Variational Gridded Graph Convolution Network for Node Classification

Xiaobin Hong, Tong Zhang, Zhen Cui, and Jian Yang

***Abstract*—The existing graph convolution methods usually suffer high computational burdens, large memory requirements, and intractable batch-processing. In this paper, we propose a high-efficient variational gridded graph convolution network (VG-GCN) to encode non-regular graph data, which overcomes all these aforementioned problems. To capture graph topology structures efficiently, in the proposed framework, we propose a hierarchically-coarsened random walk (hcr-walk) by taking advantage of the classic random walk and node/edge encapsulation. The hcr-walk greatly mitigates the problem of exponentially explosive sampling times which occur in the classic version, while preserving graph structures well. To efficiently encode local hcr-walk around one reference node, we project hcr-walk into an ordered space to form image-like grid data, which favors those conventional convolution networks. Instead of the direct 2-D convolution filtering, a variational convolution block (VCB) is designed to model the distribution of the random-sampling hcr-walk inspired by the well-formulated variational inference. We experimentally validate the efficiency and effectiveness of our proposed VG-GCN, which has high computation speed, and the comparable or even better performance when compared with baseline GCNs.**

***Index Terms*—Graph coarsening, gridding, node classification, random walk, variational convolution.**

I. Introduction

IN recent years, convolutional neural networks (CNNs) [1] have achieved great success in a variety of machine learning tasks such as object detection [2], [3], machine translation [4], and speech recognition [5]. Basically, CNNs aim to explore the local correlation through neighborhood convolution, and are rather sophisticated to encode Euclidean structure data w.r.t. shape-gridded images and videos. In real-world applications, however, there is a large amount of non-Euclidean structure data such as social networks [6], citation networks [7], knowledge graphs [8], protein-protein interaction [9], and time series system [10]–[14], which are usual non-grid data and cannot be habitually encoded with the conventional convolution.

Manuscript received January 10, 2021; revised March 5, 2021 and April 4, 2021; accepted April 17, 2021. This work was supported by the Natural Science Foundation of Jiangsu Province (BK20190019, BK20190452), the National Natural Science Foundation of China (62072244, 61906094), and the Natural Science Foundation of Shandong Province (ZR2020LZH008). This work was partly collaborated with State Key Laboratory of High-end Server and Storage Technology. Recommended by Associate Editor Xin Luo. *(X. B. Hong and T. Zhang contributed equally to this work. Corresponding author: Zhen Cui.)*

Citation: X. B. Hong, T. Zhang, Z. Cui, and J. Yang, "Variational gridded graph convolution network for node classification," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 10, pp. 1697–1708, Oct. 2021.

The authors are with the Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: xbhong@njust.edu.cn; tong.zhang@njust.edu.cn; zhen.cui@njust.edu.cn; csjyang@njust.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2021.1004201

As graphs are natural and frequently-used to describe non-grid data, researchers have recently attempted to introduce convolution filtering into graph modeling, which is called graph convolution or graph convolution network (GCN). Generally, they fall into two categories: spectral based approaches [15]–[22] and spatial based approaches [23]–[26].

Spectral based approaches employ the recent emerging spectral graph theory to filter signals in the frequency domain of graph topology. Bruna *et al*. [15] proposed the first spectral convolution neural network (Spectral CNN), which defines the filter as a set of learnable parameters to process graph signals. Chebyshev Spectral CNN (ChebNet) [16] defines the Chebyshev polynomial of eigenvector diagonal as the convolutional filter, which avoids the computation of the graph Fourier basis, reducing the computation complexity from $O(N^3)$ to $O(Ke)$ (where N is the number of nodes, e is the number of edges, and K is the number of engine vectors). Kipf and Welling [17] proposed the most commonly used GCN, which is essentially a first-order approximation of ChebNet assuming $K = 1$ and the max-eigenvalue $\lambda_{\max} = 2$. Wu *et al*. [22] proposes a disordered graph convolutional neural network (DGCNN) based on the Gaussian Mixture Model, which extends CNN by adding a preprocessing layer called disordered graph convolutional layer (DGCL). DGCL uses a mixture of Gaussian functions to achieve the mapping between the convolution kernel and nearby nodes in the graph. Besides, some other GCN variants have also been proposed, including Hessian GCN (HesGCN) [19] and Hypergraph p-Laplacian GCN (HpLapGCN) [20]. Specifically, HesGCN and HpLapGCN belong to the first-order variant of GCN, while TGCN [21] is the second-order approximation of GCN.

These spectral based methods above are well-supported by the strong theory of graph signals, but they usually suffer high computational burdens because of the eigenvalue decomposition on graph topology. To mitigate this problem, the fast approximation algorithm [27]–[29] defines convolution filtering as a recursive calculation on graph topology, which actually may belong to the category of spatial convolution.

Spatial based approaches often use explicitly spatial edge-connection relations to aggregate those nodes locally adjacent to one reference node. The local aggregation with mean/sum/max operation on neighbor nodes cannot yet satisfy the Weisfeiler-Lehman (WL) test [30], where non-isomorphism graph structures need different filtering responses. The critical reason is that graph topology structures are degraded to certain-degree confusion after aggregation, even though the recent graph attention networks (GAT) [31] attempts to adaptively/discriminatively aggregate local neighbor nodes with different weights learnt by the attention mechanism between one central node and its neighbor nodes. Moreover, spatial based GCNs usually need to optimize the entire graph during training due to the intractability of batch-processing, which will result in high-memory requirements for large-scale graphs and thus cannot run on plain platforms. This means that, when graph structures change with newly-added/deleted nodes or links, the GCN models should be well-restarted or even re-trained for new structures. In addition, the time complexity of spatial based GCNs will be exponentially increased with the receptive field size (w.r.t. the hop step l), i.e., $O(Nm^l d^2)$ in each convolution layer, where m denotes average degree of nodes, and d is the dimension of input signal.

In this paper, we propose a high-efficient variational gridded graph convolution network (VG-GCN) with high-computation efficiency, low-memory cost, easy batch-process, and comparable or even better performance when compared with the baseline GCNs. To efficiently capture local structures, we introduce the random sampling strategy through random walks, which can well preserve graph topology under randomly sampling sufficient walks [32], [33]. As the quantity of walks has the exponentially-explosive increase with the walk step, i.e., $O(Nm^l)$, the burden of sampling sufficient walks tends to overwhelm the entire algorithm especially for the larger node degree $m \gg 2$. Instead of the original random walk, specifically, we propose a hierarchically-coarsened random walk (hcr-walk) to reduce sampling times. The strategy of hcr-walk can efficiently reduce traversal edges through random combinations (to form hyper-edge) of connection edges during walking. As a result, the hcr-walk balances the advantages of random walk as well as node aggregation. Under the fixed hyper-edge number $\widetilde{m}$, the hcr-walk will fall into a deep-first traversal on $\widetilde{m}$-tree, whose height may be limited in the radius of graph to cover the global receptive field. In view of the limited height, as well as the small $\widetilde{m}$ value, a small amount of sampling times could well preserve most information of topology structures as well as node signals.

To efficiently encode the local hcr-walk around each reference node, we project the hcr-walk onto an ordered space to form image-like grid-shape data, which better favors those conventional convolution networks. Thus, a 2-D convolution filtering can be performed in the normalized hcr-walk space to encode the correlation of within-walk adjacencies and cross adjacent walks. To characterize the uncertainty of latent feature representation, we design a 2-D variational convolution block (VCB) inspired by the recent variational inference, rather than directly adopting 2-D convolution filtering. The benefit of variational convolution is that the probability distribution of random sampled walks could be well modeled and the performance could be further boosted. The proposed hcr-walk can be framed in an end-to-end neural network with high running speed even on large-scale graphs.

Our contributions are three-fold:

1) We propose the hcr-walk to describe local topology structures of graphs, which can efficiently mitigate the problem of exponentially-explosive sampling times occurring in the original random walk.

2) We project the hcr-walk onto the grid-shape space and then introduce 2-D variational convolution to describe the uncertainty of latent features, which makes the convolution operation on graphs more efficient and flexible, just as the standard convolution on images, and well support batch-processing.

3) We experimentally validate the efficiency and effectiveness of our proposed VG-GCN, which has a high-efficient computation speed, and comparable or even better performance when compared with those baseline GCNs.

II. Related Work

In this section, we will introduce previous literatures which are related to our work. Generally, they can be divided into three parts: graph convolutional neural networks, random walk, and variational inference.

