kernel is taken as a polynomial function of the diagonal matrix of eigenvalues. Subsequently, Kipf and Welling [Kipf and Welling, 2017] proposed graph convolutional network (GCN) via a localized first-order approximation to ChebyNet. However, both ChebyNet and GCN fail to filter out high-frequency noise carried by eigenvectors associated with high eigenvalues. GWNN [Xu *et al.*, 2019] leverages graph wavelet to implement localized convolution. In sum, to the best of our knowledge, previous works lack an effective graph convolution method to capture the smoothness manifested in the structure of network.

In this paper, we propose graph convolutional network with heat kernel, namely GraphHeat, for graph-based semi-supervised learning. Different from existing spectral methods, GraphHeat uses heat kernel to assign larger importance to low-frequency filters, explicitly discounting the effect of high-frequency variation of signals on graph. In this way, GraphHeat performs well at capturing the smoothness of labels or features over nodes exerted by graph structure. From the perspective of spatial methods, GraphHeat leverages the

process of heat diffusion to determine neighboring nodes that reflect the local structure of the target node and the relevant information of smoothness manifested in graph structure. Experiments show that GraphHeat achieves state-of-the-art results in the task of graph-based semi-supervised classification across benchmark datasets: Cora, Citeseer and Pubmed.

2 Preliminary

2.1 Graph Definition

$G = \{V, E, A\}$ denotes an undirected graph, where V is the set of nodes with $|V| = n$, E is the set of edges, and A is the adjacency matrix with $A_{i,j} = A_{j,i}$ to define the connection between node i and node j. Graph Laplacian matrix is defined as $\mathcal{L} = D - A$ where D is a diagonal degree matrix with $D_{i,i} = \sum_j A_{i,j}$, normalized Laplacian matrix $L = I_n - D^{-1/2} A D^{-1/2}$ where I_n is the identity matrix. Since L is a real symmetric matrix, it has a complete set of orthonormal eigenvectors $U = (u_1, u_2, ..., u_n)$, known as Laplacian eigenvectors. These eigenvectors have associated real, non-negative eigenvalues $\{\lambda_l\}_{l=1}^n$, identified as the frequencies of the graph. Without loss of generality, we have $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Using eigenvectors and eigenvalues of L, we have $L = U\Lambda U^\top$, where $\Lambda = \text{diag}(\{\lambda_l\}_{l=1}^n)$.

2.2 Graph Fourier Transform

Taking the eigenvectors of normalized Laplacian matrix as a set of bases, graph Fourier transform of a signal $x \in R^n$ on graph G is defined as $\hat{x} = U^\top x$, and the inverse graph Fourier transform is $x = U\hat{x}$ [Shuman *et al.*, 2013]. According to convolution theorem, graph Fourier transform offers us a way to define the graph convolution operator, represented as $*_G$. f denotes the convolution kernel in spatial domain, and $*_G$ is defined as

$$x *_G f = U\big((U^\top f) \odot (U^\top x)\big), \quad (1)$$

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

2.3 Graph Convolutional Networks

With the vector $U^\top f$ replaced by a diagonal matrix g_θ, Hadamard product can be written in the form of matrix multiplication. Filtering the signal x by convolution kernel g_θ, Spectral CNN [Bruna *et al.*, 2014] is obtained as

$$y = U g_\theta U^\top x = (\theta_1 u_1 u_1^\top + \theta_2 u_2 u_2^\top + \cdots + \theta_n u_n u_n^\top)x, \quad (2)$$

where $g_\theta = \text{diag}(\{\theta_i\}_{i=1}^n)$ is defined in spectral domain.

The above non-parametric convolution kernel is not localized in space and its parameter complexity scales up to $O(n)$. To combat these issues, ChebyNet [Defferrard *et al.*, 2016] is proposed to parameterize g_θ with a polynomial expansion

$$g_\theta = \sum_{k=0}^{K-1} \alpha_k \Lambda^k, \quad (3)$$

where K is a hyper-parameter and the parameter α_k is the polynomial coefficient. Then graph convolution operation in ChebyNet is defined as

$$
\begin{aligned}
y &= U g_\theta U^\top x = U(\sum_{k=0}^{K-1} \alpha_k \Lambda^k) U^\top x \\
&= \{\alpha_0 I + \alpha_1(\lambda_1 u_1 u_1^\top + \cdots + \lambda_n u_n u_n^\top) \\
&\quad + \cdots + \alpha_{K-1}(\lambda_1^{K-1} u_1 u_1^\top + \cdots + \lambda_n^{K-1} u_n u_n^\top)\}x \\
&= (\alpha_0 I + \alpha_1 L + \alpha_2 L^2 + \cdots + \alpha_{K-1} L^{K-1})x.
\end{aligned}
\quad (4)
$$

