SIMPLE SPECTRAL GRAPH CONVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Convolutional Networks (GCNs) have drawn significant attention and become promising methods for learning graph representations. The most GCNs suffer the performance loss when the depth of the model increases. Similarly to CNNs, without specially designed architectures, the performance of a network degrades quickly. Some researchers argue that the required neighbourhood size and neural network depth are two completely orthogonal aspects of graph representation. Thus, several methods extend the neighbourhood by aggregating k-hop neighbourhoods of nodes while using shallow neural networks. However, these methods still encounter oversmoothing, high computation and storage costs. In this paper, we use the Markov Diffusion Kernel to derive a variant of GCN called Simple Spectral Graph Convolution (S^2GC) which is closely related to spectral models and combines strengths of both spatial and spectral methods. Our spectral analysis shows that our simple spectral graph convolution used in S^2GC is a low-pass filter which partitions networks into a few large parts. Our experimental evaluation demonstrates that S^2GC with a linear learner is competitive in text and node classification tasks. Moreover, S^2GC is comparable to other state-of-the-art methods for node clustering and community prediction tasks.

1 INTRODUCTION

In the past few years, the rise and application of deep learning have successfully promoted the research of computer vision and data mining. Although these deep learning methods have been applied to extract the features of on the Euclidean grid (spatial data) with great success, the data in many practical scenarios lies on non-Euclidean spaces. Processing non-Euclidean spatial data is a challenge for deep learning methods. By defining a convolution operator between the graph and signal, Graph Convolutional Networks (GCNs) generalize Convolutional Neural Networks (CNNs) to graph-structured inputs which contain attributes. Message Passing Neural Networks (MPNNs) (Gilmer et al., 2017) unify the graph convolution as two functions: the transformation function and the aggregation function. MPNN iteratively propagates node features based on the adjacency structure of graph in a number of rounds.

Despite their enormous success in many applications like social media, traffic analysis, biology, recommendation systems and even computer vision, most of the current GCN models use shallow architectures because many of the recent models such as GCN (Kipf & Welling, 2016) achieve their best performance with 2-layer models. In other words, 2-layer GCN models aggregate nodes in two-hops neighbourhood and thus have no ability to extract information in k-hops neighbourhoods for $k > 2$. Moreover, stacking more layers and adding a non-linearity tends to degrade the performance of these models. Such a phenomenon is called oversmoothing (Li et al., 2018), which suggests that as the number of layers increases, the representations of the nodes in GCNs are inclined to converge to a certain value and become less distinct from one another. Even adding residual connections, an effective trick for training very deep neural networks in computer vision, merely slows down the oversmoothing issue (Kipf & Welling, 2016) in GCNs. It appears that deep GCN models gain nothing but the performance degradation from the deep architecture.

One solution for that is only deepening the aggregation function but keeping the same transformation function because the required neighbourhood size and neural network depth can be regarded as two orthogonal aspects of design. SGC (Wu et al., 2019) attempts to capture information from k-hops neighbours in the graph by applying the K-th power of the graph convolution matrix in a single neural network layer. This scheme is also used for attributed graph clustering (Zhang et al.,

2019). However, SGC is also suffering from oversmoothing as shown in Theorem 1. PPNP and APPNP (Klicpera et al., 2019a) replace the power of the graph convolution matrix with the Personalized PageRank matrix to solve the oversmoothing problem. Although APPNP tackles the oversmoothing problem, it exchanges the order of the aggregation function and the transformation function. It thus needs to learn a transformation function for the input before starting aggregating which limits the applications in unsupervised learning. GDC (Klicpera et al., 2019b) further extends APPNP by generalizing Personalized PageRank (Page et al., 1999) to an arbitrary graph diffusion process. GDC, technically, has stronger expressive power than SGC (Wu et al., 2019), PPNP and APPNP (Klicpera et al., 2019a) but it leads to a dense transition matrix which makes the computation and space storage intractable for large graphs (although authors suggest that the shrinkage method can be used to sparsify the generated transition matrix by ignoring small values).

To solve aforementioned issues, we propose a Simple Spectral Graph Convolution (S^2GC) network for node representation for different tasks such as: node clustering and node classification under semi-supervised and unsupervised setting. By analyzing Markov Diffusion Kernel (Fouss et al., 2012), we obtain a simple way to compute graph convolution: averaging the result of spatial graph convolutions with different power versions of the normalized adjacency matrix, which can be seen as aggregating neighbourhood information in the fashion of k-steps random walk. Thus, S^2GC is closely related to spectral models while it is still can be viewed as operating in the spatial domain. We also show that in spectral analysis the proposed method is a low-pass filter which leads graph partition into a few large parts. Moreover, we show a close connection between S^2GC and APPNP (Klicpera et al., 2019a), and provide a generalized form of S^2GC. Our experimental results include node clustering, unsupervised node classification, semi-supervised node classification and supervised text classification. We show that S^2GC is highly competitive and often can significantly outperform state-of-the-art methods.