A. Graph Convolutional Neural Networks

With the rapid development of deep learning, more and more graph convolutional neural network models [34]–[37] are proposed to deal with the irregular data structure of graphs. Compared with regular convolutional neural networks on structured data, this is a challenge since each node's neighborhood size varies in graphs, while the regular convolutional operation requires fixed local neighborhood. To address this problem, the graph convolutional neural networks fall into two categories, spectral-based convolution and spatial-based convolution. Spectral-based filtering method was first proposed by Bruna *et al.* [15]. It defines the filter operators in spectral domain, and then implements a series of convolution operations through the Laplace decomposition of graphs. Because the spectrum filter includes the process of matrix eigenvalue decomposition, the computational complexity is generally high, especially for graphs with a large number of nodes. To alleviate the computation burden, Defferrard *et al.* [16] proposed a local spectral filtering method, which approximates the frequency responses with the Chebyshev polynomial. Spatial-based filtering methods simulate the image processing approach of regular convolutional neural networks, and employ convolution based on nodes' spatial relations. The general approach of spatial convolution is to construct the regular neighborhood of nodes through sampling (discarding a part of nodes if the neighbor number exceeds while repeating a part of nodes if the neighbor number is insufficient), and then carry out the convolution operation with the convolution kernel of rules. According to whether the data to be predicted can be known from the model in training stages, it can be divided into transductive learning [38] and inductive learning [39].

Specifically, for the inductive learning, the data to be predicted is not accessible during training, and the data of the model may be in an "open world".

B. Graph Clustering

Graph clustering aims to depart a complete graph $\mathcal{G}$ to K disjoint subsets $\mathcal{V}_1,\ldots,\mathcal{V}_K$. The vertices within clusters are densely connected, while the vertices in different clusters are sparsely connected. Graph clustering plays an important role in community detection [40]–[42], telecommunication networks [43], and email analysis [44]. He *et al.* [45] proposed a dubbed contextual correlation preserving multiview featured graph clustering (CCPMVFGC) for discovering clusters in graphs with multiview vertex features. To address the problem of identifying clusters in attributed graphs, Hu *et al.* [46] proposed an inductive clustering algorithm (MICAG) for attributed graphs from an alternative view. To overcome the problem where common clustering models are only applicable to complex networks where the attribute information is composed of attributes in binary form, Hu *et al.* [47] proposed a three-layer node-attribute-value hierarchical structure to describe the attribute information in a flexible and interpretable manner.

C. Random Walk

Random walk is an effective method to get graph embedding. It is especially useful in the situation that the graph is partially visible or the graph is too large to measure in its entirety. Given a graph composed of nodes and the connection between nodes, walk paths can be obtained by selecting the start nodes and then executing random walk with certain sampling strategies. The commonly used random walk strategies generally include the truncated random walk [48]–[50] and the second-order random walk [51]. Analogous to tasks in natural language processing where all the nodes in a graph constitute a dictionary, a walk path is regarded as a sentence, and a node in the path is regarded as a word. Graph embedding can be learned by adopting the continuous bag-of-words (CBOW) model [52] or the Skip-gram model [53]. Specifically, the CBOW model uses context to predict the central node embedding, while the Skip-gram model predicts the context nodes based on the central node embedding. Among them, Skip-gram model is the most widely used one. Skip-gram aims to maximize the co-occurrence probability among the words that appear within a window w. Among various graph embedding methods based on random walk, DeepWalk and node2vec are two typical examples.

1) DeepWalk: DeepWalk preserves the high-order proximity between nodes by maximizing the co-occurrence probability of the last k nodes and next k nodes in the path centered at vertex v_i: maximizing $\log Pr(v_{i-k},\ldots,v_{i-1},v_{i+1},\ldots,v_{i+k}|Y_i)$, where $2k+1$ is the walk length. It produces lots of walks and optimizes the logarithmic probability of all paths.

2) node2vec: node2vec controls the partial random walk on the graph by two hyper-parameters p and q, which can be seen as providing a trade-off between breadth-first (BFS) and depth-first (DFS) graph searching, and hence produces higher-quality and more informative embedding than DeepWalk.

D. Variational Inference

Variational inference is a fast and effective method from machine learning that approximates probability densities for a large amount of data. In Bayesian statics, the inference of unknown quantities can be regarded as the calculation of posterior probability, which is usually difficult to calculate. The usual approach is to make an approximation using the Markov Chain Monte Carlo (MCMC) algorithm [54], which is slow for large amounts of data due to the sampling. Different from MCMC, the idea behind variational inference is to first posit a family of densities based on latent variables, and then to find the member of that family which is close to the target. Specifically, the degree of closeness is usually described by the Kullback-Leibler (KL) divergence. Moreover, Kingma and Welling [55] proposed the reparameterization trick to solve the non-differentiable problem in optimization caused by the sampling of the involved latent variables.

III. VG-GCN

In this section, we will introduce our VG-GCN in detail. We first define the notations used in this paper and overview the entire architecture of VG-GCN, then introduce the main modules of VG-GCN, including hcr-walk, gridding, and variational convolution.

A. Notations

We use $\mathcal{G}=\{\mathcal{V},\mathcal{E},\mathbf{X}\}$ to represent a graph where $\mathcal{V},\mathcal{E}$ denote the sets of nodes and edges, respectively, and the numbers of the node and edge sets are denoted as $N=|\mathcal{V}|$ and $e=|\mathcal{E}|$. Each node is associated with a d-dimension signal/feature vector, so the signals of graph $\mathcal{G}$ form a matrix $\mathbf{X}\in\mathbb{R}^{N\times d}=[\mathbf{x}_1^T;\ldots;\mathbf{x}_N^T]$ where $\mathbf{x}_i$ is the signal vector of the i-th node. For the semi-supervised node classification, the expected output is a label matrix $\mathbf{Y}\in\mathbb{R}^{N\times c}$ in which c is the total class number. To describe the adjacent relationship among nodes, the adjacency matrix is defined as $\mathbf{A}\in\mathbb{R}^{N\times N}$ where the element $A^{(i,j)}$ located at the i-th row and j-th column indicates the connection weight between the i-th and j-th nodes. In hcr-walk, we use L to represent the maximum walk steps, T for the sampling times from each starting nodes ($T=T_0+T_1+\cdots+T_h$, where T_0 for original graph, and $T_1\cdots T_h$ for different hierarchical coarsening graphs. In our experiments $h=2$), and $\widetilde{m}$ for the maximum number of edges of the anchor nodes. The average value of node degrees is denoted by m. Besides, in variational convolution, the 2-D kernel size of convolutional filters is denoted as $k_1\times k_2$.

B. Overview

The overall network framework is shown in Fig. 1, where the input is the graph-structured data. To illustrate the convolution process, we take the corresponding local subgraph (i.e., local receptive field) around one node as an example. In order to aggregate topological information of different levels of nodes, we execute graph coarsening on the input graph according to different coarsen ratios, and then random walk on these hierarchical graphs to capture the local structures; see Section III-C. Through random walk based on

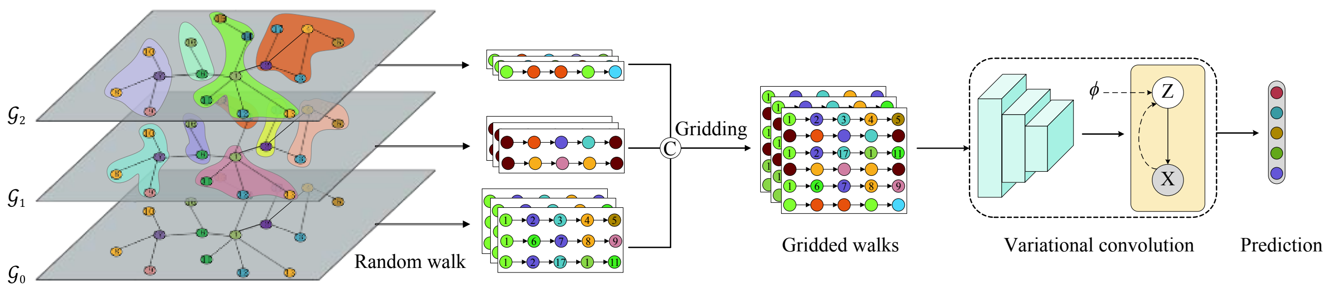

Fig. 1. The framework of our proposed VG-GCN. The description can be found in Section III-B. Specifically, $\mathcal{G}_0$ denotes the origin graph, and $\mathcal{G}_1$, $\mathcal{G}_2$ are the coarsened graphs with different coarsening ratios.

