Gaussian-Induced Convolution for Graphs

Jiatao Jiang, Zhen Cui,* Chunyan Xu, Jian Yang
PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education,
and Jiangsu Key Lab of Image and Video Understanding for Social Security,
School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
{jiatao, zhen.cui, cyx, csjyang}@njust.edu.cn

Abstract

Learning representation on graph plays a crucial role in numerous tasks of pattern recognition. Different from grid-shaped images/videos, on which local convolution kernels can be lattices, however, graphs are fully coordinate-free on vertices and edges. In this work, we propose a Gaussian-induced convolution (GIC) framework to conduct local convolution filtering on irregular graphs. Specifically, an edge-induced Gaussian mixture model is designed to encode variations of subgraph region by integrating edge information into weighted Gaussian models, each of which implicitly characterizes one component of subgraph variations. In order to coarsen a graph, we derive a vertex-induced Gaussian mixture model to cluster vertices dynamically according to the connection of edges, which is approximately equivalent to the weighted graph cut. We conduct our multi-layer graph convolution network on several public datasets of graph classification. The extensive experiments demonstrate that our GIC is effective and can achieve the state-of-the-art results.

Introduction

As witnessed by the widespread applications, graph is one of the most successful models to conduct structured and semi-structured data, ranging from text (Defferrard, Bresson, and Vandergheynst 2016), bioinformatics (Yanardag and Vishwanathan 2015; Niepert, Ahmed, and Kutzkov 2016; Song et al. 2018) and social network (Gomez, Chiem, and Delvenne 2017; Orsini, Baracchi, and Frasconi 2017) to images/videos (Marino, Salakhutdinov, and Gupta 2016; Cui et al. 2018; Cui, Yang, and others 2017). Among these applications, learning robust representations from structured graphs becomes the main topic. To this end, various methods have come forth in recent years. Graph kernels (Yanardag and Vishwanathan 2015) and recurrent neural networks (RNNs) (Scarselli et al. 2009) are the most representative ones. Graph kernels usually take the classic R-convolution strategy (Haussler 1999) to recursively decompose graphs into atomic sub-structures and then define local similarities between them. RNNs based methods sequentially traverse neighbors with tied parameters in depth. With the increase of graph size, graph kernels would suffer diagonal dominance of kernels (Schölkopf et al. 2002) while RNNs would have the explosive number of combinatorial paths in the recursive stage.

Recently convolutional neural networks (CNNs) (LeCun, Bengio, and Hinton 2015) have achieved breakthrough progresses on representing grid-shaped image/video data. In contrast, graphs are with irregular structures and fully coordinate-free on vertices and edges. The vertices/edges are not strictly ordered, and can not be explicitly matched between two graphs. To generalize the idea of CNNs onto graphs, we need to solve this problem therein that the same responses should be produced for those homomorphic graphs/subgraphs when performing convolutional filtering. To this end, recent graph convolution methods (Defferrard, Bresson, and Vandergheynst 2016; Atwood and Towsley 2016; Hamilton, Ying, and Leskovec 2017) attempted to aggregate neighbor vertices as shown in Fig. 1e. This kind of methods actually employ a fuzzy filtering (i.e., a tied/shared filter) on neighbor vertices because only first-order statistics (mean) is used. Two examples are shown in Fig. 1a and Fig. 1b. Although they have different structures, the responses on them are fully equal. Oppositely, Niepert et.al (Niepert, Ahmed, and Kutzkov 2016) ranked neighbor vertices according to weights of edges, and then used different filters on these sorted vertices, as shown in Fig. 1f. However, this rigid ranking method will suffer some limitations: i) probably consistent responses to different structures (e.g., Fig. 1b and Fig. 1c) because weights of edges are out of consideration after ranking; ii) information loss of node pruning for a fixed-size receptive field as shown in Fig. 1b; and iii) ranking ambiguity for equal connections as shown in Fig. 1d; and iv) ranking sensitivity to (slightly) changes of edge weights/connections.

In this paper we propose a Gaussian-induced graph convolution framework to learn graph representation. For a coordinate-free subgraph region, we design an *edge-induced* Gaussian mixture model (EI-GMM) to implicitly coordinate the vertices therein. Specifically, the edges are used to regularize Gaussian models such that variations of subgraph can be well-encoded. In analogy to the standard convolutional kernel as shown in Fig. 1h, EI-GMM can be viewed as a coordinate normalization by projecting variations of subgraph into several Gaussian components. For example, the four subgraphs w.r.t. Fig. 1a$\sim$1d will have different repre-