2 PRELIMINARIES

Notations. Let $G = (V, E)$ be a simple and connected undirected graph with n nodes and m edges. We use $\{1, \cdots, n\}$ to denote the node index of G, and d_j denote the degree of node j in G. Let $\mathbf{A}$ be the adjacency matrix and $\mathbf{D}$ the diagonal degree matrix. Let $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ denote the adjacency matrix with added self-loops and the corresponding diagonal degree matrix $\widetilde{\mathbf{D}}$ where $\mathbf{I}_n \in \mathcal{S}^n_{++}$ is an identity matrix. Finally, let $\mathbf{X} \in \mathbb{R}^{n \times d}$ denote the node feature matrix and each node v is associated with a d-dimensional feature vector $\mathbf{X}_v$. The normalized graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \in \mathcal{S}^n_+$, a symmetric positive semidefinite matrix with eigendecomposition $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. Here $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues of $\mathbf{L}$, and $\mathbf{U} \in \mathbb{R}^{n \times n}$ is a unitary matrix that consists of the eigenvectors of $\mathbf{L}$.

Spectral Graph Convolution (Defferrard et al., 2016). We consider spectral convolutions on graphs defined as the multiplication of a signal $\mathbf{x} \in \mathbb{R}^n$ (a scalar for every node) with a filter g_θ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$ in the Fourier domain, *ie.*:

$$g_\theta(\mathbf{L}) * x = \mathbf{U} g^*_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \mathbf{x}, \tag{1}$$

where the parameter $\boldsymbol{\theta} \in \mathbb{R}^n$ corresponds to a vector of spectral filter coefficients. We can understand g_θ as a function operating on eigenvalues of $\mathbf{L}$, that is $g^*_\theta(\mathbf{\Lambda})$. To avoid eigendecomposition, $g_\theta(\mathbf{\Lambda})$ can be approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(\mathbf{\Lambda})$ up to the K-th order (Defferrard et al., 2016):

$$g^*_\theta(\mathbf{\Lambda}) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{\Lambda}}), \tag{2}$$

with a rescaled $\tilde{\mathbf{\Lambda}} = \frac{1}{2\lambda_{\max}}\mathbf{\Lambda} - \mathbf{I}_n$ where $\lambda_{\max}$ denotes the largest eigenvalue of $\mathbf{L}$ and $\boldsymbol{\theta} \in \mathbb{R}^K$ is now a vector of Chebyshev coefficients.

Vanila Graph Convolutional Network (GCN) (Kipf & Welling, 2016). The vanilla GCN can be regarded as a first-order approximation of spectral graph convolutions. If one sets $K = 1$, $\theta_0 = 2$, and $\theta_1 = -1$ for Eq. 2 they obtain the convolution operation $g_\theta(\mathbf{L}) * \mathbf{x} = (\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}$.

Finally, by the renormalization trick, replacing the matrix $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ by a normalized version $\widetilde{\mathbf{T}} = \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2} = (\mathbf{D}+\mathbf{I}_n)^{-1/2}(\mathbf{A}+\mathbf{I}_n)(\mathbf{D}+\mathbf{I}_n)^{-1/2}$ leads to the Graph Convolutional Layer:

$$H^{(l+1)} = \sigma(\widetilde{\mathbf{T}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \tag{3}$$

where σ is a non-linear function, *e.g*. ReLU.

Graph Diffusion Convolution (GDC) (Klicpera et al., 2019b). A generalized graph diffusion is given by the diffusion matrix

$$\mathbf{S} = \sum_{k=0}^{\infty} \theta_k \mathbf{T}^k, \tag{4}$$

with the weighting coefficients θ_k and the generalized transition matrix $\mathbf{T}$. Eq. 4 can be regarded as related to the Taylor expansion of matrix-valued functions. Thus, the choice of θ_k and $\mathbf{T}^k$ must at least ensure that Eq. 4 converges. Klicpera et al. (2019b) provide two special cases as low-pass filters, *ie*. heat kernel and the kernel based on random walk with restarts. If $\mathbf{S}$ denote the adjacency matrix and $\mathbf{D}$ be the diagonal degree matrix of $\mathbf{S}$ then the corresponding graph diffusion convolution is defined as $\mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}\mathbf{x}$.