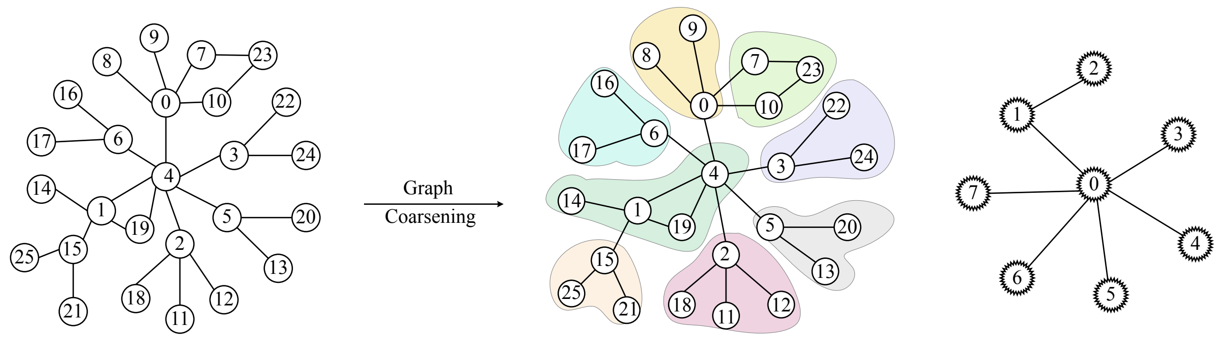

Fig. 2. Graph coarsening with the coarsen ratio of 0.3.

hierarchically coarsening, the hcr-walk could effectively mitigate the problem of exponentially-explosive increases of sampled walks incurred in the original random walk as sufficient sampling could well guarantee to cover graph structures. Next, the sampled walks are adaptively gridded into an ordered space through the computation of correlation to the first principal component of random-walks; see Section III-D. The gridding walks are spanned to a 2-D plane of $T \times L$, which thus favors the conventional convolution. If stacking multi-dimensional signals, the gridded representation of local subgraph is a 3-D tensor of $d \times T \times L$. Thus, the high-efficient and powerful CNNs run on images can be extended for this case to encode the correlation of within-walk adjacencies and cross adjacent walks. To describe the variations of latent feature representation, we introduce variational inference into the 2-D convolution process, referred to as the variation convolution block, to encode the distribution of random-walks therein. Finally, the output features of variation convolution are passed through a fully connected layer and a softmax function for node classification.

C. Hierarchically-Coarsened Random Walk

The random sampling strategy is introduced to characterize the topology structure of local receptive fields. There are two critical questions which need to be solved: i) random sampling should well preserve topology property of original graph, and ii) sampling times should be as few as possible for high-efficient computation as well as low-memory requirement. The first condition dedicates to the accuracy of representation, while the second one focuses on the efficiency of learning. Random walk can satisfy the first condition well under the sufficient samplings, but the sampling complexity heavily depends on node degrees during traversals on graph. Given the average node degree m and walk length t, the combinatorial number of walks is m^t, which has the exponentially-explosive quantity when m is a bit large (especially density graph) even for a small walk step. For example, suppose $m = 10$ and $t = 8$, the combinatorial walks can reach the number of 10 000 000 for each staring node. In order to guarantee the accuracy in the first condition, the practical sampling times might be huge even if a small sampling ratio is taken. It will cause high-computation burdens and require high storage space.

To address this problem, we extend the random walk to hierarchically-coarsened random walk by leveraging the powerful topology preservation ability of random walk and the high efficiency of random aggregation. The schematic diagram of graph coarsening is shown in Fig. 2. If we use $\mathcal{G}_1 = \{\mathcal{V}_1, \mathcal{E}_1, \mathbf{X}_1\}$ to denote the original graph before coarsening and p_1 to denote the coarsen ration, the coarsening graph for $\mathcal{G}_1$ can be represented as $\mathcal{G}_2 = \{\mathcal{V}_2, \mathcal{E}_2, \mathbf{X}_2\}$, where $|\mathcal{V}_1|$ and $|\mathcal{V}_2|$ represent the numbers of nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$. Specifically, $|\mathcal{V}_1|$ and $|\mathcal{V}_2|$ satisfy $|\mathcal{V}_2| = \lceil |\mathcal{V}_1| \times p_1 \rceil$, where $\lceil \cdot \rceil$ means round up to an integer. We randomly seeded $\mathcal{G}_1$ with $|\mathcal{V}_2|$ cluster seeds based on the coarsen ratio p_1, and the nodes in $\mathcal{G}_1$ converge to each cluster seed according to the adjacency relationship in $\mathcal{E}_1$. The connection relation of coarsening graph $\mathcal{G}_2$ is defined by the connection and the number of connections among the clusters. For example, $\mathcal{V}_2{}^{(i)}$ and $\mathcal{V}_2{}^{(j)}$ denote node i and node j in $\mathcal{G}_2$, they are composed of m_1 and m_2 nodes in $\mathcal{G}_1$

$$\mathcal{V}_2{}^{(i)} = \{\mathcal{V}_1{}^{(k_1)}, \mathcal{V}_1{}^{(k_1+1)}, \ldots, \mathcal{V}_1{}^{(k_1+m_1-1)}\}$$
$$\mathcal{V}_2{}^{(j)} = \{\mathcal{V}_1{}^{(k_2)}, \mathcal{V}_1{}^{(k_2+1)}, \ldots, \mathcal{V}_1{}^{(k_2+m_2-1)}\}. \quad (1)$$

Considering that the number of connections between cluster nodes is not consistent, we processed the coarsening graph into a weighted graph (For consistency, the original unweighted graph can convert to the weighted graph according to the node's degree). The weight ${A_2}^{(i,j)}$ between ${\mathcal{V}_2}^{(i)}$ and ${\mathcal{V}_2}^{(j)}$ can be computed as

$$ {A_2}^{(i,j)} = \frac{\sum_{v\in {\mathcal{V}_2}^{(i)}, u\in {\mathcal{V}_2}^{(j)}} {\mathcal{E}_1}^{(u,v)}}{\sum_{v\in {\mathcal{V}_2}^{(i)}, u\in \mathcal{V}_1} {\mathcal{E}_1}^{(u,v)}}. \tag{2} $$

The feature of the node in coarsening graph is related to the nodes belonging to the cluster. In graph learning task, the larger the degree of the node, the more important the node (for example, in social networks, large degree nodes represent popular users and play a more important role in pattern mining). Therefore, the degree of nodes in the cluster can be used as the weight of their feature synthesis

$$ {\mathbf{X}_2}^{(i)} = \sum_{v\in {\mathcal{V}_2}^{(i)}} \frac{D_v}{\sum_{u\in {\mathcal{V}_2}^{(i)}} D_u} {\mathbf{X}_1}^{(v)} \tag{3} $$

where D_v denotes the degree of node v. We construct the hierarchical coarsening graphs by coarsening the input original graph according to different coarsen ratios, and then execute random walk on the original graph and hierarchical graphs. We employ the alias sampling method [56] to sample truncated random walks from the discrete probability distribution in $\mathbf{A}_i$ for hierarchical graph $\mathcal{G}_i$, and concatenate the paths which belong to the same start node. The walks on different hierarchical graphs can aggregate the graph topologies with few sampling times, high-efficient computation, and low-memory requirement.

D. Gridding

One problem of the classic random walk is the irregularity along different paths caused by random sampling, which makes it rather challenging to exploit the underlying local correlation across adjacent walks. To solve this problem, we perform gridding on the output features of random-walks to project them into an ordered space. Based on this operation, the gridding paths are spanned to a 2-D plane based on two axes, i.e., T and L, representing the time of sampling and the number of walk step, respectively. The operation of gridding brings one notable benefit that the high-efficient and powerful CNNs run on images can be extended for the paths to jointly encode the correlation of within-walk adjacencies and cross adjacent walks. For multi-dimensional signals, the gridded representation of local subgraph may be a 3-D tensor of $d\times T\times L$, which is also suitable for the application of CNN.

To adaptively capture the correlation among random paths, we conduct gridding from the perspective of distribution and consider each sampled path based on its correlation to the first principal component of hcr-walk. For the hcr-walk on one node with the representation denoted as $\mathcal{D}\in\mathbb{R}^{d\times T\times L}$, we first split it into a set of path samples denoted as $\{\mathbf{d}_1,\ldots,\mathbf{d}_T\}$ along the sampling time axis T, where $\mathbf{d}_i\in\mathbb{R}^{(d\times L)}$ is the vectorized representation of the i-th path sample. For gridding, the clustering center of path samples is first calculated as

$$ \mathbf{d}_c = \frac{1}{T}\sum_{i=1}^{T}\mathbf{d}_i. \tag{4} $$

Then, the correlation between the i-th path sample and the cluster center is defined as follows:

$$ S_i = \frac{\mathbf{d}_i^T\mathbf{d}_c}{\|\mathbf{d}_i\|\times\|\mathbf{d}_c\|}. \tag{5} $$

Then, the path of each node is gridded based on its corresponding value correlation related to the cluster center.

