When Work Matters: Transforming Classical Network Structures to Graph CNN

Wenting Zhao[1], Chunyan Xu[2], Zhen Cui[3], Tong Zhang[4], Jiatao Jiang[5], Zhenyu Zhang[6], Jian Yang[7]
[1 2 3 5 6 7] Nanjing University of Science and Technology
[4] Southeast University

***Abstract*—Numerous pattern recognition applications can be formed as learning from graph-structured data, including social network, protein-interaction network, the world wide web data, knowledge graph, etc. While convolutional neural network (CNN) facilitates great advances in gridded image/video understanding tasks, very limited attention has been devoted to transform these successful network structures (including Inception net, Residual net, Dense net, etc.) to establish convolutional networks on graph, due to its irregularity and complexity geometric topologies (unordered vertices, unfixed number of adjacent edges/vertices). In this paper, we aim to give a comprehensive analysis of when work matters by transforming different classical network structures to graph CNN, particularly in the basic graph recognition problem. Specifically, we firstly review the general graph CNN methods, especially in its spectral filtering operation on the irregular graph data. We then introduce the basic structures of ResNet, Inception and DenseNet into graph CNN and construct these network structures on graph, named as G_ResNet, G_Inception, G_DenseNet. In particular, it seeks to help graph CNNs by shedding light on how these classical network structures work and providing guidelines for choosing appropriate graph network frameworks. Finally, we comprehensively evaluate the performance of these different network structures on several public graph datasets (including social networks and bioinformatic datasets), and demonstrate how different network structures work on graph CNN in the graph recognition task.**

***Index Terms*—Graph CNN, Spectral filtering, ResNet, Inception, DenseNet.**

I. Introduction

A graph-structured data sample consists of a finite set of vertices/nodes, together with a set of connections revealing the relationship between unordered pairs of these vertices (named edges). Numerous important high-level applications, especially in the increasingly connected and blended world, can be framed as learning from graph data, including social network [1], [2], protein-interaction network [3], the world wide web data [4], knowledge graph, etc. Among these practical problems, learning an appropriate neural network from such structured graphs becomes the most critical topic.

Recently, convolutional neural networks (CNNs), together with multiple evolved variants, have achieved very promising performance in processing grid-shaped images/videos [5], [6], [7], [8], such as image recognition, object detection, depth estimation, image restoration, object segmentation, etc. LeCun *et al.* [9] originally trained the layer-wise convolutional network with the back-propagation algorithm in 1998. Krizhevsky *et al.* [5] introduced a similar CNN network with a deeper and much wider version, and achieved a breakthrough, outperforming the existing handcrafted features on ILSVRC 2012 competition. GoogLeNet [10] proposed more effective inception module to design a local network topology by adopting multiple receptive field sizes. VGGNet [6], which consisted of 16 convolutional layers, was also very appealing because of its very uniform architecture. He *et al.* [8] introduced a substantially deeper architecture (dubbed Residual Neural Network, ResNet)) with skip connections, which are also known as gated units or gated recurrent units. Dense Convolutional Network (DenseNet) [7] further proposed a densely connected structure, which can connect each layer to every other layer in a feed-forward fashion. However, due to the irregularity and complexity geometric topologies of graph-structured data, these successful CNN structures on representing grid-shaped image/video data cannot be straightforwardly applied to the graph data, especially these elementary operators including convolutional filtering, pooling, translation, etc.

Driven by the developments and limitation of above CNNs, various algorithms devoted to graph CNNs have been proposed in previous literatures. In general, these algorithms can be divided into two main categories according to their ways of conducting convolution on graphs: one can be named as spatial graph CNNs convolving directly on graphs according to the spatial information of nodes, while the other employs spectral filtering based on Spectral Graph Theory [11]. The former one is analogues to CNN which firstly constructs a window of certain size for nodes on the graph and then conducts convolution operation [12]. However, its disadvantage is the loss of structural information of graph data. To overcome this shortcoming, DGCNN [13] proposed dynamic convolutional kernel to adapt to the size and order of the node neighborhood, supporting different scales of convolutional receptive fields. As another kind of spatial graph CNNs, a mixture model networks (MoNet) is proposed in [14], using a parametric patch to match template for convolution operations on graphs or manifolds. Moreover, as a particular instance of MoNet [14], [15] introduces an attention-based architecture to perform node classification of graph-structured data. The latter category of graph CNNs is based on the Spectral Graph Theory [11]. In the graph setting, the graph Laplacian eigenvalues and eigenvectors provide a notion of frequency [16] and spectral filtering has been successfully applied to the field of