Simple Graph Convolution (SGC) (Wu et al., 2019). For a classical MPNN (Gilmer et al., 2017), in each layer the hidden representations are averaged among neighbors that are 1-hop away. This implies that each node in the k-th layer obtains feature information from all nodes that are k-hops away in the graph. By hypothesizing that the non-linearity between GCN layers is not critical, SGC attempts to capture information from k-hops neighbourhood in the graph by applying the K-th power of the transition matrix in a single neural network layer. The SGC can be regarded as a special case of GDC without normalization by $\mathbf{D}^{-1/2}$ if we set $\theta_k = 1$ and $\theta_{i<K} = 0$ for Eq. 5, and $\mathbf{T} = \tilde{\mathbf{T}}$:

$$\hat{Y} = \text{softmax}(\widetilde{\mathbf{T}}^K\mathbf{X}\mathbf{W}). \tag{5}$$

Although SGC is an efficient and effective method, increasing Kleads to oversmoothing. Thus, SGC uses similar K number of layers as GCN. We provide Theorem 1 to demonstrate that oversmoothing is a result of convergence to the stationary distribution in graph diffusion as time $t \to \infty$.

Theorem 1. *(Chung & Graham, 1997) Let λ_2 denote second largest eigenvalue of transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, $\mathbf{p}(t)$ be the probability distribution vector and $\boldsymbol{\pi}$ the stationary distribution. If walk starts from the vertex i , $p_i(0) = 1$, then after t steps for every vertex:*

$$|p_j(t) - \pi_j| \leq \sqrt{\frac{d_j}{d_i}}\lambda_2^t, \tag{6}$$

Note that Eq. 6 converges to the stationary distribution only if the graph is non-bipartite.

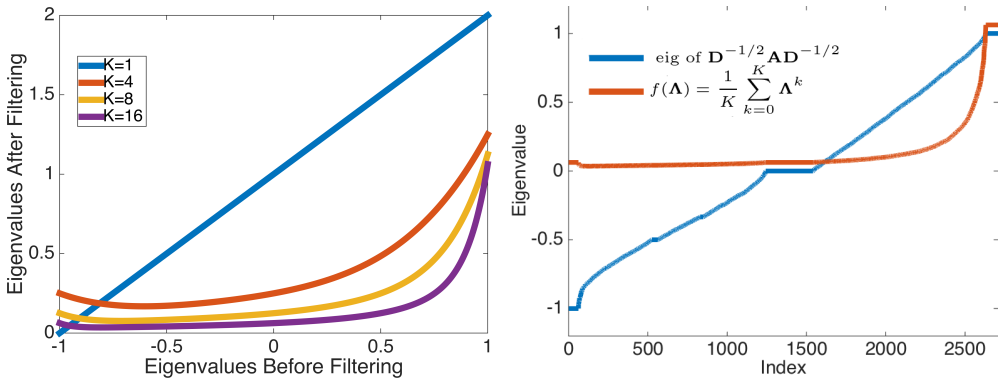
APPNA. Klicpera et al. (2019a) proposed APPNP which uses Personalized PageRank to derive a fixed filter of order K. Let $f_\theta(\mathbf{X})$ denote the output of a two-layer fully connected neural network on the feature matrix $\mathbf{X}$, the PPNP model is defined as:

$$\mathbf{H} = \alpha\mathbf{I}_n - (1-\alpha)\widetilde{\mathbf{T}}^{-1}f_\theta(\mathbf{X}). \tag{7}$$

To avoid calculating the inverse of matrix $\widetilde{\mathbf{T}}$, Klicpera et al. (2019a) also proposes APPNP which replaces Eq. 7 with an approximation derived by the truncated power iteration, defined as:

$$\mathbf{H}^{(l+1)} = (1-\alpha)\widetilde{\mathbf{T}}\mathbf{H}^{(l)} + \alpha\mathbf{H}^{(0)}, \tag{8}$$

where $\mathbf{H}^{(0)} = f_\theta(\mathbf{X}) = \text{ReLU}(\mathbf{X}\mathbf{W})$. By decoupling the feature transformation and propagation steps, PPNP and APPNP aggregate information from multi-hop neighbors without increasing the number of layers in the neural network.



(a) (b)

Figure 1: (a) Function $f(\lambda) = \frac{1}{K}\sum_{k=0}^{K}\lambda^k$ with $\lambda \in [-1, 1]$, $K \in \{1, 4, 8, 16\}$; (b) Sorted by index, eigenvalues of $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and push-forward eigenvalues $f(\mathbf{\Lambda}) = \frac{1}{K}\sum_{k=0}^{K}\mathbf{\Lambda}^k$ on Cora network ($K = 16$).

3 METHODOLOGY

In this section, firstly we briefly discuss the spectral analysis and graph partitioning, and conclude the small eigenvalues of a Laplacian matrix control global clustering which partitions the graph into a few of large clusters. Secondly, we analyze the Markov Diffusion Kernel (Fouss et al., 2012) and note that the corresponding feature mapping function acts as a low-pass filter. Finally, based on the feature mapping function we present our Simple Spectral Graph Convolution network.

3.1 SPECTRAL ANALYSIS AND GRAPH PARTITIONING

Firstly, we introduce the definitions of expansion and k-way Cheeger constant.