E. Variational Convolution

Variational convolution is used to characterize the uncertainty of latent feature representation, which is inspired by the variational inference. We stack multiple 2-D convolutional layers on the ordered grid-like feature map $\widetilde{\mathbf{D}}$: $H = f(\widetilde{\mathbf{D}})$, where $\widetilde{\mathbf{D}}\in\mathbb{R}^{b\times d\times T\times L}$ is the output of gridding operation, and $f(\cdot)$ denotes the convolutional layers. We view the path number T and the walk length L as the height and width in one image, and use the feature dimension d to represent the in-channel of convolutional layers. Because of the dimensions of T and L contain the practical structure significance of the graph, instead of the standard convolution kernel $(k\times k)$, we employ an irregular convolution kernel $(k_1\times k_2)$ to better aggregate the neighborhood structure information sampled by the hcr-walk and the node information in different size of receptive field. For one start node, the irregular kernel collects the depth information from the L dimension and the breadth information from the T dimension, which performs as the local aggregation process with an increasing receptive field. Ideally, if $L=2$ and the central node's first neighbor number is T, our convolutional layer is equal to one layer of GCN.

For each start node v_i, we get the feature representation $\mathbf{H}_i$ from stacked convolutional layers. Then, we adopt VCB to describe the probability distribution of node's aggregated feature representation $\mathbf{H}_i$. The structure of VCB is shown in Fig. 3. According to the theory of variational inference, the marginal likelihood of $\mathbf{H}_i$ can be written as

$$ \log p_\theta(\mathbf{H}_i) = KL(q_\phi(\mathbf{Z}|\mathbf{H}_i)\|p_\theta(\mathbf{Z}|\mathbf{H}_i)) + \mathcal{L}(\theta,\phi;\mathbf{H}_i). \tag{6} $$

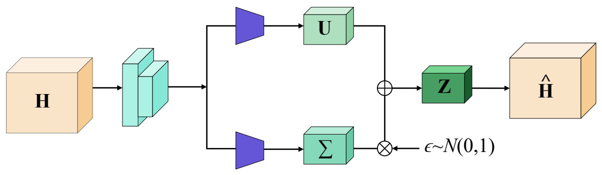

Fig. 3. Variational convolution block (VCB).

The first term denotes the KL divergence of the approximate from the true posterior, and the second term $\mathcal{L}(\theta,\phi;\mathbf{H}_i)$ is called the variational lower bound. According to the conditional probability formula

$$ p_\theta(\mathbf{Z}|\mathbf{H}_i) = \frac{p_\theta(\mathbf{Z},\mathbf{H}_i)}{p_\theta(\mathbf{H}_i)} \tag{7} $$

and the definition of KL divergence

$$KL(q_\phi(\mathbf{Z}|\mathbf{H}_i)\|p_\theta(\mathbf{Z}|\mathbf{H}_i)) = \mathbb{E}_{q_\phi}\log\frac{q_\phi(\mathbf{Z}|\mathbf{H}_i)}{p_\theta(\mathbf{Z}|\mathbf{H}_i)}. \quad (8)$$

Then, (6) can be rewritten as

$$\mathcal{L}(\theta,\phi;\mathbf{H}_i) = -KL(q_\phi(\mathbf{Z}|\mathbf{H}_i)\|p_\theta(\mathbf{Z})) + \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{H})}[\log p_\theta(\mathbf{H}_i|\mathbf{Z})]. \quad (9)$$

We can use Monte Carlo sampling [57] to approximate the expectation of the second logarithmic likelihood term in (9), and adopt reparameterization trick $(\mathbf{Z}=g(\mathbf{H},\epsilon), \epsilon\sim N(\mathbf{0},\mathbf{1}))$ to make the process of calculating parameter gradient be differentiable. Let the prior of $\mathbf{Z}$ denote the standard normal distribution: $\mathbf{Z}\sim N(\mathbf{0},\mathbf{1})$, the resulting estimator for this model and datapoint $\mathbf{H}_i$ is

$$\mathcal{L}(\theta,\phi;\mathbf{H}_i) \simeq \frac{1}{2}\sum_{j=1}^{J}(1+\log(\mathbf{\Sigma}_j^2)-\mathbf{U}_j^2-\mathbf{\Sigma}_j^2) + \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{H})}[\log p_\theta(\mathbf{H}_i|\mathbf{Z})] \quad (10)$$

where $\mathbf{Z}=g(\mathbf{H},\epsilon)=\mathbf{U}+\mathbf{\Sigma}\odot\epsilon$, and $\epsilon\sim N(\mathbf{0},\mathbf{1})$. Different from common variational models, we creatively use the two-dimensional convolution kernel to generate mean matrix $\mathbf{U}$ and covariance matrix $\mathbf{\Sigma}$.

$$\mathbf{U}_j^{(x,y)} = \sigma(\mathbf{b}_j + \sum_{p=0}^{k_1-1}\sum_{q=0}^{k_2-1} w_j^{pq}\mathbf{H}^{(x+p)(y+q)}) \quad (11)$$

$$\mathbf{\Sigma}_j^{(x,y)} = \mathbf{e}^{\sigma(\mathbf{b}_j+\sum_{p=0}^{k_1-1}\sum_{q=0}^{k_2-1} w_j^{pq}\mathbf{H}^{(x+p)(y+q)})} \quad (12)$$

where j denotes the j-th feature map, (x,y) denotes the local position of (T,L), k_1 and k_2 are the height and width of the kernel, and w_j^{pq} is the value at the position (p,q) of the kernel connect to the j-th feature map.

IV. Algorithm and Analysis

Our VG-GCN algorithm flow is shown in Algorithm 1. VG-GCN is an end-to-end online learning model. The inputs are the adjacency matrices $\mathbf{A}$ and feature matrices $\mathbf{X}$ of the starting nodes, and the outputs are the corresponding predicted labels $\hat{\mathbf{Y}}$. The loss function $\mathcal{L}$ of the model consists of two parts: cross-entropy loss $\mathcal{L}_1$ and variational lower bound $\mathcal{L}_2$ in (10).

$$\mathcal{L}_1 = -\sum_{i=1}^{n} y_i\log(\hat{y}_i), \quad \mathcal{L}_2 = \mathcal{L}(\theta,\phi;\mathbf{H}_i) \quad (13)$$

$$\mathcal{L} = \mathcal{L}_1 + \alpha\mathcal{L}_2 \quad (14)$$

where α is a hyper parameter.

Algorithm 1 VG-GCN

```
Input: Original input graph $\mathcal{G}_0=\{\mathcal{V}_0,\mathcal{E}_0,\mathbf{X}_0\}$, weighted adjacency matrix $\mathbf{A}_0$, coarsen ratio set $p=\{p_1,p_2\}$, walk length $L$, repeat times $T=T_0+T_1+T_2$, start nodes index $inds\in\mathbb{R}^n$, train data labels $\mathbf{Y}\in\mathbb{R}^{n\times c}$ and model parameters $\mathbf{W}$.
Output: Start nodes' labels prediction: $\hat{\mathbf{Y}}\in\mathbb{R}^{n\times c}$
1) Construct hierarchical coarsening graphs $\mathcal{G}_1=\{\mathcal{V}_1,\mathcal{E}_1,\mathbf{X}_1\}$ and $\mathcal{G}_2=\{\mathcal{V}_2,\mathcal{E}_2,\mathbf{X}_2\}$ by graph coarsening operation according to coarsening ratio set $p$, calculate the hierarchical weighted adjacency matrix $\mathbf{A}_1$ and $\mathbf{A}_2$
2) Execute random walk for start nodes $inds$ with $L$ walk length and repeat $T_i$ times on hierarchical graphs $\mathcal{G}_i$, gather path nodes' features from $\mathbf{X}_i (i=0,1,2)$. Concatenate the different feature maps $\mathbf{D}_i\in\mathbb{R}^{n\times d\times T_i\times L}$ from different hierarchical graphs, get the feature map $\mathbf{D}\in\mathbb{R}^{n\times d\times T\times L}$
3) for epoch in range[0, num_epochs] do
4)     Gridding $\mathbf{D}$, get the regular and sorted feature map $\widetilde{\mathbf{D}}$
5)     Execute 2-D convolution operations on the feature map $\widetilde{\mathbf{D}}$, $\mathbf{H}=f(\widetilde{\mathbf{D}})$, $f$ is stacked convolution layers
6)     Adopt VCB to $\mathbf{H}$, simulate the mean $\mathbf{U}$ and covariance matrix $\mathbf{\Sigma}$ by convolution kernels, and sample noise $\epsilon$ from standard normal distribution ($\epsilon\sim N(\mathbf{0},\mathbf{1})$). Calculate the latent variable $\mathbf{Z}=\mathbf{U}+\mathbf{\Sigma}\odot\epsilon$
7)     Predict nodes' labels $\hat{\mathbf{Y}}$ with $\mathbf{Z}$: $\hat{\mathbf{Y}}=\mathrm{SoftMax}(\mathbf{Z}\times\mathbf{W}_k+\mathbf{b}_k)$, $k$ denotes $k$-th layer
8)     Calculate model loss $\mathcal{L}=\mathcal{L}_1+\alpha\mathcal{L}_2$ (see (13)), back propagation to update parameters.
9) end
```