*Corresponding author

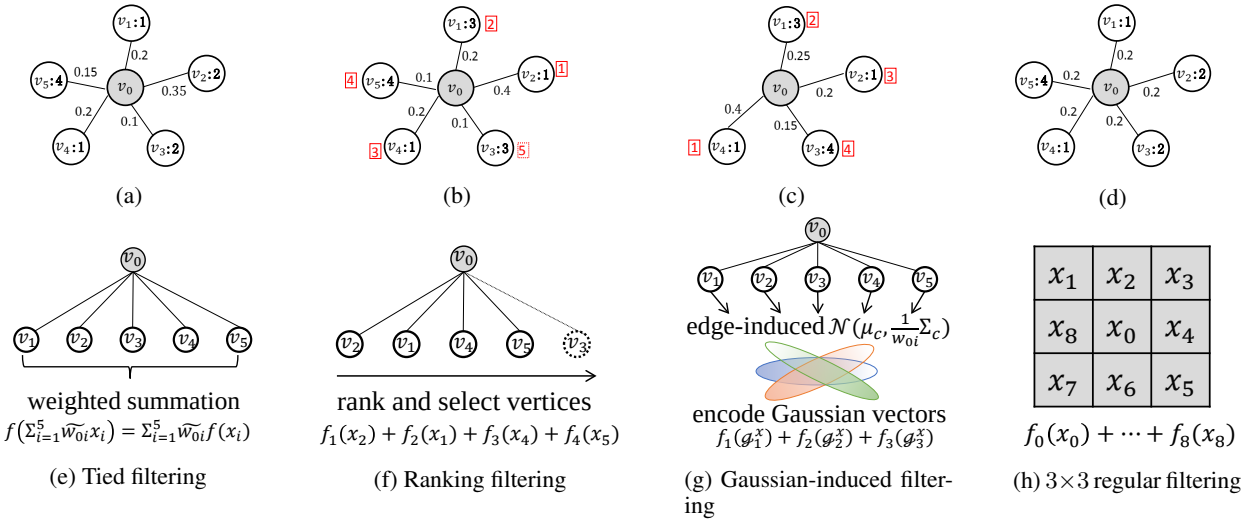

Figure 1: Different filters operations on graph vertices. Examples of one-hop subgraphs are given in (a)-(d), where v_0 is the reference vertex and each vertex is assigned to a signal. The tied filtering (e) summarizes all neighbor vertices, and generates the same responses to (a) and (b) under the filter f, i.e., $f(\sum \widetilde{w_{0i}}x_i) = f(1.9)$, although the two graphs are completely different in structures. The ranking filtering (f) sorts/prunes neighbor vertices and then performs different filtering on them. It might result into the same responses $f_1(1) + f_2(3) + f_3(1) + f_4(4)$ to different graphs such as (b) and (c), where the digits in red boxes denote the ranked indices and the vertex of dashed box in (b) is pruned. Moreover, the vertex ranking is uncertain/non-unique for equal connections in (d).To address these problems, we derive edge-induced GMM to coordinate subgraphs as shown in (g). Each of Gaussian model can be viewed as one variation component (or direction) of subgraph. Like the standard convolution (h), the Gaussian encoding is sensitive to different subgraphs, e.g., (a)-(d) will have different responses. Note f, f_i are linear filters, and the non-linear activation functions are put on their responses.

sentations[1] through our Gaussian encoding in Fig. 1g. To make the network inference forward, we transform Gaussian components of each subgraph into the gradient space of multivariate Gaussian parameters, instead of employing the sophisticated EM algorithm. Then the filters (or transform functions) are performed on different Gaussian components like latticed kernels on different directions in Fig. 1h. Further, we derive a *vertex-induced* Gaussian mixture model (VI-GMM) to favor dynamic coarsening of graph. We also theoretically analyze the approximate equivalency of VI-GMM to weighted graph cut (Dhillon, Guan, and Kulis 2007). Finally, EI-GMM and VI-GMM can be alternately stacked into an end-to-end optimization network.

In summary, our main contributions are four folds: i) propose an end-to-end Gaussian-induced convolutional neural network for graph representation; ii) propose edge-induced GMM to encode variations of different subgraphs; iii) derive vertex-induced GMM to perform dynamic coarsening of graphs, which is an approximation to the weighted graph cut; iv) verify the effectiveness of our method and report state-of-the-art results on several graph datasets.

[1]Suppose three Gaussian models are $\mathcal{N}(0,1), \mathcal{N}(0,2)$ and $\mathcal{N}(0,3)$, then we can compute the responses on (a)-(d) respectively as $f_1([0.49,-0.93]) + f_2([0.17,-0.65]) + f_3([0.07,-0.44])$, $f_1([0.35,-0.73]) + f_2([0.15,-0.58]) + f_3([0.10,-0.64]$, $f_1([0.35,-0.71]) + f_2([0.15,-0.39]) + f_3([0.10,-0.43])$, $f_1([0.46,-0.99]) + f_2([0.18,-0.62]) + f_3([0.08,-0.42])$. Please refer to incoming section.

Related Work

Graph CNNs mainly fall into two categories: spectral and spatial methods. Spectral methods (Bruna et al. 2014; Scarselli et al. 2009; Henaff, Bruna, and LeCun 2015; Such et al. 2017; Li et al. 2018a; 2018b) construct a series of spectral filters by decomposing graph Laplacian, which often suffers high-computational burden. To address this problem, the fast local spectral filtering method (Defferrard, Bresson, and Vandergheynst 2016) parameterizes the frequency responses as a Chebyshev polynomial approximation. However, as shown in Fig. 1b, after summarizing all nodes, this method will discard topology structures of a local receptive field. This kind of methods usually require equal sizes of graphs like the same sizes of images for CNNs (Kipf and Welling 2017). Spatial methods attempt to define spatial structures of adjacent vertices and then perform filtering on structured graphs. Diffusion CNNs (Atwood and Towsley 2016) scans a diffusion process across each node. PATCHY-SAN (Niepert, Ahmed, and Kutzkov 2016) linearizes neighbors by sorting weights of edges and deriving convolutional filtering on graphs, as shown in Fig. 1c. As an alternative, random walks based approach is also used to define the neighborhoods (Perozzi, Al-Rfou, and Skiena 2014). For the linearized neighbors, RNNs (Li et al. 2016) could be used to model the structured sequences. Similarly, NgramCNN (Luo et al. 2017) serializes each graph by introducing the concept of n-gram block. GAT (Velickovic et al. 2018) attempts to weight edges through the attention mechanism. WSC (Jiang

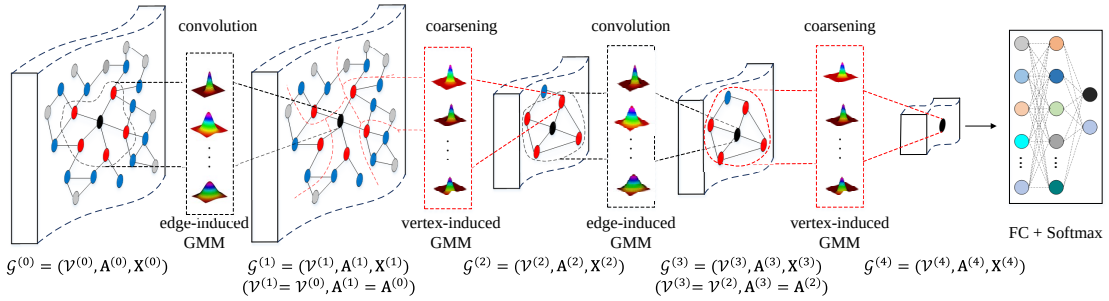

Figure 2: The GIC network architecture. The GIC main contains two module: convolution layer (EI-GMM) and coarsening layer (VI-GMM). The GIC stacks several convolution and coarsening layers alternatively and iteratively. More details can be found in incoming section.

et al. 2018) attempts to aggregate walk fields defined by random walks into Gaussian mixture models. Zhao (Zhao et al. 2018) attempts to define a standard network with different graph convolutions. Besides, some variants (Hamilton, Ying, and Leskovec 2017; Duran and Niepert 2017; Zhang et al. 2018) employ the aggregation or propagation of local neighbor nodes. Different from these tied filtering or ranking filtering methods, we use Gaussian models to encode local variations of graph. Also different from the recent mixture models (Monti et al. 2017), which uses GMM to only learn the importance of adjacent nodes, our method uses weighted GMM to encode the distributions of local graph structures.