Definition 3.1. Expansion: For a node subset $S \subseteq V, \phi(S) = \frac{|\mathrm{E}(S)|}{\min\{\mathrm{vol}(S), \mathrm{vol}(V \setminus S)\}}$,

where $\mathrm{E}(\mathbf{S})$ is the set of edges with one node in $\mathbf{S}$ and $\mathrm{vol}(\mathbf{S})$ is the sum of degree of nodes in set S.

Definition 3.2. The k-way Cheeger constant is given as: $\rho_G(k) = \min_{S_1, S_2, \cdots, S_k} \max\{\phi(S_i) : i = \{1, \cdots, k\}\}$ where the minimum is over all collections of k non-empty disjoint subsets $S_1, S_2, \cdots, S_k \subseteq V$.

According to the definitions, the expansion in Def. 3.1 describes the effect of graph partitioning according to subset S while the k-way Cheeger constant reflects the effect of the graph partitioning into k parts–the smaller the value the better the partitioning is. Higher-order Cheeger's inequality (Bandeira et al., 2013; Lee et al., 2014) bridges the gap between the network spectral analysis and graph partitioning by controlling the bounds of k-way Cheeger constant:

$$\frac{\lambda_k}{2} \leq \rho_G(k) \leq \mathcal{O}\left(k^2\right)\sqrt{\lambda_k}, \tag{9}$$

where λ_k is the k-th eigenvalue of the normalized Laplacian matrix and $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. From inequality 9, we can conclude that small (large) eigenvalues control global clustering (local smoothing) effect of the graph partitioned into a few large parts (many small parts).

3.2 MARKOV DIFFUSION KERNEL

Before we define the diffusion distance, we briefly introduce the intuition behind it: two nodes are considered similar when they diffuse in a similar way through the graph, and therefore when they influence the other nodes in a similar manner (Fouss et al., 2012). In other words, two nodes are close if they are in the same cluster which has a consistent local structure. More precisely, the diffusion distance at time K between nodes i and j is defined as follows:

$$d_{ij}(K) = \|\mathbf{x}_i(K) - \mathbf{x}_j(K)\|_2^2, \tag{10}$$

where the average visiting rate $\mathbf{x}_i(K)$ after K steps for a process that started at time $k = 0$ is computed as follows:

$$\mathbf{x}_i(K) = \frac{1}{K}\sum_{k=0}^{K}\mathbf{T}^k\mathbf{x}_i(0). \tag{11}$$

By defining $\mathbf{Z}(K) = \frac{1}{K}\sum_{k=0}^{K}\mathbf{T}^k$, we reformulate Eq. 10 as:

$$d_{ij}(K) = \|\mathbf{Z}(K)(\mathbf{x}_i(0) - \mathbf{x}_j(0))\|_2^2. \tag{12}$$

The underlying feature map of Markov Diffusion Kernel (MDK) is given as $\mathbf{Z}(K)\mathbf{x}_i(0)$ for node i.

The effect of the linear projection $\mathbf{Z}(K)$ (filter) acting on spectrum as $f(\lambda) = \frac{1}{K}\sum_{k=0}^{K}\lambda^k$ is plotted in Figure 1, from which we observe the following two properties: (i) $\mathbf{Z}(K)$ preserves leading (large) eigenvalues of $\mathbf{T}$ and (ii) the higher K is the stricter the low-pass filter becomes. In other words, as K grows, this filter tries to approximate a low-rank positive semi-definite matrix by keeping the largest positive eigenvalues. Note the relationship between the normalized Laplacian matrix $\mathbf{L}$ and the corresponding normalized adjacent matrix $\mathbf{T}$ which is $\mathbf{L} = \mathbf{I} - \mathbf{T}$, and thus keeping large positive eigenvalues for $\mathbf{T}$ equals keeping small eigenvalues for $\mathbf{L}$.

3.3 Simple Spectral Graph Convolution

Based on the aforementioned Markov Diffusion Kernel, we propose the Simple Spectral Graph Convolution (S^2GC) network with a softmax after a linear layer:

$$\hat{Y} = \text{softmax}(\frac{1}{K}\sum_{k=0}^{K}\widetilde{\mathbf{T}}^k\mathbf{X}\mathbf{W}). \tag{13}$$

As $K \to \infty$, $\mathbf{H} = \sum_{k=0}^{\infty}\widetilde{\mathbf{T}}^k$ is the optimal diffused representation of the normalized Laplacian Regularization problem (Chapelle et al., 2006):

$$\min_{\mathbf{H}} \frac{1}{2}\sum_{i,j=1}^{n}\widetilde{\mathbf{A}}_{ij}\|\frac{\mathbf{h}_i}{\sqrt{d_i}} - \frac{\mathbf{h}_j}{\sqrt{d_j}}\|_2^2 + \|\mathbf{h}_i - \mathbf{x}_i\|_2^2, \tag{14}$$

where each vector $\mathbf{h}_i$ denotes the i-th row of $\mathbf{H}$.