For each starting node, we sample T paths with length L in hcr-walks, so as to ensure that the neighborhood nodes could be covered as much as possible. The complexity of VG-GCN is $O(nTL\sum_{k=1}^{K} d_k d k_1 k_2)$, where n is the number of nodes to be processed (i.e., training set and testing set), k is the k-th layer, d_k is the output dimension of the k-th layer, and k_1 and k_2 denote the height and width of convolution kernels, respectively. The complexity of GCN is $O(Nm^l\sum_{k=1}^{K} d_k d)$ and $N\gg n$ in commons (i.e., in Pubmed dataset, $n=1060$ and $N=19\,717$), indicates the speed of our proposed VG-GCN.

V. Experiments

In this section, we comprehensively evaluate the effectiveness of our method on five widely used public datasets: Cora, Citeseer, Pubmed [58], AMZ PHOTOS [59], and NELL [60]. We first briefly introduce these datasets, then report our experimental results on them and compare the performance with other state-of-the-art methods. Finally, we conduct an ablation study to dissect the proposed model.

A. Datasets

Five public graph-structured datasets are employed to evaluate our proposed method, including three citation network datasets (i.e., Cora, Citeseer, Pubmed), a co-purchase dataset (i.e., AMZ PHOTOS), and a knowledge graph dataset (i.e., NELL). For fair compassion, the dataset split protocols of these citation networks strictly follow the widely used ones in [60]. The overall information about these five datasets is listed in Table I.

1) Cora: Cora is a citation network about machine learning papers categorized into seven classes: case-based, genetic algorithm, neural network, probabilistic method, reinforcement learning, principle learning, and theoretical. In total, Cora contains 2708 nodes and 5429 edges, where each node can be described by a 1433 dimensional vector consisting of

TABLE I
GRAPH DATASETS INFORMATION

Dataset	Nodes	Edges	Degrees	Features	Labels
Cora	2708	5429	4	1433	7
Citeseer	3327	4732	3	3703	6
AMZ PHOTOS	7487	119 043	32	745	8
Pubmed	19 717	44 338	5	500	3
NELL	65 755	266 144	4	61 278	210

0/1-valued elements. The average degree of nodes in Cora is about four. For the protocol on this dataset, there are 5.2% of nodes labeled for training (20 nodes in each class, 140 nodes in total), 500 nodes for validating, and 1000 nodes for testing.

2) Citeseer: Citeseer depicts a citation network of 3327 nodes and 4732 links, where the nodes are divided into six classes. Each node can be described by a 3703 dimensional 0/1-valued vector. For evaluation, 3.6% nodes are labeled for training (20 nodes in each class, 120 nodes in total), and the numbers of nodes in the validation and test sets are 500 and 1000, respectively, which are as same as those in Cora. Each node in Citeseer is connected by three nodes in average.

3) AMZ PHOTOS: AMZ PHOTOS is a Co-purchase dataset. It contains 7487 nodes of 8 classes with 119 043 edges. Each node is described by a 745 dimensions vector, and the average degree of nodes is 32. This dataset is split by 200/4800/2487 for train/val/test.

4) Pubmed: Pubmed contains 19 717 nodes of three classes with 44 338 edges. Each node is described by a term frequency-inverse document frequency (TF-IDF) vector drawn from a dictionary with 500 terms. For the widely accepted protocol on this dataset, there are only 0.3% of nodes for training (20 nodes in each class, 60 nodes in total), 500 nodes for validating, and 1000 nodes for testing. The average degree of each node is about five.

5) NELL: NELL dataset is extracted from the never ending language learning (NELL) knowledge graph [61]. Each relation in NELL links the selected entities (9897 in total) with text descriptions in ClueWeb09 [62], and can be described with a triplet (e_h, r, e_t) where e_h and e_t are the head and tail entity vector, and r denotes the relation between them. By splitting every triplet (e_h, r, e_t) into two edges (e_h, r_1) and (e_t, r_2), a graph of 65 755 nodes (including relations and entities) and 266 144 edges can be obtained, where each node can be described by a 61 278 dimensional vector and approximately connected by four nodes.

B. Baseline Methods

To verify the superiority of our proposed VG-GCN model, various state-of-the-art methods are used for performance comparison. Basically, the results of these baseline methods are obtained either according to their reported performance in previously literatures, or through conducting the experiments based on their released public codes. For the baseline methods of our implementation, sophisticated hyper-parameter fine-grained tuning is performed to report their performances.

DeepWalk [48] is a generative model for graph embedding, which samples multiple walk paths from a graph by the truncated random walk, and learns the representation by regarding the paths as sentences, and path nodes as tokens in natural language processing (NLP). The source code for DeepWalk is publicly available[1]. Planetoid [60] is inspired by the Skipgram [53] model from NLP, and it embeds a graph through both positive and negative samplings while considering the node information and graph structure. The source code of Planetoid is available[2]. Chebyshev [16] designs a fast localized graph convolution by employing localized filters with polynomial parametrization, and adopts graph coarsening procedure to group together similar vertices. Graph convolutional networks (GCN) [17] updates the feature expression of the central node by synthesizing the information of the gradually expanding sensing nodes in the field. GCN's source code is publicly available[3]. Graph attention networks (GAT) [31] applies the attention mechanism to graph convolution. It calculates the attention coefficient between central node and its neighborhood nodes to express the different contribution of neighbor connections. Moreover, GAT has an additional sparse version which is also involved as the baseline. For these two versions of GAT, the performance is reported in Table II. GAT's source code is publicly available[4]. Dual graph convolutional networks (DGCN) [63] executes dual graph convolution based on the adjacency matrix and positive pointwise mutual information (PPMI) matrix, respectively, and combines the output of different convolved data transformations. The source code of DGCN is publicly available[5]. Graph learning-convolutional networks (GLCN) [64] learns a discriminative S to replace the adjacency matrix A for graph convolution based on the topological between nodes and on high-dimensional manifold. gLGCN [65] adds the local invariance constraint to the loss function that the same label samples should have the same data distribution. Hypergraph neural networks (HGNN) [66] designs a hypergraph structure, where one edge can connect multiple nodes. Then, robust node representation can be learned by aggregating the node information to the hyper-edge and then returning the integrated information to each node.

C. Experiment Setting

The parameters of our VG-GCN model are traversed in certain ranges and finally set when the best performance on the validation set is obtained. For the basic architecture of the VG-GCN model, there are two convolutional layers in VCB. The coarse layers number is 2 and the coarse ratios are 0.8 and 0.4, respectively. In the hcr-walk process, the number of walks, denoted as T, is set to 15, and the hcr-walk length L is

[1] https://github.com/phanein/deepwalk

[2] https://github.com/kimiyoung/planetoid

[3] https://github.com/tkipf/gcn

[4] https://github.com/PetarV-/GAT

[5] https://github.com/ZhuangCY/DGCN

TABLE II
PERFORMANCE OF GRAPH NODE CLASSIFICATION, COMPARED WITH DEEPWALK, PLANETOID, CHEBYSHEV, GCN, GAT, DGCN, GLCN, GLGCN, AND HGNN METHODS

Method	Cora	Citeseer	AMZ PHOTOS	Pubmed	NELL
DeepWalk [48]	67.2%	43.2%	78.82%	65.3%	58.1%
Planetoid [60]	75.7%	64.7%	68.34%	77.2%	–
Chebyshev [16]	81.2%	69.8%	79.32%	74.4%	–
GCN [17]	81.4%	70.5%	92.08%	79.0%	66.0%
GAT [31]	83.0%	72.5%	53.40%	79.0%	–
DGCN [63]	82.5%	**72.6%**	91.07%	79.3%	74.9%
GLCN [64]	**85.5%**	72.0%	91.25%	78.3%	–
gLGCN-F	82.2%	70.8%	–	79.2%	–
gLGCN-L	82.7%	71.3%	–	79.2%	–
gLGCN-F-L [65]	83.3%	71.4%	–	79.3%	–
HGNN [66]	81.6%	–	–	80.1%	–
VG-GCN	82.7%	71.5%	**92.13%**	**81.3%**	**79.1%**

5. For the convolutional layers, the sizes of convolution kernels are set to 5×3, while in the VCB the convolution kernel sizes are both set to 1×1 to produce the mean and covariance matrices, respectively. During the training process, we run the model for 500 epochs with the learning rate of 0.01 and dropout rate of 0.5 for tuning the network parameters.