The GIC Network

Attribute Graph

Here we consider an undirected attribute graph $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$ of m vertices (or nodes), where $\mathcal{V} = \{v_i\}_{i=1}^m$ is the set of vertices, $\mathbf{A}$ is a (weighted) adjacency matrix, and $\mathbf{X}$ is a matrix of graph attributes (or signals). The adjacency matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ records the connections between vertices. If v_i, v_j are not connected, then $A(v_i, v_j) = 0$, otherwise $A(v_i, v_j) \neq 0$. We sometimes abbreviate $A(v_i, v_j)$ as A_{ij}. The attribute matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ is associated with the vertex set $\mathcal{V}$, whose i-th row $\mathbf{X}_i$ (or $\mathbf{X}_{v_i}$) denotes a d-dimension attribute of the i-th node (i.e., v_i).

The graph Laplacian matrix $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} \in \mathbb{R}^{m \times m}$ is the diagonal degree matrix with $D_{ii} = \sum_j A_{ij}$. The normalized version is written as $\mathbf{L}^{norm} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. where $\mathbf{I}$ is the identity matrix. Unless otherwise specified, we use the latter. We give the definition of subgraph used in the following.

Definition 1. *Given an attribute graph $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$, the attribute graph $\mathcal{G}' = (\mathcal{V}', \mathbf{A}', \mathbf{X}')$ is a subgraph of $\mathcal{G}$, denoted $\mathcal{G}' \subseteq \mathcal{G}$, if (i) $\mathcal{V}' \subseteq \mathcal{V}$, (ii) $\mathbf{A}'$ is the submatrix of $\mathbf{A}$ w.r.t. the subset $\mathcal{V}'$, and (iii) $\mathbf{X}' = \mathbf{X}_{\mathcal{V}'}$.*

Overview

The GIC network architecture is shown in Fig. 2. Given an attribute graph $\mathcal{G}^{(0)} = (\mathcal{V}^{(0)}, \mathbf{A}^{(0)}, \mathbf{X}^{(0)})$, where the superscript denotes the layer number, we construct multi-scale receptive fields for each vertex based on the adjacency matrix $\mathbf{A}^{(0)}$. Each receptive field records k-hop neighborhood relationships around the reference vertex, and forms a local centralized subgraph. To encode the centralized subgraph, we project it into edge-induced Gaussian models, each of which defines one variation "direction" of the subgraph. We perform different filtering operations on different Gaussian components and aggregate all responses as the convolutional output. After the convolutional filtering, the input graph $\mathcal{G}^{(0)}$ is transformed into a new graph $\mathcal{G}^{(1)} = (\mathcal{V}^{(1)}, \mathbf{A}^{(1)}, \mathbf{X}^{(1)})$, where $\mathcal{V}^{(1)} = \mathcal{V}^{(0)}$ and $\mathbf{A}^{(1)} = \mathbf{A}^{(0)}$. To further abstract graphs, we next stack a coarsening layer on the graph $\mathcal{G}^{(1)}$. The proposed vertex-induced GMM is used to downsample the graph $\mathcal{G}^{(1)}$ into the low-resolution graph $\mathcal{G}^{(2)} = (\mathcal{V}^{(2)}, \mathbf{A}^{(2)}, \mathbf{X}^{(2)})$. Taking the convolution and coarsening modules, we may alternately stack them into a multi-layer GIC network, With the increase of layers, the receptive field size of filters will become larger, so the higher layer can extract more global graph information. In the supervised case of graph classification, we finally concatenate with a fully connected layer followed by a softmax loss layer.

Multi-Scale Receptive Fields

In the standard CNN, receptive fields may be conveniently defined as latticed spatial regions. Thus convolution kernels on grid-shaped structures are accessible. However, the construction of convolutional kernels on graphs are intractable due to coordinate-free graphs, e.g., unordered vertices, unfixed number of adjacent edges/vertices. To address this problem, we resort to the adjacent matrix $\mathbf{A}$, which expresses connections between vertices. Since $\mathbf{A}^k$ exactly records the k-step reachable vertices, we may construct a k-neighbor receptive field by using the k-order polynomial of $\mathbf{A}$, denoted as $\psi_k(\mathbf{A})$. Taking the simplest case, $\psi_k(\mathbf{A}) = \mathbf{A}^k$ reflects the k-hop neighborhood relationships. In order to remove the scale effect, we may normalize $\psi_k(\mathbf{A})$ as $\psi_k(\mathbf{A})\text{diag}(\psi_k(\mathbf{A})\mathbf{1})^{-1}$, which describes the reachable possibility in a k-hop walking. Formally, we define the k-th scale receptive field as a subgraph.

Definition 2. *The k-th scale receptive field around a reference vertex v_i is a subgraph $\mathcal{G}_{v_i}^k = (\mathcal{V}', \mathbf{A}', \mathbf{X}')$ of the k-order graph $(\mathcal{V}, \widetilde{\mathbf{A}} = \psi_k(\mathbf{A}), \mathbf{X})$, where $\mathcal{V}' = \{v_j | \widetilde{A}_{ij} \neq$*

$0\}\cup\{v_i\}$, $\mathbf{A}'$ *is the submatrix of* $\widetilde{\mathbf{A}}$ *w.r.t.* $\mathcal{V}'$, *and* $\mathbf{X}'=\mathbf{X}_{\mathcal{V}'}$.