GCN [Kipf and Welling, 2017] simplifies ChebyNet by only considering the first-order polynomial approximation, i.e., $K = 2$, and setting $\alpha = \alpha_0 = -\alpha_1$. Graph convolution is defined as

$$
\begin{aligned}
y &= U g_\theta U^\top x = U(\sum_{k=0}^{1} \alpha_k \Lambda^k) U^\top x \\
&= \{\alpha I - \alpha(\lambda_1 u_1 u_1^\top + \cdots + \lambda_n u_n u_n^\top)\}x \\
&= \alpha(I - L)x.
\end{aligned}
\quad (5)
$$

Note that the above three methods all take $\{u_i u_i^\top\}_{i=1}^n$ as a set of basic filters. Spectral CNN directly learns the coefficients of each filter. ChebyNet and GCN parameterize the convolution kernel g_θ to get combined filters, e.g., L, accelerating the computation and reducing parameter complexity.

3 GraphHeat for Semi-supervised Learning

Graph smoothness is the prior information over graph that connected nodes tend to have the same label or similar features. Graph-based semi-supervised learning, e.g., node classification, gains success via leveraging the smoothness of labels or features over nodes exerted by graph structure. Graph convolutional neural network offers us a promising and flexible framework for graph-based semi-supervised learning. In this section, we analyze the weakness of previous graph convolution neural networks at capturing smoothness manifested in graph structure, and propose GraphHeat to circumvent the problem of graph-based semi-supervised learning.

3.1 Motivation

Given a signal x defined on graph, its smoothness with respect to the graph is measured using

$$x^\top L x = \sum_{(a,b) \in E} A_{a,b} \big(\frac{x(a)}{\sqrt{d_a}} - \frac{x(b)}{\sqrt{d_b}}\big)^2, \quad (6)$$

where $a, b \in V$, $x(a)$ represents the value of signal x on node a, d_a is the degree of node a. The smoothness of signal x characterizes how likely it has similar values, normalized by degree, on connected nodes.

For an eigenvector u_i of the normalized Laplacian matrix L, its associated eigenvalue λ_i captures the smoothness of u_i [Shuman *et al.*, 2013] as

$$\lambda_i = u_i^\top L u_i. \quad (7)$$

According to Eq. (7), eigenvectors associated with small eigenvalues are smooth with respect to graph structure, i.e., they carry low-frequency variation of signals on graph. In

contrast, eigenvectors associated with large eigenvalues are signals with high-frequency, i.e., less smoother signals.

Note that the eigenvectors $U = (u_1, u_2, \cdots, u_n)$ are orthonormal, i.e.,

$$u_i^\top u_i = 1 \quad \text{and} \quad u_i^\top u_j = 0, \ j \neq i. \tag{8}$$

Given a signal x defined on graph, we have

$$x = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_n u_n, \tag{9}$$

where $\{\alpha_i\}_{i=1}^n$ is the coefficient of each eigenvector. Each eigenvector u_i offers us a basic filter, i.e., $u_i u_i^T$. Only the component $\alpha_i u_i$ of x can pass the filter:

$$u_i u_i^\top x = \alpha_1 u_i u_i^\top u_1 + \cdots + \alpha_n u_i u_i^\top u_n = \alpha_i u_i. \tag{10}$$

Therefore, $\{u_i u_i^\top\}_{i=1}^n$ form a set of basic filters, and the eigenvalue associated with u_i represents the frequency of signals that can pass the filter $u_i u_i^\top$, i.e., $\alpha_i u_i$.