The S^2GC can be regarded as a special case of GDC without normalization by $\mathbf{D}^{-1/2}$ they use if we set $\theta_i = 1/K$ and for Eq. 5 to obtain the convolution operation $g_\theta(\widetilde{\mathbf{T}}) * \mathbf{x} = \frac{1}{K}\sum_{k=0}^{K}\widetilde{\mathbf{T}}^k\mathbf{x}$ which can be decomposed into two terms: $\frac{1}{K}\sum_{k=1}^{K}\widetilde{\mathbf{T}}^k\mathbf{X}$ and $\frac{1}{K}\mathbf{X}$. When setting $K = 2$ and $\widetilde{\mathbf{P}} = \frac{\widetilde{\mathbf{T}}+\widetilde{\mathbf{T}}^2}{2}$ for Eq. 13, we obtain:

$$\mathbf{H}^{(1)} = 0.5\widetilde{\mathbf{P}}\mathbf{H}^{(0)} + 0.5\mathbf{H}^{(0)}, \tag{15}$$

where $\mathbf{H}^{(0)} = \mathbf{X}\mathbf{W}$. It is easy to observe that it is a special case of APPNA (Klicpera et al., 2019a). Specifically, it is an expansion containing the first step of APPNA without ReLU for $f_\theta(\mathbf{X})$, and the propagation matrix is a 2-hops matrix $\widetilde{\mathbf{P}} = \frac{\widetilde{\mathbf{T}}+\widetilde{\mathbf{T}}^2}{2}$.

We have shown Eq. 13 implicitly consider that attributes can be propagated back to themselves due to $\widetilde{T}^0 = \mathbf{I}$ which prevents oversmoothing (Kipf & Welling, 2016). Furthermore, by introducing the α parameter to preserve the node-self information as in PPNP and APPNP, we can generalize the Eq. 13 and explicitly consider the following form:

$$\hat{Y} = \text{softmax}\left(\frac{1}{K}\sum_{k=1}^{K}\left((1-\alpha)\,\widetilde{\mathbf{T}}^k\mathbf{X} + \alpha\mathbf{X}\right)\mathbf{W}\right), \tag{16}$$

where $\alpha \in (0, 1]$.

Relation with APPNP. If we define $\mathbf{H}^0 = \mathbf{X}\mathbf{W}$ then Eq. 16 can be reformulated as:

$$\hat{Y} = \text{softmax}\left(\frac{1}{K}\sum_{k=1}^{K}\left((1-\alpha)\,\widetilde{\mathbf{T}}^k\mathbf{H}^0 + \alpha\mathbf{H}^0\right)\right) \tag{17}$$

Thus, the Eq. 16 can be regarded as an aggregation on the first step of APPNP with a propagation matrix of different order. In spectral analysis, the diffusion projection matrix of APPNP is given as $(1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{P}})^{-1}$.

Table 1: Test Micro F1 Score (%) averaged over 10 runs on Reddit. Performance of other models are cited from their original papers.

Setting	Model	Test F1
Supervised	SAGE-mean	95.0
	SAGE-LSTM	**95.4**
Unsupervised	SAGE-GCN	93.0
	FastGCN	93.7
	SAGE-mean	89.7
	SAGE-LSTM	90.7
No Learning	SAGE-GCN	90.8
	DGI	94.0
	SGC	94.9
	S^2GC	**95.3**

Table 2: Clustering performance with three different metrics on four datasets.

Methods	Input	Cora			Citeseer			Pubmed			Wiki		
		Acc%	NMI%	F1%	Acc%	NMI%	F1%	Acc%	NMI%	F1%	Acc%	NMI%	F1%
k-means	Feature	34.65	16.73	25.42	38.49	17.02	30.47	57.32	29.12	57.35	33.37	30.20	24.51
Spectral-f	Feature	36.26	15.09	25.64	46.23	21.19	33.70	59.91	32.55	58.61	41.28	43.99	25.20
Spectral-g	Graph	34.19	19.49	30.17	25.91	11.84	29.48	39.74	3.46	51.97	23.58	19.28	17.21
DeepWalk	Graph	46.74	31.75	38.06	36.15	9.66	26.70	61.86	16.71	47.06	38.46	32.38	25.74
DNGR	Graph	49.24	37.29	37.29	32.59	18.02	44.19	45.35	15.38	17.90	37.58	35.85	25.38
GAE	Both	53.25	40.69	41.97	41.26	18.34	29.13	64.08	22.97	49.26	17.33	11.93	15.35
VGAE	Both	55.95	38.45	41.50	44.38	22.71	31.88	65.48	25.09	50.95	28.67	30.28	20.49
ARGE	Both	64.00	44.90	61.90	57.30	35.00	54.60	59.12	23.17	58.41	41.40	39.50	38.27
ARVGE	Both	63.80	45.00	62.70	54.40	26.10	52.90	58.22	20.62	23.04	41.55	40.01	37.80
AGC	Both	68.92	53.68	65.61	67.00	41.13	62.48	69.78	31.59	68.72	47.65	45.28	40.36
S^2GC	Both	**69.60**	**54.71**	**65.83**	**68.84**	**42.77**	**64.38**	**70.98**	**33.21**	**70.28**	**52.67**	**49.62**	**44.31**