Convolution: Edge-Induced GMM

Given a reference vertex v_i, we can construct the centralized subgraph $\mathcal{G}_{v_i}^k$ of the k-th scale. To coordinate the subgraph, we introduce Gaussian mixture models (GMMs), each of which may be understood as one principal direction of its variations. To encode the variations accurately, we jointly formulate attributes of vertices and connections of edges into Gaussian models. The edge weight $A'(v_i,v_j)$ indicates the relevance of v_j to the central vertex v_i. The higher weight is, the stronger impact on v_i is. So the weights can be incorporated into a Gaussian model by observing $A'(v_i,v_j)$ times. As the likelihood function, it is equivalent to raise the power $A'(v_i,v_j)$ on Gaussian function, which is proportional to $\mathcal{N}(\mathbf{X}'_{v_j},\mu,\frac{1}{A'(v_i,v_j)}\mathbf{\Sigma})$. Formally, we estimate the probability density of the subgraph $\mathcal{G}_{v_i}^k$ from the C_1-component GMM,

$$p_{v_i}(\mathbf{X}'_{v_j};\mathbf{\Theta}_1,A'_{ij})=\sum_{c=1}^{C_1}\pi_c\mathcal{N}(\mathbf{X}'_{v_j};\mu_c,\frac{1}{A'_{ij}}\mathbf{\Sigma}_c),$$
$$\text{s.t. }\pi_c>0,\sum_{c=1}^{C_1}\pi_c=1, \tag{1}$$

where $\mathbf{\Theta}_1=\{\pi_1,\cdots,\pi_{C_1},\mu_1,\cdots,\mu_{C_1},\mathbf{\Sigma}_1,\cdots,\mathbf{\Sigma}_{C_1}\}$ are the mixture parameters, $\{\pi_c\}$ are the mixture coefficients, $\{\mu_c,\mathbf{\Sigma}_c\}$ are the parameters of the k-th component, and $A'_{ij}>0$ [2]. Intuitively, edge weight A'_{ij} is, the stronger impact of the node v_j w.r.t. the reference vertex v_i is. We will refer to the model in Eqn. (1) as the *edge-induced Gaussian mixture model* (EI-GMM).

In what follows, we assume all attributes of nodes are independent on each other, which is often used in signal processing. That means, the covariance matrix $\mathbf{\Sigma}_c$ is diagonal, so we denote it as $\text{diag}(\sigma_c^2)$. To avoid the explicit constraints for π_c in Eqn. (1), we adopt the soft-max normalization with the re-parameterization variable α_c, i.e., $\pi_c=\exp(\alpha_c)/\sum_{k=1}^{C_1}\exp(\alpha_k)$. Thus, the entire subgraph log-likelihood can be written as

$$\zeta(\mathcal{G}_{v_i}^k)=\sum_{j=1}^{m}\ln p_{v_i}(\mathbf{X}'_{v_j};\mathbf{\Theta}_1,\mathbf{A}')$$
$$=\sum_{j=1}^{m}\ln\sum_{c=1}^{C_1}\pi_c\mathcal{N}(\mathbf{X}'_{v_j};\mu_c,\frac{1}{A'_{ij}}\mathbf{\Sigma}_c), \tag{2}$$

To infer forward, instead of the expectation-maximization (EM) algorithm, we use the gradients of the subgraph with regard to the parameters of the EI-GMM model $\mathbf{\Theta}_1$, motivated by the recent Fisher vector work (Sanchez et al. 2013), which has been proven to be effective in representation.

For a convenient calculation, we simplify the notations, $\mathcal{N}_{jc}=\mathcal{N}(\mathbf{X}'_{v_j},\mu_c,\frac{1}{A'_{ij}}\sigma_c^2)$ and $Q_{jc}=\frac{\pi_c\mathcal{N}_{jc}}{\sum_{k=1}^{C_1}\pi_k\mathcal{N}_{jk}}$, then we can derive the gradients of model parameters from Eqn. (2) as follows

$$\frac{\partial\zeta(\mathcal{G}_{v_i}^k)}{\partial\mu_c}=\sum_{j=1}^{m}\frac{A'_{ij}Q_{jc}(\mathbf{X}'_{v_j}-\mu_c)}{\sigma_c^2},$$
$$\frac{\partial\zeta(\mathcal{G}_{v_i}^k)}{\partial\sigma_c}=\sum_{j=1}^{m}\frac{Q_{jc}(A'_{ij}(\mathbf{X}'_{v_j}-\mu_c)^2-\sigma_c^2)}{\sigma_c^3}, \tag{3}$$

where the division of vectors means a term-by-term operation. Note we do not use $\partial\zeta(\mathcal{G}_{v_i}^k)/\partial\alpha_c$ due to no improvement in our experience. The gradients describe the contribution of the corresponding parameters to the generative process. The subgraph variations are adaptively allocated to C_1 Gaussian models. Finally, we ensemble all gradients w.r.t. Gaussian model (i.e., directions of graph) to analogize the collection of local square receptive field on image. Formally, for the k-scale receptive field $\mathcal{G}_{v_i}^k$ around the vertex v_i, the attributes produced from Gaussian models are filtered respectively and then concatenated,

$$F(\mathcal{G}_{v_i}^k,\mathbf{\Theta}_1,f)=\text{ReLU}(\sum_{c=1}^{C_1}f_i(\text{Cat}[\frac{\partial\zeta(\mathcal{G}_{v_i}^k)}{\partial\mu_c},\frac{\partial\zeta(\mathcal{G}_{v_i}^k)}{\partial\sigma_c}]), \tag{4}$$

where $\text{Cat}[\cdot,\cdot]$ is a concatenation operator, f_i is a linear filtering function (i.e., a convolution function) and ReLU is the rectified linear unit. Therefore we can produce the feature vectors that have same dimensionality depending on the number of Gaussian models for different subgraphs. If the soft assignment distribution Q_{jc} is sharply peaked on a single value of one certain Gaussian for the vertex v_j, the vertex will be only projected onto one Gaussian direction.

Coarsening: Vertex-Induced GMM

Like the standard pooling in CNNs, we need to downsample graphs so as to abstract them as well as reduce the computational cost. However, the pooling on images are tailored for latticed structures, and cannot be used for irregular graphs. One solution is to use some clustering algorithms to partition vertices to several clusters, and then produce a new vertex from each cluster. However, we expect that two vertices should not fall into the same cluster with a larger possibility if there is a high transfer difficulty between them. To this end, we derive vertex-induced Gaussian mixture models (VI-GMM) to weight each vertex. To utilize the edge information, we construct a latent observation $\phi(v_i)$ w.r.t. each vertex v_i from the graph Laplacian (or adjacent matrix if semi-positive definite), i.e., the kernel calculation $\langle\phi(v_i),\phi(v_j)\rangle=L_{ij}$. Moreover, for each vertex v_i, we define an influence factor w_i for Gaussian models. Formally, given C_2 Gaussian models, VI-GMM is written as

$$p(\phi(v_i);\mathbf{\Theta}_2,w_i)=\sum_{c=1}^{C_2}\pi_c\mathcal{N}(\phi(v_i);\mu_c,\frac{1}{w_i}\mathbf{\Sigma}_c),$$
$$\text{s.t. }w_i=h(\mathbf{X}_{v_i})>0, \tag{5}$$

where h is a mapping function to be learnt. To reduce the computation cost of matrix inverse on $\mathbf{\Sigma}$, we specify it as an