We now use these basic filters to analyze the weakness of previous methods at capturing the smoothness manifested in graph structure. Spectral CNN, Eq. (2), directly learns the coefficients of each basic filter to implement graph convolution. ChebyNet, Eq. (4), defines graph convolution based on a set of combined filters, i.e., $\{L^k\}_{k=0}^{K-1}$, which are obtained through assigning higher weights to high-frequency basic filters. For example, for a combined filter L^k, the weight assigned to $u_i u_i^T$ is λ_i^k, which increases with respect to λ_i. GCN only uses the first order approximation, i.e., L, for semi-supervised learning. In sum, the three methods do not suppress high-frequency signals by assigning lower weight to high-frequency basic filters, failing to or not well capture the smoothness manifested in graph structure.

3.2 GraphHeat: Graph Convolutional using Heat Kernel

We propose GraphHeat to capture the smoothness of labels or features over nodes exerted by graph structure. GraphHeat comprises a set of combined filters which discount high-frequency basic filters via heat kernel (see [Chung and Graham, 1997], Chapter10).

Heat kernel is defined as

$$f(\lambda_i) = e^{-s\lambda_i}, \tag{11}$$

where $s \geq 0$ is a scaling hyper-parameter. For clarity, we denote with Λ_s as $\Lambda_s = \text{diag}(\{e^{-s\lambda_i}\}_{i=1}^n)$. Using heat kernel, we define the convolution kernel as

$$g_\theta = \sum_{k=0}^{K-1} \theta_k (\Lambda_s)^k, \tag{12}$$

where θ_k is the parameter.

For a signal x, graph convolution is achieved by

$$
\begin{aligned}
y &= U g_\theta U^\top x = U\left(\sum_{k=0}^{K-1} \theta_k (\Lambda_s)^k\right) U^\top x \\
&= \{\theta_0 I + \theta_1 (e^{-s\lambda_1} u_1 u_1^\top + \cdots + e^{-s\lambda_n} u_n u_n^\top) + \cdots \\
&\quad + \theta_{K-1}(e^{-(K-1)s\lambda_1} u_1 u_1^\top + \cdots + e^{-(K-1)s\lambda_n} u_n u_n^\top)\} x \\
&= (\theta_0 I + \theta_1 e^{-sL} + \theta_2 e^{-2sL} + \cdots + \theta_{K-1} e^{-(K-1)sL}) x.
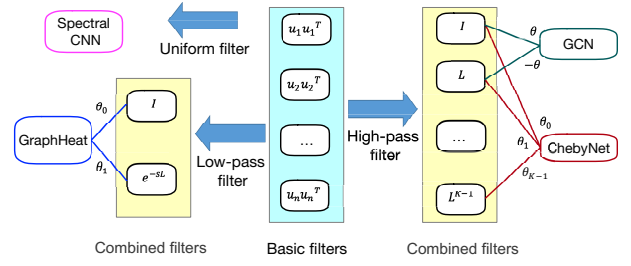\end{aligned}
\tag{13}
$$

Figure 1: The connection and difference between GraphHeat and previous graph convolution methods. GraphHeat captures the smoothness over graph by suppressing high-frequency signals, acting like a low-pass filter. In contrast, previous methods can be viewed as high-pass filters or uniform filters.

The key insight of GraphHeat is that it achieves a smooth graph convolution via discounting high-frequency basic filters. The weight assigned to the basic filter $u_i u_i^\top$ is $e^{-ks\lambda_i}$, which decreases with respect to λ_i. This distinguishes GraphHeat from existing methods that promotes high-frequency basic filters.

To reduce parameter complexity for semi-supervised learning, we only retain the first two items in Eq. (13)

$$y = (\theta_0 I + \theta_1 e^{-sL}) x. \tag{14}$$

The computation of e^{-sL} is completed via Chebyshev polynomials without eigendecomposition of L [Hammond *et al.*, 2011]. The computational complexity is $O(m \times |E|)$, where $|E|$ is the number of edges, and m is the order of Chebyshev polynomials. Such a linear complexity makes GraphHeat applicable to large-scale networks.

Figure 1 illustrates the connection and difference between our GraphHeat and previous graph convolution methods. GraphHeat captures the smoothness over graph by suppressing high-frequency signals, acting like a low-pass filter. In contrast, previous methods can be viewed as high-pass filters or uniform filters, lacking the desirable capability to filter out high-frequency variations over graph. Therefore, GraphHeat is expected to perform better on graph-based semi-supervised learning.