D. Experiment Results

The experimental results of our proposed VG-GCN on the three citation datasets (Cora, Citeseer, and Pubmed), one co-purchase dataset (AMZ PHOTOS), and large-scale dataset of knowledge graph (NELL) are reported in Table II. These performances are also compared with various state-of-the-art methods, where the metric of accuracy is employed for quantitative evaluation of the semi-supervised node classification. Our VG-GCN obtains the best results on AMZ PHOTOS, Pubmed, and NELL datasets (there are 0.05% performance gain on AMZ PHOTOS, 1.2% on Pubmed, and 4.2% on NELL), and achieves the competitive performances on the Cora and Citerseer datasets.

We calculated the average and variance of node degree of five datasets. We found that the degree variances of two small datasets (Cora and Citeseer) are small (27.3 and 11.4, respectively), while the other three large scale datasets (AMZ PHOTOS, Pubmed, and NELL) have the larger degree variances (55.2, 1852.3, and 2262.6, respectively). Therefore, we speculate that the reason our VG-GCN does not achieve the best performance on the two small datasets may be that the two dataset have fewer random walk patterns and are more likely to fall into over-smoothing during the training process. This also shows that our hierarchical coarsening and random walk can model complex data patterns.

Besides the competitive performance, it should be especially noticed that our model is more advantageous in computation efficiency compared with all other baseline methods. We present the time costs of our VG-GCN for running one epoch on Cora and Pubmed datasets in Table III, and compare them with GAT, DGCN, and GCN. For Cora, the smallest dataset, GAT takes about 7 s per epoch while its sparse version takes 1 s, and DGCN takes about 0.5 s per epoch. The time consumed by our VG-GCN is 0.05 s, which is much less than those of GAT and DGCN, while almost the same as GCN. However, on the large graph Pubmed (19 717 nodes), our VG-GCN takes about 0.04 s per epoch, and is the fastest compared with the sparse GAT of 2.5 s per epoch, DGCN of 3.2 s, and GCN of 0.6 s.

TABLE III
EACH EPOCH TIME COST ON CORA AND PUBMED, COMPARED WITH GAT, DGCN, AND GCN

Method	Cora	Pubmed
GAT	7 s	10 s
GAT_sparse	1 s	2.5 s
DGCN	0.5 s	3.2 s
GCN	0.05 s	0.6 s
VG-GCN	**0.05 s**	**0.04 s**

The convergence of our VG-GCN on the Nell datasets is shown in Fig. 4, and is compared with that of DGCN which also achieves a considerable performance. Both VG-GCN and DGCN are trained 1000 epochs on NELL dataset. According to Fig. 4, our VG-GCN can converge faster and obtains better performances. In terms of running time and accuracy comparisons, our VG-GCN takes about 6 minutes with the accuracy of 79.1%. In contrast, DGCN takes about 2.8 hours, which is 29.2 times that of VG-GCN, while obtaining an accuracy of 74.9%, which is 4.2% lower.

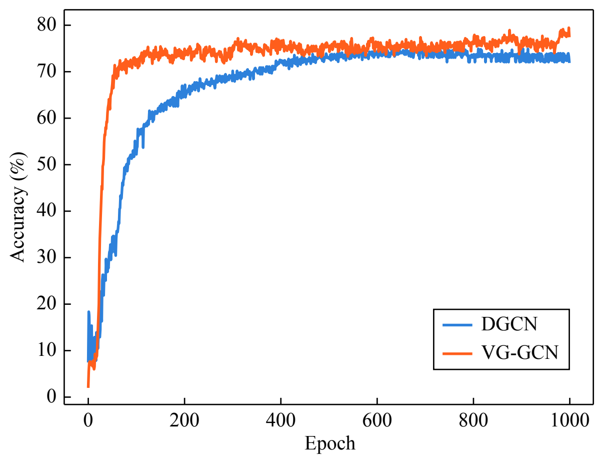

Fig. 4. The convergence on NELL datasets, compared with DGCN.

E. Ablation Study and Parameter Sensitivity

As the proposed VG-GCN achieves promising performance with high computational efficiency, it is interesting for us to dissect the model to evaluate the contribution of each part. Moreover, it is also meaningful to evaluate the sensitivity of those critical parameters in the VG-GCN model to make clear how their variation influences the performance. Therefore, we conduct several additional experiments:

1) Comparison between the hcr-walk and random walk. To verify the superiority of the proposed hcr-walk over random walk, we simply replace the hcr-walk unit with random-walk on original graph, and test the performance on the four public datasets. The results are shown in Table IV.

TABLE IV
COMPARISON BETWEEN HCR-WALK AND RANDOM-WALK

Dataset	Random-walk	Hcr-walk
Cora	81.8%	**82.5%**
Citeseer	71.0%	**71.5%**
Pubmed	79.9%	**81.4%**
NELL	77.9%	**79.5%**

2) Evaluation of VCB. To evaluate the effectiveness of the proposed VCB, we compare its performance with the classic convolutional layer and MLP-VCB on the four datasets, and the results are shown in Table V. Specifically, MLP-VCB means that the calculation of mean and covariance matrices in VCB are revised to be obtained through multilayer perception instead of convolutional layers.

TABLE V
COMPARISON OF CONVOLUTIONAL LAYERS, MLP-VCB, AND VCB

Dataset	Convolutional layers	MLP-VCB	VCB
Cora	81.3%	81.4%	**82.5%**
Citeseer	70.4%	70.9%	**71.6%**
Pubmed	79.6%	79.9%	**81.0%**
NELL	77.4%	78.2%	**79.5%**

Based on the results of the multiple experiments above, we can get the following observations:

1) Hcr-walk outperforms classic random walk and promotes the node classification performance. On all the four evaluated datasets, the classification accuracies of our hcr-walk are higher than those of random walk. The performance gain verifies the effectiveness of our hcr-walk, which constructs coarsening graphs, while avoiding an explosive growth of walk paths with increasing walk steps.

2) VCB is effective to promote the node classification task. Comparing with both convolutional layers and MLP-VCB, VCB obtains better node classification performances with an average performance gain of about 1% on the four public datasets. The performance improvement verifies the superiority of VCB, which encodes the comprehensive correlation of within-walk adjacencies and cross adjacent walks.

Kernel size k_1, k_2, hcr-walk numbers T and length L, and coarsening ratio p are hyper parameters in VG-GCN. To analyze the sensitivity and value selection of each hyper parameter, we designed the following comparative experiments.

1) Kernel Size: The non-square 2D convolution kernel is employed in hcr-walk convolution. The different classification performance with different kernel sizes on three citation datasets is plotted in Fig. 5. From the experimental results of Fig. 5, the irregular convolution kernel with size (5, 3) has better performance on three datasets. Specifically, (5, 3) means the cross-path filtering height is 5 and within-path filtering width is 3.

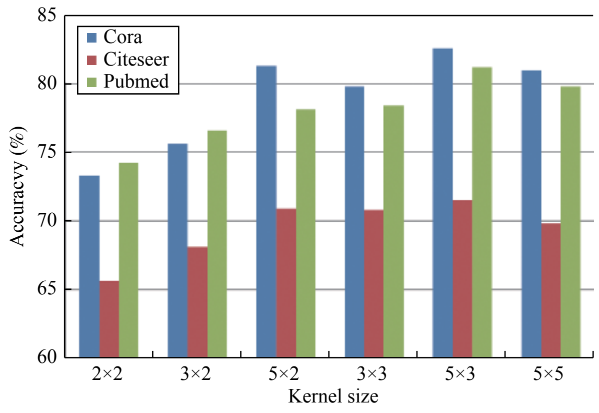

Fig. 5. Classification performance with different kernel sizes on three citation datasets.

2) Hcr-Walk Numbers and Length: The parameter sensitivity experiments with the hcr-walk numbers T and hcr-walk length L belonging to one start node are shown in Figs. 6 and 7. For Cora dataset, 15 hcr-walks are sampled for each node, and the length of 5 is the most appropriate. $(T = 20, L = 5)$ and $(T = 12, L = 6)$ are the best combinations of parameters for Citeseer and Pubmed datasets, respectively.