[2] In practice, we normalize $\mathbf{A}'$ into a non-negative matrix.

identity matrix. Then we have

$$p(\phi(v_i);\mathbf{\Theta}_2,w_i)=\sum_{c=1}^{C_2}\frac{\pi_c}{(\frac{2\pi}{w_i})^{d/2}}\exp^{-\frac{w_i}{2}\|\phi(v_i)-\mu_c\|^2}, \quad (6)$$

Given a graph with m vertices, the objective is to maximize the following log-likelihood:

$$\arg\max_{\mathbf{\Theta}_2}\zeta(\mathbf{\Theta}_2)=\sum_{i=1}^{m}\ln\sum_{c=1}^{C_2}\pi_c\mathcal{N}(\phi(v_i);\mu_c,\frac{1}{w_i}\mathbf{I})). \quad (7)$$

To solve above model in Eqn. (7), we use the iterative expectation maximization algorithm, which has closed-form solution at each step. Meanwhile, the algorithm may automatically conduct the required constraints. The graphical clustering process is summarized as follows:

(1) E-Step: the posteriors, i.e., the i-th vertex for the c-th cluster, are updated with $p_{ic}=\frac{\pi_c p(\phi(v_i);\theta_c,w_i)}{\sum_{k=1}^{C}\pi_k p(\phi(v_i);\theta_k,w_i)}$, where θ_c is the c-th Gaussian parameters, and $\mathbf{\Theta}_2=\{\theta_1,\cdots,\theta_{C_2}\}$.

(2) M-Step: we optimize Gaussian parameters π,μ. The parameter estimatation is given by $\pi_c=\frac{1}{m}\sum_{i=1}^{m}r_{ic}$, $\mu_c=\frac{\sum_{v_i\in\mathcal{G}_c}w_i\phi(v_i)}{\sum_{v_i\in\mathcal{G}_c}w_i}$. π_c indicates the energy summation of all vertices assigned to the cluster c, and μ_c may be understood as a doubly weighted (w_i,r_{ic}) average on the cluster c.

After several iterations of the two steps, we perform hard quantification. The i-th vertex is assigned as the class with the maximum possibility, formally, $r_{ic}=1$ if $c=\arg\max_k p_{ik}$, otherwise 0. Thus we can obtain the cluster matrix $\mathbf{P}\in\{0,1\}^{m\times C_2}$, where $P_{ic}=1$ if the i-th vertex falls into the cluster c. During coarsening, we take maximal responses of each cluster as the attributes of new vertex, and derive a new adjacency matrix by using $\mathbf{P}^\intercal\mathbf{A}\mathbf{P}$.

It is worth noting that we need not compute the concrete ϕ during the clustering process. The main calculation $\|\phi(v_i)-\mu_c\|^2$ in EM can be reduced to the kernel version: $K_{ii}-\frac{2\sum_{v_j\in\mathcal{G}_c}w_jK_{ij}}{\sum_{v_j\in\mathcal{G}_c}w_j}+\frac{\sum_{v_j,v_k\in\mathcal{G}_c}w_jw_kK_{jk}}{(\sum_{v_j\in\mathcal{G}_c}w_j)^2}$, where $K_{ij}=\langle\phi(v_i),\phi(v_j)\rangle$. In practice, we can use the graph Laplacian $\mathbf{L}$ as the kernel. In this case, we can easily reach the following proposition, which is relevant to graph cut (Dhillon, Guan, and Kulis 2007).

Proposition 1. *In EM, if the kernel matrix takes the weight-regularized graph Laplacian, i.e.,* $\mathcal{K}=diag(\mathbf{w})\mathbf{L}diag(\mathbf{w})$*, then VI-GMM is equal to an approximate optimization of graph cut, i.e.,* $\min\sum_{c=1}^{C}\frac{links(\mathcal{V}_c,\mathcal{V}\backslash\mathcal{V}_c)}{w(\mathcal{V}_c)}$*, where* $links(\mathcal{A},\mathcal{B})=\sum_{v_i\in\mathcal{A},v_j\in\mathcal{B}}A_{ij}$*, and* $w(\mathcal{V}_c)=\sum_{j\in\mathcal{V}_c}w_j$.

Experiments

Graph Classification

For graph classification, each graph is annotated with one label. We use two types of datasets: Bioinformatics and Network datasets. The former contains MUTAG (Debnath et al. 1991), PTC (Toivonen et al. 2003), NCI1 and NCI109 (Wale, Watson, and Karypis 2008), ENZYMES (Borgwardt et al. 2005) and PROTEINS (Borgwardt et al. 2005). The latter has COLLAB (Leskovec, Kleinberg, and Faloutsos 2005), REDDIT-BINARY, REDDIT-MULTI-5K, REDDIT-MULTI-12K, IMDB-BINARY and IMDB-MULTI.

Experiment Settings We verify our GIC on the above bioinformatics and social network datasets. In default, GIC mainly consists of three graph convolution layers, each of which is followed by a graph coarsening layer, and one fully connected layer with a final softmax layer as shown in Fig 2. Its configuration can simply be set as C(64)-P(0.25)-C(128)-P(0.25)-C(256)-P-FC(256), where C, P and FC denote the convolution, coarsening and fully connected layers respectively. The choices of hyperparameters are mainly inspired from the classic VGG net. For example, the coarsening factor is 0.25 (w.r.t. 0.5×0.5 in VGG), the attribute dimensions at three conv. layers are 64-128-256 (w.r.t. the channel numbers of conv1-3 in VGG). The scale of respective field and the number of Gaussian components are both set to 7. We train GIC network with stochastic gradient descent for roughly 300 epochs with a batch size of 100, where the learning rate is 0.1 and the momentum is 0.95.

In the bioinformatics datasets, we exploit labels and degrees of the vertices to generate initial attributes of each vertex. In the social network datasets, we use degrees of vertices. We closely follow the experimental setup in PSCN (Niepert, Ahmed, and Kutzkov 2016). We perform 10-fold cross-validation, 9-fold for training and 1-fold for testing. The experiments are repeated 10 times and the average accuracies are reported.