3.3 GraphHeat: Defining Neighboring Nodes under Heat Diffusion

In the previous subsection, GraphHeat is formulated as a kind of spectral method for graph convolution. The benefit of GraphHeat is also highlighted through comparing it with existing spectral methods from the perspective of frequency and graph signal processing. Now we offer an understanding of GraphHeat by taking it as a kind of spatial method.

Spatial methods define graph convolution as a weighted average over the neighboring nodes of target node. A general framework for spatial methods is to define graph convolution as a weighted average of a set of weighting functions, with each weighting function characterizing certain influence exerting to the target node by its neighboring nodes [Monti *et al.*, 2017]. For GraphHeat, the weighting functions are defined via heat kernels, i.e., e^{-ksL}, and graph convolution kernel is the coefficients θ_k.
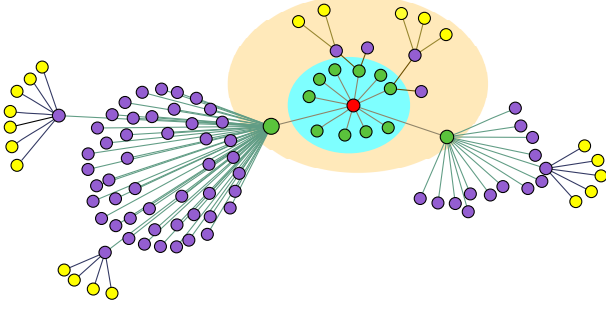
Figure 2: Neighboring nodes defined by GraphHeat and previous methods that are based on shortest-path distance. Target node is colored in red, and the 1-st, 2-nd, 3-rd order neighboring nodes are colored in green, purple, and yellow, respectively. The small circle marks neighboring nodes of GraphHeat with a small s. The big circle includes neighboring nodes when s is larger.

In GraphHeat, each e^{-ksL} actually corresponds to a similarity metric among nodes under heat diffusion. The similarity between node i and node j characterizes the amount of energy received by node j when a unit of heat flux or energy is offered to node i, and vice versa. Indeed, the scaling parameter s acts like the length of time during which diffusion proceeds, and k represents different energy levels.

With the heat diffusion similarity, GraphHeat offers us a way to define neighboring nodes for target node. Specifically, for a target node, nodes with the similarity higher than a threshold ϵ are regarded as neighboring nodes. Such a way of defining neighboring nodes is fundamentally different from previous graph convolution methods, which usually adopts an order-style way, i.e., neighboring nodes are within K-hops away from target node.

Figure 2 illustrates the difference between the two kinds of ways for defining neighboring nodes. Previous methods define neighboring nodes according to the shortest path distance, K, away from the target node (the red node in Figure 2). When $K = 1$, only green nodes are included, and this may lead to ignoring some relevant nodes. When $K = 2$, all purple neighbors are included. This may bring noise to the target node, since connections with high-degree nodes may represent popularity of high-degree nodes instead of correlation. Compared with constraining neighboring nodes via the shortest path distance, our method has the following benefits based on heat diffusion: (1) GraphHeat defines neighboring nodes in a continuous manner, via tuning the scaling parameter s; (2) It is flexible to leverage high-order neighbors while discarding some irrelevant low-order neighbors; (3) The range of neighboring nodes varies across target node, which is demonstrated in Section 4.6; (4) Parameter complexity does not increase with the order of neighboring nodes, since e^{-sL} can include the relation of all neighboring nodes in one matrix.

3.4 Architecture

Semi-supervised node classification assigns labels to unlabeled nodes according to the feature matrix of nodes and graph structure, supervised by the labels of a small set of nodes. In this paper, we consider a two-layer GraphHeat for semi-supervised node classification. The architecture of our model is

first layer :

$$X_j^2 = \text{ReLU}(\sum_{i=1}^{p}(\theta_{0,i,j}^1 I + \theta_{1,i,j}^1 e^{-sL})X_i^1), j = 1, \cdots, q,$$

second layer :

$$Z_j = \text{softmax}(\sum_{i=1}^{q}(\theta_{0,i,j}^2 I + \theta_{1,i,j}^2 e^{-sL})X_i^2), j = 1, \cdots, c,$$

$$(15)$$

where p is the number of input feature, q is the number of output feature in first layer, X_i^m is the i-th feature of m-th layer, $\theta_{0,i,j}^m$ and $\theta_{1,i,j}^m$ are parameters in m-th layer, c is the number of classes in node classification, Z_j is an n-dimensional vector representing the prediction result for all nodes in class j. The loss function is the cross-entropy error over all labeled nodes:

$$Loss = -\sum_{i \in y_L}\sum_{j=1}^{c}Y_{ij}\ln Z_j(i),\qquad (16)$$

where y_L is the set of labeled nodes, $Y_{ij} = 1$ if the label of node i is j, and $Y_{ij} = 0$ otherwise. The parameters θ are trained using gradient descent. Figure 3 shows the architecture of our model.