3) Coarsening Ratio: The hierarchically-coarsened random walk is employed to leverage the powerful topology preservation ability of random walk and the high efficiency of random aggregation. We experimentally compare the effects of one coarsening layer, two coarsening layers and different coarsening ratios p on the classification accuracy of Cora dataset in Table VI. For the two coarsening layers, the coarsening ratio of the second layer is half of the first. Two coarsening layers with ratios of (0.8, 0.4) achieve the best performance.

TABLE VI
SENSITIVITY OF COARSEN LAYER NUMBER AND COARSE RATIO ON MODEL PERFORMANCE ON CORA DATASET. FOR TWO COARSEN LAYERS, WE SET THE COARSEN RATIO OF SECOND LAYER IS HALF OF FIRST LAYER

One layer	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Accuracy	79.9%	80.8%	81.4%	80.9%	82.0%	82.0%	81.3%	81.0%	81.2%
Two layers	(0.9, 0.45)	(0.8, 0.4)	(0.7, 0.35)	(0.6, 0.3)	(0.5, 0.25)	(0.4, 0.2)	(0.3, 0.15)	(0.2, 0.1)	(0.1, 0.05)
Accuracy	81.7%	82.9%	81.2%	80.2%	81.7%	81.2%	82.0%	81.3%	81.9%

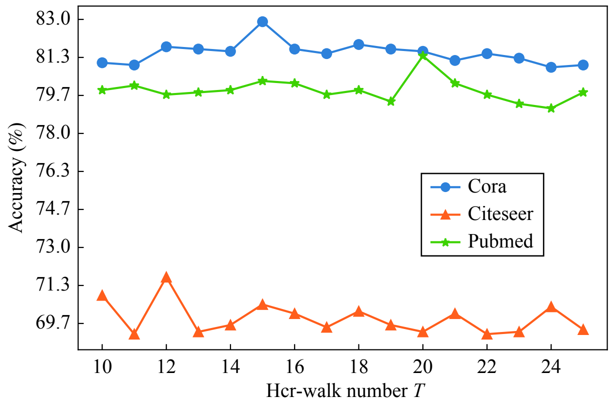

Fig. 6. The results of VG-GCN model with different hcr-walk numbers on Cora, Citeseer, and Pubmed datasets.

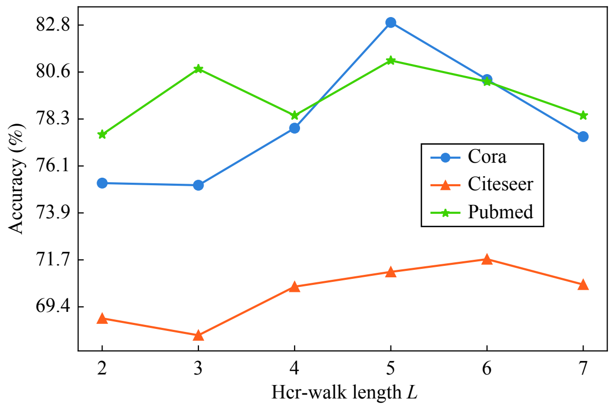

Fig. 7. The results of VG-GCN model with different hcr-walk lengths on Cora, Citeseer, and Pubmed datasets.

4) Convolution Layers: The number of variational convolution layer is traversed in [2,8] on Cora dataset, and the experiments results are listed in Fig. 8. The experimental results show that the too deep network can not surely achieve higher classification accuracy, and may cause performance degradation which may be due to the over-smoothing.

In order to show that our VG-GCN can better capture the graph local invariance than other methods (GCN), we perform a standard set of "evolving" t-SNE [67] plots of the test set embeddings on Cora dataset, given in Fig. 9. The raw features come from the node original feature matrix, and the GCN and

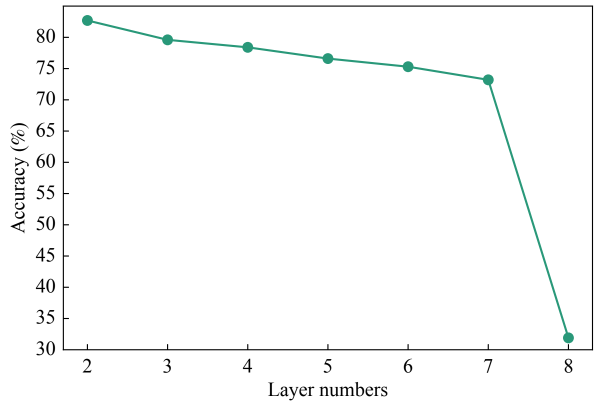

Fig. 8. The results of different variational convolution layers on Cora dataset.

our VG-GCN embeddings come from the output of the last layer respectively. Intuitively, our VG-GCN is best placed to clearly separate different categories of nodes. And from the perspective of quantitative analysis, the Silhouette score [68] of our VG-GCN is the largest (The larger the Silhouette score, the better the clustering effect).

VI. Conclusion

In this paper, we proposed a VG-GCN framework for node classification. We developed the random walk and proposed the hcr-walk to effectively avoid possible exponential explosion of walk paths as the path length increases, and cover the whole neighborhood by coarsening graphs. In order to maintain the permutation invariance of the generated paths belonging to the same node of each epoch, we sorted them after projection and constructed a grid-like feature map for 2-D convolution. Moreover, we designed a 2-D convolution variational inference block to learn the probability distribution characteristics of latent variables in two-dimensional space. As a result, VG-GCN learns the aggregation pattern of node topological neighborhood in an inductive way, which can be easily extended to the inference problem of unknown nodes. Meanwhile, VG-GCN can process large scale graphs quickly with the tensor graph structure and consumes less memory. Experiments on a variety of public datasets verified the effectiveness of our method for solving the node classification problem.

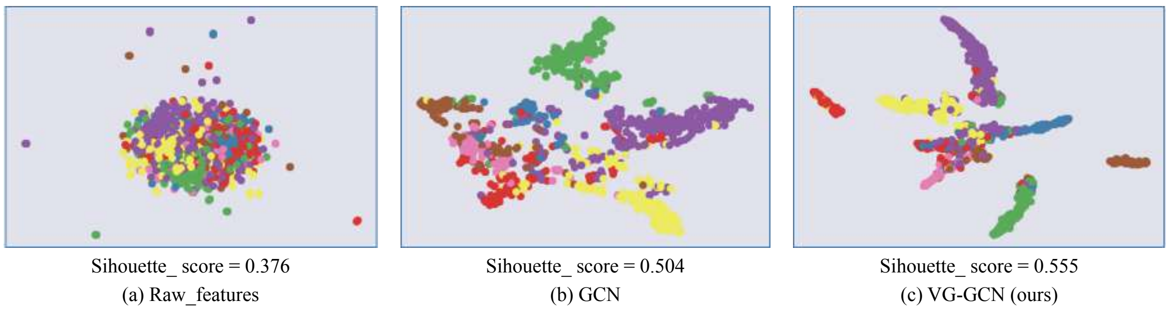

Fig. 9. t-SNE embeddings of the nodes in the test set of Cora citation network from the raw features (**left**), GCN model (**middle**), and our VG-GCN model (**right**). Our VG-GCN performs the best clustering effect of embedding among the three plot, and the Silhouette scores support evidence.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[4] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.

[5] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, 2012.

[6] F. Orsini, D. Baracchi, and P. Frasconi, "Shift aggregate extract networks," *Frontiers in Robotics and AI*, p. 42, 2018.

[7] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins, "The web as a graph: Measurements, models, and methods," in *Proc. Int. Computing and Combinatorics Conf.*, Springer, 1999, pp. 1–17.

[8] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 5410–5419.

[9] K. M. Borgwardt, H.-P. Kriegel, S. Vishwanathan, and N. N. Schraudolph, "Graph kernels for disease outcome prediction from proteinprotein interaction networks," in *Biocomputing*. World Scientific, 2007, pp. 4–15.

[10] X. Luo, H. Wu, H. Yuan, and M. Zhou, "Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors," *IEEE Trans. Cybernetics*, vol. 50, no. 5, pp. 1798–1809, 2019.

[11] T. D. Pham, K. Wardell, A. Eklund, and G. Salerud, "Classification of short time series in early Parkinson's disease with deep learning of fuzzy recurrence plots," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 6, pp. 1306–1317, 2019.

[12] L. Wei and E. Keogh, "Semi-supervised time series classification," in *Proc. 12th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2006, pp. 748–753.

[13] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A datacharacteristic-aware latent factor model for web services QoS prediction," *IEEE Trans. Knowledge and Data Engineering*, 2020.