Comparisons with the State-of-the-arts We compare our GIC with several state-of-the-arts, which contain graph convolution networks (PSCN (Niepert, Ahmed, and Kutzkov 2016), DCNN (Atwood and Towsley 2016), NgramCNN (Luo et al. 2017)), neural networks (SAEN (Orsini, Baracchi, and Frasconi 2017)), feature based algorithms (DyF (Gomez, Chiem, and Delvenne 2017), FB (Bruna et al. 2014)), random walks based methods (RW (Gärtner, Flach, and Wrobel 2003)), graph kernel approaches (GK (Shervashidze et al. 2009), DGK (Yanardag and Vishwanathan 2015), WL (Morris, Kersting, and Mutzel 2017)). We present the comparisons with the state-of-the-arts, as shown in Table 1. All results come from the related literatures. We have the following observations.

Deep learning based methods on graphs (including DCNN, PSCN, NgramCNN, SAEN and ours) are superior to those conventional methods in most cases. The conventional kernel methods usually require the calculation on graph kernels with high-computational complexity. In contrast, these graph neural networks attempt to learn more abstract high-level features by performing inference-forward, which need relatively low computation cost.

Compared with recent graph convolution methods, ours can achieve better performance on most datasets, such as PTC, NCI1, NCI109, ENZYMES and PROTEINS. The main reason should be that local variations of subgraphs are accurately described with Gaussian component analysis.

The proposed GIC achieves state-of-the-art results on most datasets. The best performance is gained in some bioin-

Table 1: Comparisons with state-of-the-art methods.

DATASET	PSCN	DCNN	NGRAMCNN	FB	DYF	WL	GK	DGK	RW	SAEN	GIC
MUTAG	92.63 ±4.21	66.98 –	**94.99** ±**5.63**	84.66 ±2.01	88.00 ±2.37	78.3 ±1.9	81.66 ±2.11	82.66 ±1.45	83.72 ±1.50	84.99 ±1.82	94.44 ±4.30
PTC	60.00 ±4.82	56.60 –	68.57 ±1.72	55.58 2.30	57.15 ±1.47	– –	57.26 ±1.41	57.32 ±1.13	57.85 ±1.30	57.04 ± 1.30	**77.64** ± **6.98**
NCI1	78.59 ±1.89	62.61 –	– –	62.90 ±0.96	68.27 ±0.34	83.1 ±0.2	62.28 ±0.29	62.48 ±0.25	48.15 ±0.50	77.80 ± 0.42	**84.08** ±**1.77**
NCI109	– –	62.86 –	– –	62.43 ±1.13	66.72 ± 0.20	**85.2** ± **0.2**	62.60 ± 0.19	62.69 ± 0.23	49.75 ± 0.60	– –	82.86 ± 2.37
ENZYMES	– –	18.10 –	– –	29.00 ±1.16	33.21 ± 1.20	53.4 ± 1.4	26.61 ± 0.99	27.08 ± 0.79	24.16 ± 1.64	– –	**62.50** ± **5.12**
PROTEINS	75.89 ± 2.76	– –	75.96 ±2.98	69.97 ±1.34	75.04 ± 0.65	73.7 ± 0.5	71.67 ± 0.55	71.68 ± 0.50	74.22 ± 0.42	75.31 ± 0.70	**77.65** ± **3.21**
COLLAB	72.60 ± 2.15	– –	– –	76.35 1.64	80.61 ± 1.60	– –	72.84 ± 0.28	73.09 ± 0.25	69.01 ± 0.09	75.63 ± 0.31	**81.24** ± **1.44**
REDDIT-B	86.30 ± 1.58	– –	– –	88.98 ±2.26	**89.51** ± **1.96**	75.3 ± 0.3	77.34 ± 0.18	78.04 ± 0.39	67.63 ± 1.01	86.08 ± 0.53	88.45 ± 1.60
REDDIT-5K	49.10 ± 0.70	– –	– –	50.83 1.83	50.31 ± 1.92	– –	41.01 ± 0.17	41.27 ± 0.18	– –	**52.24** ± **0.38**	51.58 ± 1.68
REDDIT-12K	41.32 ± 0.42	– –	– –	42.37 1.27	40.30 ± 1.41	– –	31.82 ± 0.08	32.22 ± 0.10	– –	**46.72** ± **0.23**	42.98 ± 0.87
IMDB-B	71.00 ± 2.29	– –	71.66 ±2.71	72.02 ±4.71	72.87 ± 4.05	72.4 ± 0.5	65.87 ± 0.98	66.96 ± 0.56	64.54 ± 1.22	71.26 ± 0.74	**76.70** ± **3.25**
IMDB-M	45.23 ± 2.84	– –	50.66 ±4.10	47.34 3.56	48.12 ± 3.56	– –	43.89 ± 0.38	44.55 ± 0.52	34.54 ± 0.76	49.11 ± 0.64	**51.66** ± **3.40**

Table 2: Node label prediction on Reddit and PPI data (micro-averaged F1 score).

DATASET	REDDIT	PPI
RANDOM	0.042	0.396
RAW FEATURES	0.585	0.422
DEEP WALK	0.324	–
DEEP WALK + FEATURES	0.691	–
NODE2VEC + REGRESSION	0.934	–
GRAPHSAGE-GCN	0.930	0.500
GRAPHSAGE-MEAN	0.950	0.598
GRAPHSAGE-LSTM	**0.954**	0.612
GIC	0.952	**0.661**

formatics datasets and some social network datasets including PTC, NCI1, ENZYMES, PROTEINS, COLLAB, IMDB-BINARY and IMDB-MULTI. Although NgramCNN, DyF, WL and SEAN approaches have obtained the best performance on MUTAG, REDDIT-BINARY, NCI109, REDDIT-MULTI-5K and REDDIT-MULTI-12K respectively, our method is fully comparable to them.

Node Classification

For node classification, one node is assigned one/multiple labels. It is challenging if the label set is large. During training, we only use a fraction of nodes and their labels. The task is to predict the labels for the remaining nodes. Following the setting in (Hamilton, Ying, and Leskovec 2017), we conduct the experiments on Reddit data and PPI data. For a fair comparison to graphSAGE (Hamilton, Ying, and Leskovec 2017), we use the same initial graph data, mini-batch iterators, supervised loss function and neighborhood sample. The other network parameters are similar to graph classification except removing the coarsening layer.