4 Experiments

We evaluate the effectiveness of GraphHeat on three benchmarks. A detailed analysis about the influence of hyperparameter is also conducted. Lastly, we show a case to intuitively demonstrate the strengths of our method.

4.1 Datasets

To evaluate the proposed method, we conduct experiments on three benchmark datasets, namely, Cora, Citeseer and Pubmed [Sen *et al.*, 2008]. In these citation network datasets, nodes represent documents and edges are citation links. Table 1 shows an overview of three datasets. Label rate denotes the proportion of labeled nodes for training.

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

Table 1: The statistics of datasets

4.2 Baselines

We compare with some traditional graph semi-supervised learning methods, including MLP, only leveraging node features for classification, manifold regularization (ManiReg) [Belkin *et al.*, 2006], semi-supervised embedding (SemiEmb) [Weston *et al.*, 2012], label propagation (LP) [Zhu *et al.*, 2003], graph embeddings (DeepWalk) [Perozzi *et al.*, 2014], iterative classification algorithm (ICA) [Lu and Getoor, 2003] and Planetoid [Yang *et al.*, 2016]. Furthermore, since graph convolutional networks are proved to
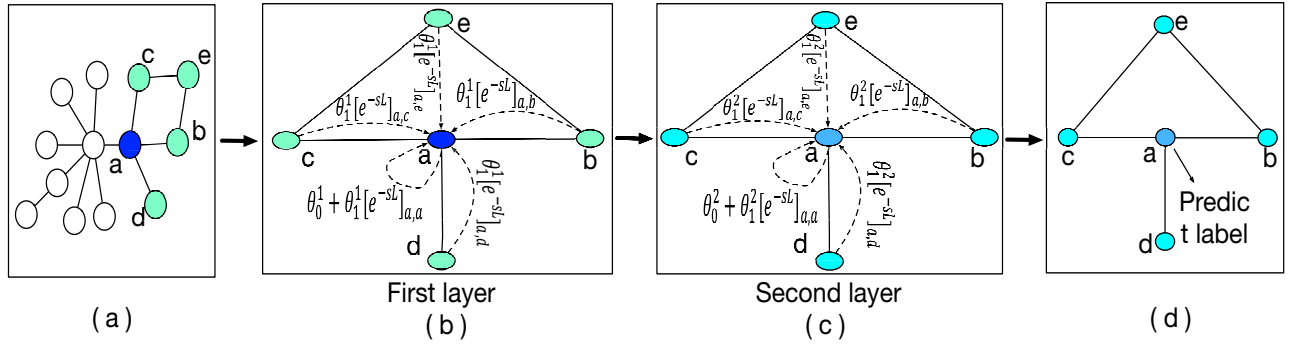
Figure 3: Two-layer GraphHeat for semi-supervised node classication. (a) select neighboring nodes (solid nodes) for target node a via e^{-sL}. (b-c) the solid line represents edge in graph. Dotted line represents the weights between selected nodes and target node a. Representation of a is updated through weighted average of solid nodes and itself. Node color reflects the update of node representation. (d) label prediction using updated representation in the last layer.

be effective in semi-supervised learning, we also compare against the state-of-the-art spectral graph convolutional networks, i.e., ChebyNet [Defferrard *et al.*, 2016], GCN [Kipf and Welling, 2017], and the state-of-the-art spatial convolutional networks, i.e., MoNet [Monti *et al.*, 2017] and GAT [Velickovic *et al.*, 2017].

4.3 Experimental Settings

We train a two-layer GraphHeat with 16 hidden units, and prediction accuracy is evaluated on a test set of 1000 labeled nodes. The partition of datasets is the same as GCN [Kipf and Welling, 2017] with an additional validation set of 500 labeled samples to determine hyper-parameters.