Relation to Jumping Knowledge Network. In (Xu et al., 2018b), Jumping Knowledge Networks (JKN) combine intermediate node representations from each layer by forwarding them to the output of the final layer. Eq. 13 can be regarded as a variant of JKN with the average aggregation function. However, the mean-pooling aggregation is not mentioned in (Xu et al., 2018b) which use the max-pooling, concatenation- and attention-based aggregation layers. For Eq. 17, the intermediate representation for the mean-pooling aggregation function is not the same as in the next layers. Specifically, for the k-th layer, the input is $\widetilde{\mathbf{T}}^{k-1}\mathbf{X}$ while the intermediate representation of the (k-1)-th layer is $(1-\alpha)\widetilde{\mathbf{T}}^{k-1}\mathbf{X} + \alpha\mathbf{X}$.

4 Experiments

In this section, we evaluate the proposed method on four different tasks: node clustering, community prediction, semi-supervised node classification and text classification.

4.1 Node Clustering

We compare S^2GC with three kinds of clustering methods: (i) Methods that only use node features: k-means and spectral clustering (spectral-f) that constructs a similarity matrix with the node features by a linear kernel. (ii) Structural clustering methods that only use graph structures: spectral clustering (spectral-g) that takes the node adjacency matrix as the similarity matrix, DeepWalk (Perozzi et al., 2014), and Deep Neural Networks for Graph Representations (DNGR) (Cao et al., 2016), and (iii) Attributed graph clustering methods that utilize both node features and graph structures: Graph Autoencoder (GAE) and Graph Variational Autoencoder (VGAE) (Kipf & Welling, 2016), and Adversarially Regularized Graph Autoencoder (ARGE), Variational Graph Autoencoder (ARVGE) (Pan et al., 2018) and AGC (Zhang et al., 2019). To evaluate the clustering performance, three performance measures are adopted: clustering Accuracy (Acc), Normalized Mutual Information (NMI) and macro F1-score (F1). We run each method 10 times on four datasets: Cora, CiteSeer, PubMed, and Wiki, and we report the average clustering results in Table 2, where top-1 results are highlighted in bold.

Table 3: The statistics of datasets used for node classification and clustering.

Dataset	# Nodes	# Edges	class	feature	Train/Dev/Test Nodes
Cora	2, 708	5, 429	7	1433	140/500/1, 000
Citeseer	3, 327	4, 732	6	3703	120/500/1, 000
Pubmed	19, 717	44, 338	3	500	60/500/1, 000
Reddit	232, 965	11, 606, 919	41	602	152K/24K/55K
wiki	2405	17981	17	4973	

Table 4: Test accuracy (%) averaged over 10 runs on citation networks.

methods	Cora	Citeseer	Pubmed
GCN	81.4± 0.4	70.9± 0.5	79.0± 0.4
GAT	83.3± 0.7	72.6± 0.6	78.5± 0.3
FastGCN	79.8± 0.3	68.8± 0.6	77.4± 0.3
GIN	77.6± 1.1	66.1± 0.9	77.0± 1.2
DGI	82.5± 0.7	71.6± 0.7	78.4± 0.7
SGC	81.0± 0.0	71.9± 0.1	78.9± 0.0
MixHop	81.8±0.6	71.4±0.8	80.0±1.1
APPNP	**83.3±0.5**	71.7±0.6	80.1±0.2
Ours	83.0± 0.0	**73.6± 0.1**	**80.4± 0.0**

4.2 Community Prediction

We supplement our social network analysis by using S^2GC to inductively predict the community structure on Reddit, a large scale dataset as shown in Table 3 which cannot be processed by the vanilla GCN Kipf & Welling (2016) and GDC (Klicpera et al., 2019b) due to the memory issues. On the Reddit dataset, we train S^2GC with L-BFGS using no regularization, and we set $K = 5$ and $\alpha = 0.05$. We evaluate S^2GC inductively according to protocol (Chen et al., 2018). We train S^2GC on a subgraph comprising only training nodes and test on the original graph. On all datasets, we tune the number of epochs based on both convergence behavior and the obtained validation accuracy.

For Reddit, we compare S^2GC to the reported performance of supervised and unsupervised variants of GraphSAGE (Hamilton et al., 2017), FastGCN (Chen et al., 2018), SGC (Wu et al., 2019) and DGI (Velickovic et al., 2019). Table 1 also highlights the setting of the feature extraction step for each method. Note that S^2GC and SGC involve no learning because they do not learn any parameters to extract features. The logistic regression is used as a classifier for both unsupervised and no-learning approaches to train with labels afterward.