Tabel 2 summarizes the comparison results. Our GIC can obtain the best performance 0.661 on PPI data and a comparable result 0.952 on Reddit data. The raw features provide an important initial information for node multi-label classification. Based on the raw features, deep walk (Perozzi, Al-Rfou, and Skiena 2014) improves about 0.36 (micro-F1 scores) on Reddit data. Meanwhile, we conduct an experiment of node2vec and use regression model to classification. Our method gains better performance than node2vec (Grover and Leskovec 2016). Comparing different aggregation methods like GCN (Kipf and Welling 2017), mean and LSTM, our GIC has a significant improvement about 0.16 on PPI data and gains a competitive performance on Reddit data. The results demonstrate our approach is robust to infer unknown labels of partial graphs.

Model Analysis

EI-GMM and VI-GMM: To directly analyze convolution filtering with EI-GMM, we compare our method with ChebNet (Defferrard, Bresson, and Vandergheynst 2016) and GCN (Kipf and Welling 2017) approaches by using the same coarsening mechanism VI-GMM. As shown in Table 3, under the same coarsening operation, our GIC is superior to

Table 3: The verification of our convolution and coarsening.

DATASET	CHEBNET w/ VI-GMM	GCN w/ VI-GMM	GIC w/O VI-GMM	GIC
MUTAG	89.44 ± 6.30	92.22 ± 5.66	93.33 ± 4.84	$\mathbf{94.44 \pm 4.30}$
PTC	68.23 ± 6.28	71.47 ± 4.75	68.23 ± 4.11	$\mathbf{77.64 \pm 6.98}$
NCI1	73.96 ± 1.87	76.39 ± 1.08	79.17 ± 1.63	$\mathbf{84.08 \pm 1.77}$
NCI109	72.88 ± 1.85	74.92 ± 1.70	77.81 ± 1.88	$\mathbf{82.86 \pm 2.37}$
ENZYMES	52.83 ± 7.34	51.50 ± 5.50	52.00 ± 4.76	$\mathbf{62.50 \pm 5.12}$
PROTEINS	78.10 ± 3.37	$\mathbf{80.09 \pm 3.20}$	78.19 ± 2.04	77.65 ± 3.21

Table 4: Comparisons on K and C_1.

DATASET	$K, C_1 = 1$	$K, C_1 = 3$	$K, C_1 = 5$	$K, C_1 = 7$
MUTAG	67.77 ± 11.05	83.88 ± 5.80	90.55 ± 6.11	$\mathbf{94.44 \pm 4.30}$
PTC	72.05 ± 8.02	77.05 ± 4.11	76.47 ± 5.58	$\mathbf{77.64 \pm 6.98}$
NCI1	71.21 ± 1.94	83.26 ± 1.17	$\mathbf{84.47 \pm 1.64}$	84.08 ± 1.77
NCI109	70.02 ± 1.57	81.74 ± 1.56	$\mathbf{83.39 \pm 1.65}$	82.86 ± 2.37
ENZYMES	33.83 ± 4.21	$\mathbf{64.00 \pm 4.42}$	63.66 ± 3.85	62.50 ± 5.12
PROTEINS	75.49 ± 4.00	77.47 ± 3.37	$\mathbf{78.10 \pm 2.96}$	77.65 ± 3.21

Table 5: Comparisons on the layer number.

DATASET	$N = 2$	$N = 4$	$N = 6$	$N = 8$
MUTAG	86.66 ± 8.31	91.11 ± 5.09	93.88 ± 5.80	$\mathbf{94.44 \pm 4.30}$
PTC	64.11 ± 6.55	74.41 ± 6.45	75.29 ± 6.05	$\mathbf{77.64 \pm 6.98}$
NCI1	71.82 ± 1.85	81.36 ± 1.07	83.01 ± 1.54	$\mathbf{84.08 \pm 1.77}$
NCI109	71.09 ± 2.41	80.02 ± 1.67	81.60 ± 1.83	$\mathbf{82.86 \pm 2.37}$
ENZYMES	42.33 ± 4.22	61.83 ± 5.55	$\mathbf{64.83 \pm 6.43}$	62.50 ± 5.12
PROTEINS	77.38 ± 2.97	$\mathbf{79.81 \pm 3.84}$	78.37 ± 4.00	77.65 ± 3.21

ChebNet+VI-GMM and GCN+VI-GMM. It indicates EI-GMM can indeed encode the variations of subgraphs more effectively. On the other hand, we remove the coarsening layer from our GIC. For different size graphs, we pad new zero vertices into a fixed size and then concatenate attributes of all vertices for classification. As shown in this table, the performance of GIC still outperforms GIC without VI-GMM coarsening, which verifies the effectiveness of the coarsening layer VI-GMM.

K **and** C_1: The kernel size K and the number of Gaussian components C_1 are the most crucial parameters. Generally, the C_1 is proportional to the K. The reason is that the larger receptive field usually contains more vertices (i.e., a relative large subgraph). Thus we simply take the equal values for them, $K = C_1 = \{1, 3, 5, 7\}$. The experimental results are shown in Table 4. With the increase of K, C_1, the performance improves at most cases. The reasons are two folds: i) with increasing receptive field size, the convolution will cover the farther hopping neighbors; ii) with the increase of C_1, the variations of subgraphs are encoded more accurately. But for the larger values of K and C_1 will increase the computational burden. Moreover, the overfitting phenomenon might occur with the increase of model complexity. Take the example of NCI109, in the first convolution layer, the encoded attributes (in Eqn. (4)) will be $2 \times 39 \times 7 = 546$ for each scale of receptive field, where 39 is the dimension of attributes (w.r.t the number of node labels) and 7 is the number of Gaussian components. Thus, for 7 scales of receptive field, the final encoded attributes will be $546 \times 7 = 3822$ dimensions, which will be mapping to 64 dimensions by the function $f = [f_1, \cdots, f_{C_1}]$ in Eqn. (4). Thus the model parameter is $3822 \times 64 = 244608$ in the first layer. Similarly, if the number of node label is 2, the model parameter will sharply decrease into 18816. Besides, the parameter complexity is related to the number of classes and nodes. The comparison results in Table 4 demonstrate the trend of the parameters K and C_1 in our GIC framework.

Number of stacked layers: Here we test on the number of stacked network layers with $N = 2, 4, 6, 8$. When $N = 2$, only one fully connected layer and one softmax layer are preserved. When $N = 4$, we add two layers: the convolution layer and the coarsening layer. When continuing to stack both, the depth of network will be 6 and 8. The results are shown in Table 5. Deeper networks can gain better performance in most cases, because the larger receptive field is observed and more abstract structures will be extracted in the topper layer. Of course, there is an extra risk of overfitting due to the increase of model complexity.