Weights are initialized following [Glorot and Bengio, 2010]. We adopt the Adam optimizer [Kingma and Ba, 2014] for parameter optimization with an initial learning rate $lr = 0.01$. If e^{-sL} is smaller than a given threshold ϵ, we set it as zero to accelerate the computation and avoid noise. The optimal hyper-parameters, e.g., scaling parameter s and threshold ϵ, are chosen through validation set. For Cora, $s = 3.5$ and $\epsilon = 1e-4$. For Citeseer, $s = 4.5$ and $\epsilon = 1e-5$. For Pubmed, $s = 3.0$ and $\epsilon = 1e-5$. To avoid overfitting, dropout [Srivastava *et al.*, 2014] is applied and the value is set as 0.5. The training process is terminated if the validation loss does not decrease for 200 consecutive epochs.

4.4 Performance on Node Classification Task

We now validate the effectiveness of GraphHeat on node classification. Similar to previous methods, we use classification accuracy metric for quantitative evaluation. Experimental results are reported in Table 2. Graph convolutional networks, including spatial methods and spectral methods, all perform much better than previous methods. This is due to that graph convolutional networks are trained in an end-to-end manner, and update representations via graph structure under the guide of labels.

As for our GraphHeat model, it outperforms all baseline methods, achieving state-of-the-art results on all the three datasets. Note that the key to graph-based semi-supervised learning is to capture the smoothness of labels or features over nodes exerted by graph structure. Different from all previous

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg	59.5%	60.1%	70.7%
SemiEmb	59.0%	59.6%	71.7%
LP	68.0%	45.3%	63.0%
DeepWalk	67.2%	43.2%	65.3%
ICA	75.1%	69.1%	73.9%
Planetoid	75.7%	64.7%	77.2%
ChebyNet	81.2%	69.8%	74.4%
GCN	81.5%	70.3%	79.0%
MoNet	81.7±0.5%	—	78.8±0.3%
GAT	83.0±0.7%	**72.5±0.7%**	79.0±0.3%
GraphHeat	**83.7%**	**72.5%**	**80.5%**

Table 2: Results of node classification

graph convolutional networks, our GraphHeat achieves such a smooth graph convolution via discounting high-frequency basic filters. In addition, we provide our neighboring nodes via heat diffusion and modulate parameter s to suit diverse networks. In Cora and Pubmed, especially the Pubmed which has the smallest label rate, GraphHeat achieves its superiority significantly. In Citeseer which is sparser than the other two datasets, our method matches GAT. It may be due to that GAT learns the weight based on representation of nodes in hidden layer, while our model uses Laplacian matrix which only depends on the sparser graph.

4.5 Influence of Hyper-parameter s and ϵ

GraphHeat uses e^{-sL} as the combined filter, where s is a modulating parameter. As s becomes larger, the range of feature diffusion becomes larger. Additionally, e^{-sL} is truncated via ϵ. This setting can speed up computation and remove noise.

Figure 4 demonstrates the influence of hyper-parameter s and ϵ on classification accuracy of Cora. As s becomes larger, the range of neighboring nodes becomes larger, capturing much more information. Figure 4 shows that the classification accuracy exactly increases as s becomes larger at first. However, with continuous increase of s, some irrelevant nodes are leveraged to update the target node's feature, which violate the smoothness of graph and lead to a drop on
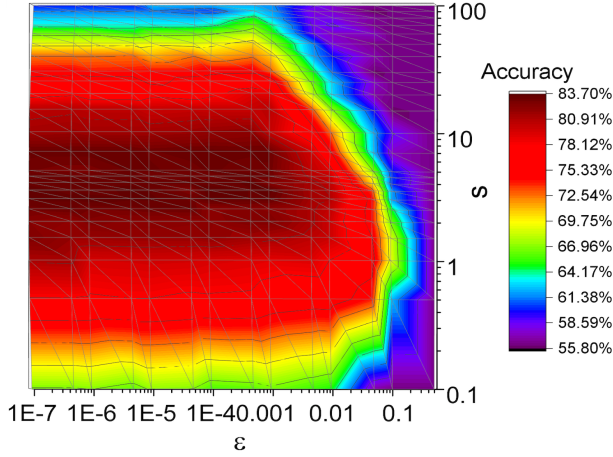
Figure 4: The influence of hyper-parameter s and ϵ on classification accuracy.

accuracy. The classification accuracy decreases a lot with the increasement of ϵ. Using a small ϵ as threshold can speed up computation and remove noise. However, as ϵ becomes large, the graph structure is overlooked and some relevant nodes are discarded, then the classification accuracy decreases.