[14] X. Luo, M. Zhou, S. Li, Y. Xia, Z.-H. You, Q. Zhu, and H. Leung, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data," *IEEE Trans. Cybernetics*, vol. 48, no. 4, pp. 1216–1228, 2017.

[15] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," arXiv preprint arXiv: 1312.6203, 2013.

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 3844– 3852.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learning Representations*, 2017.

[18] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proc. 32nd AAAI Conf. Artificial Intelligence*, 2018.

[19] S. Fu, W. Liu, D. Tao, Y. Zhou, and L. Nie, "Hesgcn: Hessian graph convolutional networks for semi-supervised classification," *Information Sciences*, vol. 514, pp. 484–498, 2020.

[20] S. Fu, W. Liu, Y. Zhou, and L. Nie, "Hplapgcn: Hypergraph p-laplacian graph convolutional networks," *Neurocomputing*, vol. 362, pp. 166–174, 2019.

[21] F. Sichao, L. Weifeng, L. Shuying, and Z. Yicong, "Two-order graph convolutional networks for semi-supervised classification," *IET Image Processing*, vol. 13, no. 14, pp. 2763–2771, 2019.

[22] B. Wu, Y. Liu, B. Lang, and L. Huang, "DGCNN: Disordered graph convolutional neural network based on the gaussian mixture model," *Neurocomputing*, vol. 321, pp. 346–356, 2018.

[23] S. Franco, G. Marco, T. Ah Chung, H. Markus, and M. Gabriele, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, Article No. 61, 2009.

[24] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Neural Information Processing Systems*, 2017.

[25] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[26] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, "Relational inductive biases, deep learning, and graph networks," *Computing Research Repository*, 2018.

[27] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[28] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *Proc. Int. Conf. Learning Representations*, 2016.

[29] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steadystates of iterative algorithms over graphs," in *Proc. Int. Conf. Machine Learning*, 2018, pp. 1114–1122.

[30] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. Learning Representations*, 2019.

[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learning Representations*, 2018.

[32] L. Lovász, "Random walks on graphs: A survey, combinatorics, Paul Erdos is eighty," *Lecture Notes in Mathematics*, vol. 2, no. 1, pp. 1–46, 1993.

[33] X. Hong, T. Zhang, Z. Cui, C. Xu, L. Zhang, and J. Yang, "Fast hyperwalk gridded convolution on graph," in *Proc. Chinese Conf. Pattern Recognition and Computer Vision,* Springer, 2020, pp. 197–208.

[34] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," in *Proc. 20th Int. Conf. Machine Learning*, 2003, pp. 321–328.

[35] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Proc. Artificial Intelligence and Statistics*, 2009, pp. 488–495.

[36] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proc. 21st ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2015, pp. 1365–1374.

[37] C. Morris, K. Kersting, and P. Mutzel, "Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 327–336.

[38] M. Ceci, A. Appice, N. Barile, and D. Malerba, "Transductive learning from relational data," in *Proc. Int. Workshop on Machine Learning and Data Mining in Pattern Recognition*, 2007, pp. 324–338.

[39] R. S. Michalski, "A theory and methodology of inductive learning," in *Machine Learning*. Springer, 1983, pp. 83–134.

[40] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.

[41] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Physical Review E*, vol. 80, no. 5, p. 056117, 2009.

[42] L. Bai, X. Cheng, J. Liang, and Y. Guo, "Fast graph clustering with a new description model for community detection," *Information Sciences*, vol. 388, pp. 37–47, 2017.

[43] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*. Addison-Wesley Longman Publishing Co., Inc., 1986.

[44] A. Chapanond, M. S. Krishnamoorthy, and B. Yener, "Graph theoretic and spectral analysis of enron email data," *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 265–281, 2005.

[45] T. He, Y. Liu, T. H. Ko, K. C. Chan, and Y.-S. Ong, "Contextual correlation preserving multiview featured graph clustering," *IEEE*

Trans. Cybernetics, vol. 50, no. 10, pp. 4318–4331, 2019.

[46] L. Hu, S. Yang, X. Luo, and M. Zhou, "An algorithm of inductively identifying clusters from attributed graphs," *IEEE Trans. Big Data*, 2020. DOI: 10.1109/TBDATA.2020.2964544

[47] L. Hu, K. C. Chan, X. Yuan, and S. Xiong, "A variational Bayesian framework for cluster analysis in a complex network," *IEEE Trans. Knowledge and Data Engineering*, vol. 32, no. 11, pp. 2115–2128, 2019.

[48] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[49] H. Zhang, X. Shang, H. Luan, M. Wang, and T.-S. Chua, "Learning from collective intelligence: Feature learning using social images and tags," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13, no. 1, p. 1, 2017.

[50] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," *Network*, vol. 11, no. 9, p. 12, 2016.

[51] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 855– 864.

[52] L. Qiu, Y. Cao, Z. Nie, Y. Yu, and Y. Rui, "Learning word representation considering proximity and ambiguity," in *Proc. 28th AAAI Conf. Artificial Intelligence*, 2014.

[53] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[54] C. J. Geyer, "Introduction to Markov chain Monte Carlo," *Handbook of Markov Chain Monte Carlo*, vol. 20116022, p. 45, 2011.

[55] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," arXiv: Machine Learning, 2013.

[56] L. Devroye, "Sample-based non-uniform random variate generation," in *Proc. 18th Conf. Winter Simulation*, 1986, pp. 260–265.

[57] C. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer Science & Business Media, 2013.

[58] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. EliassiRad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[59] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," arXiv preprint arXiv: 1911.05485, 2019.

[60] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semisupervised learning with graph embeddings," in *Proc. Int. Conf. Machine Learning*, 2016.

[61] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proc. 24th AAAI Conf. Artificial Intelligence*, 2010.

[62] B. Dalvi, A. Mishra, and W. W. Cohen, "Hierarchical semi-supervised classification with incomplete class hierarchies," in *Proc. 9th ACM Int. Conf. Web Search and Data Mining*, 2016, pp. 193–202.

[63] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graphbased semi-supervised classification," in *Proc. Int. World Wide Web Conf. Steering Committee*, 2018, pp. 499–508.

[64] B. Jiang, Z. Zhang, D. Lin, and J. Tang, "Graph learning-convolutional networks," in *Proc. Computer Vision and Pattern Recognition*, 2019.

[65] B. Jiang and D. Lin, "Graph laplacian regularized graph convolutional networks for semi-supervised learning," arXiv preprint arXiv: 1809.09839, 2018.

[66] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 33, 2019, pp. 3558–3565.

[67] L. V. Der Maaten and G. E. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[68] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, no. 1, pp. 53–65, 1987.

Xiaobin Hong received the B.S. degree from the School of Computer Science and Technology, Anhui University, in 2018, and the M.Sc. degree from the School of Computer Science and Engineering, Nanjing University of Science and Technology, in 2021. His research interests include graph neural networks, graph embedding learning, and their application in the social network.

Tong Zhang received the B.S. degree in information science and technology from Southeast University in 2011, the M.S. degree from the Research Center for Learning Science, Southeast University, in 2014, and the Ph.D. degree from the School of Information Science and Engineering, Southeast University, in 2018. He is now working in the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include pattern recognition, affective computing, and computer vision.

Zhen Cui received the B.S., M.S., and Ph.D. degrees from Shandong Normal University, Sun Yatsen University, and Institute of Computing Technology (ICT), Chinese Academy of Sciences, in 2004, 2006, and 2014, respectively. He was a Research Fellow in the Department of Electrical and Computer Engineering at National University of Singapore (NUS) from 2014 to 2015. He also spent half a year as a Research Assistant on Nanyang Technological University (NTU) from Jun. 2012 to Dec. 2012. Currently, he is a Professor of Nanjing University of Science and Technology. His research interests mainly include deep learning, computer vision and pattern recognition.

Jian Yang received the Ph.D. degree from Nanjing University of Science and Technology (NUST), on the subject of pattern recognition and intelligence systems in 2002. In 2003, he was a Postdoctoral Researcher at the University of Zaragoza. From 2004 to 2006, he was a Postdoctoral Fellow at Biometrics Centre of Hong Kong Polytechnic University. From 2006 to 2007, he was a Postdoctoral Fellow at Department of Computer Science of New Jersey Institute of Technology. Now, he is a Chang-Jiang Professor in the School of Computer Science and Technology of NUST. He is the author of more than 100 scientific papers in pattern recognition and computer vision. His journal papers have been cited more than 4000 times in the Web of Science, and 9000 times in the Web of Scholar Google. His research interests include pattern recognition, computer vision and machine learning. He is/was an Associate Editor of *Pattern Recognition Letters*, *IEEE Trans. Neural Networks and Learning Systems*, and *Neurocomputing*. He is a Fellow of IAPR.