An analysis of computation complexity: In the convolution layer, the computational costs of receptive fields and Gaussian encoding are about $O(Km^2)$ and $O(C_1md^2)$ respectively, where m, d are number of nodes and the feature dimensionality. Generally, $K = C_1 \ll d < m$. In the coarsening layer, the time complexity is about $O(pm^2 + md)$, where p is iteration number of the EM algorithm. In all, suppose the whole GIC alternatively stacks n convolution and coarsening layers, the entire time complexity is $O(n(K + p)m^2 + nC_1md^2 + nmd)$.

Conclusion

In this paper, we proposed a novel Gaussian-induced convolution network to handle with general irregular graph data. Considering the previous spectral and spatial methods do not well characterize local variations of graph, we derived edge-induced GMM to adaptively encode subgraph structures by projecting them into several Gaussian components and then performing different filtering operations on each Gaussian direction like the standard CNN filters on images. Meanwhile, we formulated graph coarsening into vertex-induced GMM to dynamically partition a graph, which was also proven to be equal to graph cut. Extensive experiments in two graphic tasks (i.e. graph and node classification) demonstrated the effectiveness and superiority of our GIC compared with those baselines and state-of-the-art methods. In the future, we would like to extend our method into more applications to irregular data.

Acknowledgments

The authors would like to thank the Chairs and the anonymous reviewers for their critical and constructive comments and suggestions. This work was supported by the National Science Fund of China under Grant Nos. 61602244, 61772276, U1713208 and 61472187 and Program for Changjiang Scholars.

References

Atwood, J., and Towsley, D. 2016. Diffusion-convolutional neural networks. In *NIPS*, 1993–2001.

Borgwardt, K. M.; Ong, C. S.; Schönauer, S.; Vishwanathan, S.; Smola, A. J.; and Kriegel, H.-P. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21(suppl_1):i47–i56.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral networks and locally connected networks on graphs. *ICLR*.

Cui, Z.; Xu, C.; Zheng, W.; and Yang, J. 2018. Context-dependent diffusion network for visual relationship detection. In *MM*, 1475–1482. ACM.

Cui, Z.; Yang, J.; et al. 2017. Spectral filter tracking. *arXiv preprint arXiv:1707.05553*.

Debnath, A. K.; Lopez de Compadre, R. L.; Debnath, G.; Shusterman, A. J.; and Hansch, C. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry* 34(2):786–797.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3844–3852.

Dhillon, I. S.; Guan, Y.; and Kulis, B. 2007. Weighted graph cuts without eigenvectors a multilevel approach. *TPAMI* 29(11).

Duran, A. G., and Niepert, M. 2017. Learning graph representations with embedding propagation. In *NIPS*, 5119–5130.

Gärtner, T.; Flach, P.; and Wrobel, S. 2003. On graph kernels: Hardness results and efficient alternatives. *Learning Theory and Kernel Machines* 129–143.

Gomez, L. G.; Chiem, B.; and Delvenne, J.-C. 2017. Dynamics based features for graph classification. *arXiv preprint arXiv:1705.10817*.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*, 855–864. ACM.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*, 1025–1035.

Haussler, D. 1999. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California at Santa Cruz.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Jiang, J.; Cui, Z.; Xu, C.; Li, C.; and Yang, J. 2018. Walk-steered convolution for graph classification. *arXiv preprint arXiv:1804.05837*.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. *ICLR*.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436–444.

Leskovec, J.; Kleinberg, J.; and Faloutsos, C. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *SIGKDD*, 177–187. ACM.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2016. Gated graph sequence neural networks. *ICLR*.

Li, C.; Cui, Z.; Zheng, W.; Xu, C.; Ji, R.; and Yang, J. 2018a. Action-attending graphic neural network. *TIP* 27(7):3657–3670.

Li, C.; Cui, Z.; Zheng, W.; Xu, C.; and Yang, J. 2018b. Spatio-temporal graph convolution for skeleton based action recognition. *AAAI*.

Luo, Z.; Liu, L.; Yin, J.; Li, Y.; and Wu, Z. 2017. Deep learning of graphs with ngram convolutional neural networks. *TKDE* 29(10):2125–2139.

Marino, K.; Salakhutdinov, R.; and Gupta, A. 2016. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*.

Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, 3.

Morris, C.; Kersting, K.; and Mutzel, P. 2017. Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs. In *ICDM*, 327–336. IEEE.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *ICML*, 2014–2023.

Orsini, F.; Baracchi, D.; and Frasconi, P. 2017. Shift aggregate extract networks. *arXiv preprint arXiv:1703.05537*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*, 701–710. ACM.

Sanchez, J.; Perronnin, F.; Mensink, T.; and Verbeek, J. 2013. Image classification with the fisher vector: Theory and practice. *IJCV* 105(3):222–245.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The graph neural network model. *TNN* 20(1):61–80.

Schölkopf, B.; Weston, J.; Eskin, E.; Leslie, C.; and Noble, W. S. 2002. A kernel approach for learning from almost orthogonal patterns. In *ECML*, 511–528. Springer.

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, 488–495.

Song, T.; Zheng, W.; Song, P.; and Cui, Z. 2018. Eeg emotion recognition using dynamical graph convolutional neural networks. *IEEE Transactions on Affective Computing*.

Such, F. P.; Sah, S.; Dominguez, M. A.; Pillai, S.; Zhang, C.; Michael, A.; Cahill, N. D.; and Ptucha, R. 2017. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing* 11(6):884–896.

Toivonen, H.; Srinivasan, A.; King, R. D.; Kramer, S.; and Helma, C. 2003. Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics* 19(10):1183–1193.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. *ICLR*.

Wale, N.; Watson, I. A.; and Karypis, G. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems* 14(3):347–375.

Yanardag, P., and Vishwanathan, S. 2015. Deep graph kernels. In *SIGKDD*, 1365–1374. ACM.

Zhang, T.; Zheng, W.; Cui, Z.; and Li, Y. 2018. Tensor graph convolutional neural network. *arXiv preprint arXiv:1803.10071*.

Zhao, W.; Xu, C.; Cui, Z.; Zhang, T.; Jiang, J.; Zhang, Z.; and Yang, J. 2018. When work matters: Transforming classical network structures to graph cnn. *arXiv preprint arXiv:1807.02653*.