4.6 Case Study

We conduct case study to illustrate the strengths of our method over competing method, i.e., GCN. Specifically, we select one node in Cora from the set of nodes that are correctly classified by our method and incorrectly classified by GCN. Figure 5 depicts the two-order ego network of the target node, marked as node a.
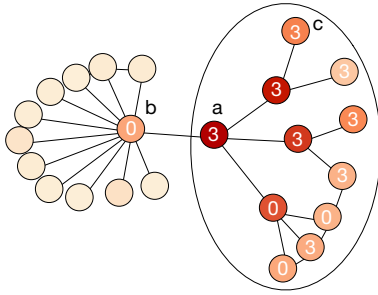

Figure 5: Ego network of node a, including first-order and second-order neighbors. Value of each node represents its label. Color of each node represents the strength of signal diffused from node a following heat diffusion. The deeper the color, the larger the value.

GCN leverages the information of the first-order neighbors of node a to assign label to node a. Checking the labels of these neighboring nodes, we can find that this is a challenging task for GCN since two neighboring nodes have label 0 and the other two have label 3.

Actually, the smoothness does not depend on order solely. GraphHeat leverages heat kernel to capture smoothness and determine neighboring nodes. When applying our Graph-Heat to this case, the neighboring nodes are obtained according to heat diffusion. As shown in Figure 5, some

second-order neighbors are used to update target node included in the black circle. Moreover, the weight of second-order neighbors can be greater than first-order neighbor, e.g., $[e^{-sL}]_{a,c} > [e^{-sL}]_{a,b}$. These second-order neighbors can enhance our method. At a glimpse of these nodes, we can easily learn why GraphHeat correctly assign label 3 to node a. In order to capture smoothness, our method enhances the low-frequency filters and discounts high-frequency filters. Instead of depending on order solely, our method exploits the local structure of target node via heat diffusion. A dense connected local structure can represent strong correlation among nodes. In contrast, although some nodes with high-degree are lower-order neighbors to target node, the connection to target node may represent its popularity rather than correlation, which suggests the weak correlation.

Index of taget node	12	75	26	1284	1351	1385	1666
Neighboring range	1	2	3	4	5	6	7
Maximum range	1	2	3	13	13	11	11

Table 3: Varying range of neighboring nodes in GraphHeat

Finally, we use the Cora dataset to illustrate that the range of neighboring nodes in GraphHeat varies across target nodes. Given a scaling parameter s, for some nodes, only a part of their immediate neighbors are included as neighboring nodes, while some nodes can reach a large range of neighboring nodes. To offer an intuitive understanding about the varying range of neighboring nodes, for a target node a, we compare the range of neighboring nodes and the potential maximum range, defined as

$$\max d_G(a,b) \qquad \forall\, b \in V, \qquad (17)$$

where $d_G(a,b)$ is the shortest path distance between node a and node b. Table 3 shows the results over some randomly-selected target nodes, confirming the flexibility of GraphHeat at defining various range of neighboring nodes. This flexibility is desired in an array of applications [Xu $et\ al.$, 2018].

5 Conclusion

The key to graph-based semi-supervised learning is capturing smoothness over graph, however, eigenvectors associated with higher eigenvalues are unsmooth. Our method leverages the heat kernel applied to eigenvalues. This method can discount high-frequency filters and assign larger importance to low-frequency filters, enforcing smoothness on graph. Moreover, we simplify our model to adapt to semi-supervised learning. Instead of restricting the range of neighboring nodes to depend on order solely, our method uses heat diffusion to determine neighboring nodes. The superiority of GraphHeat is shown in three benchmarks and it outperforms than previous methods consistently.

References

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

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

[Chung and Graham, 1997] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[Xu *et al.*, 2018] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536*, 2018.

[Xu *et al.*, 2019] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. In *International Conference on Learning Representations*, 2019.

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

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