4.3 Semi-Supervised Node Classification

For the semi-supervised node classification task, we apply the standard fixed training/validation/testing split (Yang et al., 2016) on the Cora, Citeseer, and Pubmed datasets, with 20 nodes per class for training, 500 nodes for validation and 1,000 nodes for testing. For baselines, We include three state-of-the-art shallow models: GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), FastGCN (Chen et al., 2018), APPNP (Klicpera et al., 2019a), Mixhop (Abu-El-Haija et al., 2019), SGC (Wu et al., 2019), DGI (Velickovic et al., 2019) and GIN (Xu et al., 2018a). We use the Adam SGD optimizer (Kingma & Ba, 2014) with a learning rate of 0.02 to train S^2GC. We set $\alpha = 0.05$ and $K = 16$ on all datasets. We perform a grid search to tune the weight decay for models with different depths based on the accuracy and loss of the validation set. More details of hyperparameters are listed in the supplementary material.

4.4 Text Classification

Text classification predicts the labels of documents. Yao et al. (2019) use a 2-layer GCN to achieve state-of-the-art results by creating a corpus-level graph which treats both documents and words as nodes in a graph. Word-to-word edge weights are given by Point-wise Mutual Information (PMI) and word-document edge weights are given by normalized TF-IDF scores.

Table 5: The statistics of datasets for text classification.

Dataset	# Docs	# Training	# Test	# Words	# Nodes	# Classes	Average Length
20NG	18,846	11,314	7,532	42,757	61,603	20	221.26
R8	7,674	5,485	2,189	7,688	15,362	8	65.72
R52	9,100	6,532	2,568	8,892	17,992	52	69.82
Ohsumed	7,400	3,357	4,043	14,157	21,557	23	135.82
MR	10,662	7,108	3,554	18,764	29,426	2	20.39

Table 6: Test accuracy on the document classification task.

Model	20NG	R8	R52	Ohsumed	MR
Text GCN	87.9 ± 0.2	97.0 ± 0.2	93.8 ± 0.2	68.2 ± 0.4	76.3 ± 0.3
SGC	88.5 ± 0.1	97.2 ± 0.2	94.0 ± 0.2	68.5 ± 0.3	75.9 ± 0.3
S^2GC	**88.6± 0.1**	**97.4 ± 0.1**	**94.5 ± 0.2**	**68.5 ± 0.1**	**76.7 ± 0.0**

Setting and baseline We ran our experiments on five widely used benchmark corpora including movie review (MR), 20-Newsgroups (20NG), Ohsumed, R52 and R8 of Reuters 21578. We first preprocessed all the datasets by cleaning and tokenizing text as (Kim, 2014). We then removed stop words defined in NLTK6 and low frequency words appearing less than 5 times for 20NG, R8, R52 and Ohsumed. We compare two state-of-the-art models with our method: GCN (Kipf & Welling, 2016) and SGC (Wu et al., 2019). The statistics of the preprocessed datasets are summarized in Table 5. Table 6 shows that an S^2GC rivals their models on 5 benchmark datasets.

Parameters. We follow the setting of Text GCN (Yao et al., 2019) that includes experiments on four widely used benchmark corpora such as 20-Newsgroups (20NG), Ohsumed, R52 and R8 of Reuters 21578. For Text GCN, SGC, and our approach, the embedding size of the first convolution layer is 200 and the window size is 20. We set the learning rate to 0.02, dropout rate to 0.5 and the decay rate to 0. The 10% of training set is randomly selected for validation. Following (Kipf & Welling, 2016), we trained our method and Text GCN for a maximum of 200 epochs using the Adam (Kingma & Ba, 2014) optimizer, and we stop training if the validation loss does not decrease for 10 consecutive epochs. The text graph was built according to steps detailed in the supplementary material.

5 Conclusions

We have proposed Simple Spectral Graph Convolution (S^2GC), a method based on the Markov Diffusion Kernel (Section 3.2) whose feature maps emerge from the normalized Laplacian Regularization problem (Section 3.3). The theoretical analysis shows that S^2GC is able to express a special case of the generalized graph diffusion (as shown in Eq. 4) with the same fixed coefficient ($\theta_i = 1/K$) for different order polynomial filters, and evaluations show our model is powerful. We have shown there exist a close connection between S^2GC and other methods such as SGC, APPNP and JKN by analyzing spectral properties and implementation of each model. S^2GC inherits many of the strengths of spectral methods and none of their weaknesses, *e.g.* oversmoothing. We have conducted extensive and rigorous experiments that have shown that S^2GC is competitive and frequently outperforms many state-of-the-art methods on unsupervised, semi-supervised and supervised tasks and several popular datasets.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pp. 21–29, 2019.

Afonso S Bandeira, Amit Singer, and Daniel A Spielman. A cheeger inequality for the graph connection laplacian. *SIAM Journal on Matrix Analysis and Applications*, 34(4):1611–1630, 2013.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, volume 16, pp. 1145–1152, 2016.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. Semi-supervised learning (adaptive computation and machine learning), 2006.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.

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

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pp. 3844–3852, 2016.

François Fouss, Kevin Francoisse, Luh Yen, Alain Pirotte, and Marco Saerens. An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. *Neural networks*, 31:53–72, 2012.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

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

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019a.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*, pp. 13354–13366, 2019b.

James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):1–30, 2014.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 3538–3545. Association for the Advancement of Artificial Intelligence, 2018.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

S Pan, R Hu, G Long, J Jiang, L Yao, and C Zhang. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*, 2005.

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*, pp. 701–710, 2014.

Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, 2015.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Lio, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR (Poster)*, 2019.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462, 2018b.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7370–7377, 2019.

Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. *arXiv preprint arXiv:1906.01210*, 2019.

A Supplementary Material

A.1 Node Clustering

For S^2GC and AGC, we set max iterations to 60. For other baselines, we follow the parameter settings in the original papers. In particular, for DeepWalk, the number of random walks is 10, the number of latent dimensions for each node is 128, and the path length of each random walk is 80. For DNGR, the autoencoder is of three layers with 512 neurons and 256 neurons in the hidden layers respectively. For GAE and VGAE, we construct encoders with a 32-neuron hidden layer and a 16-neuron embedding layer, and train the encoders for 200 iterations using the Adam optimizer with learning rate equal 0.01. For ARGE and ARVGE, we construct encoders with a 32-neuron hidden layer and a 16-neuron embedding layer. The discriminators are built by two hidden layers with 16 and 64 neurons respectively. On Cora, Citeseer and Wiki, we train the autoencoder-related models of ARGE and ARVGE for 200 iterations with the Adam optimizer, with the encoder and discriminator learning rates both set as 0.001; on Pubmed, we train them for 2000 iterations with the encoder learning rate 0.001 and the discriminator learning rate 0.008.

A.2 Text Classification

The 20NG dataset1 (bydate version) contains 18,846 documents evenly categorized into 20 different categories. In total, 11,314 documents are in the training set and 7,532 documents are in the test set.

The Ohsumed corpus2 is from the MEDLINE database, which is a bibliographic database of important medical literature maintained by the National Library of Medicine. In this work, we used the 13,929 unique cardiovascular diseases abstracts in the first 20,000 abstracts of the year 1991. Each document in the set has one or more associated categories from the 23 disease categories. As we focus on single-label text classification, the documents belonging to multiple categories are excluded so that 7,400 documents belonging to only one category remain. 3,357 documents are in the training set and 4,043 documents are in the test set.

R52 and R8 (all-terms version) are two subsets of the Reuters 21578 dataset. R8 has 8 categories, and was split to 5,485 training and 2,189 test documents. R52 has 52 categories, and was split to 6,532 training and 2,568 test documents.

MR is a movie review dataset for binary sentiment classification, in which each review only contains one sentence (Pang & Lee, 2005) The corpus has 5,331 positive and 5,331 negative reviews. We used the training/test split in (Tang et al., 2015).

A.2.1 TEXT GRAPH

To convert text classification into the node classification on graph, there are two relationships considered when forming graphs: (i) the relation between documents and words and (ii) the connection between words. For the first type of relations, we build edges among word nodes and document nodes based on the word occurrence in documents. The weight of the edge between a document node and a word node is the Term Frequency-Inverse Document Frequency (Rajaraman & Ullman, 2011) (TF-IDF) of the word in the document applied to build the Docs-words graph. For the second type of relations, we build edges in graph among word co-occurrences across the whole corpus. To utilize the global word co-occurrence information, we use a fixed-size sliding window on all documents in the corpus to gather co-occurrence statistics. Point-wise Mutual Information (Church & Hanks, 1990) (PMI), a popular measure for word associations, is used to calculate weights between two word nodes according to the following definition:

$$\text{PMI}(i,j) = \log \frac{p(i,j)}{p(i)p(j)} \tag{18}$$

where $p(i,j) = \frac{W(i,j)}{W}$, $p(i) = \frac{W(i)}{W}$. $\#W(i)$ is the number of sliding windows in a corpus that contain word i, $\#W(i,j)$ is the number of sliding windows that contain both word i and word j, and $\#W$ is the total number of sliding windows in the corpus. A positive PMI value implies a high semantic correlation of words in a corpus, while a negative PMI value indicates little or no semantic correlation in the corpus. Therefore, we only add edges between word pairs with positive PMI values:

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{W}_1 & \mathbf{W}_2 \\ \hline \mathbf{W}_2^\top & \mathbf{I} \end{array}\right]$$

or

$$A_{ij} = \begin{cases} \text{PMI}(i,j) & \text{if } i,j \text{ are words, PMI}(i,j) > 0, \\ \text{TF-IDF}_{ij} & \text{if } i \text{ is document}, j \text{ is word}, \\ 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$