Email address: {wtingzhao, cyx, zhen.cui, csjyang}@njust.edu.cn (W. Zhao, C. Xu, Z. Cui and J. Yang), tongzhang@seu.edu.cn (T. Zhang), zhangjesse@foxmail.com (Z. Zhang).

graphs. In [17], CNN was firstly generalized to graph using spectral filtering. On the basis, [18] considered large-scale classification problems with small learning complexity and further made some extension. [19] approximated smooth filters in the spectral domain using Chebyshev polynomials with free parameters that are learned in a neural network-like model. To further improve computation efficiency, [20] presented a first-order approximation of spectral graph convolutions, and in many cases allowed both for significantly faster training times and higher predictive accuracy. Although these approaches have devoted to design more effective convolutional filtering and pooling on graph, very limited attention has been focused on how to transform these successful network structures (including Inception net, Residual net, Dense net, etc.) to establish more promising Graph CNN architectures on graph domains.

In this paper, we pay attention to give a comprehensive analysis of when work matters by transforming different classical network structures to graph CNN, particularly in the basic graph recognition problem. Specifically, we firstly review the general graph CNN methods, especially in its spectral filtering operation on the irregular graph data. We then introduce the basic structures of ResNet, Inception and DenseNet into graph CNN and extend these network structures on graph domains, named as G_ResNet, G_Inception, G_DenseNet. G_ResNet with a similar structure with ResNet [8], is a multi-layer gated graph CNN by learning residual functions with reference to the layer inputs. G_Inception can better consider the different receptive fields of graph signal and also increase the width of the network while keeping the computational budget constant. G_DenseNet can strengthen feature propagation and encourage graph signals of all preceding layers in the graph CNN. In particular, it seeks to help graph CNNs by shedding light on how these classical network structures work and providing guidelines for choosing appropriate graph network frameworks. We comprehensively evaluate the performance of these different network structures on several public graph datasets (including social networks and bioinformatic datasets), and demonstrate how different network structures work on graph CNN in the graph recognition task.

The remainder of this paper is organized as follows. Section II reviews some related works about classical network structures and methods of graph CNN. Section III briefly introduces the approach we used to convolve on the graph in this paper. Section IV presents the three network structures of G_ResNet, G_Inception and G_DenseNet. Section V describes the implementation details, reports the performance of our three graph CNN structures on social networks and bioinformatic datasets and makes some discussions. Finally, Section VI concludes this paper and gives future research direction.

II. Related work

Classical network structures: Deep network learning has been long studied for dealing with these gridded image/video understanding problems [21], [22], [23], [24]. In 1998, LeCun *et al.* [9] trained multilayer neural networks with the back-propagation algorithm and the gradient learning technique, and then demonstrated its effectiveness on the handwritten digit recognition task. Recently, for further boosting its discriminative capability, there has been a resurgence of research interest in the exploration of various network structures. AlexNet [5] is a special type of deep CNN model and achieves a breakthrough, outperforming the existing hand-crafted features on ILSVRC 2012 which contains 1000 object classes. Another deep network structure, namely "Network In Network" (NIN) [25] is proposed to build a micro network with more complex structures to abstract the data within the receptive field, and the proposed 1×1 convolution kernel is later applied in the GoogLeNet model to reduce the dimension and avoid computational explosion. VGGNet [6] consists of 16 convolutional layers and is very appealing because of its very uniform architecture. And It is currently the most preferred choice in the community for extracting features from vision inputs. A more effective inception module, introduced by GoogLeNet [10] model, can be employed to design a local network topology. It convolves with different sized kernels, concatenates the results as input to the next layer, and implements the use of multi-scale features. ResNet [8] can easily improve performance by significantly increasing the depth of network compared to the plain nets (that simply stack layers). And it addresses the degradation problem [26], [27] of accuracy that arises with the network depth increasing by introducing a deep residual learning framework. Through the connection between each convolution layer, DenseNet [7] encourages feature reuse and then learns more and compact features while keeping fewer parameters and less computation by using 1×1 convolution.

Graph CNN: Recent CNN on graph data has raised a progressive direction in the problem of graph recognition. At present, there are two main categories to execute the convolution operation on graphs. One is analogous to the common CNN in the gridded image/video samples, constructing a window of a certain size on the graph, and then doing the convolution by a fixed-size template. The other is based on Spectral Graph Theory[11]. Features of the graph are first transformed into the frequency domain using the Fourier transform and convolved on the spectrum of the graph. Then an inverse Fourier transform is done to transform the feature maps to vertex domain.

Based on image CNN, the work of [12] proposed a general method to learn representation for arbitrary graphs and transformed the data of a graph structure into a structure that CNN can efficiently handle. It mainly consists of two steps: selecting a representative node sequence from the graph structure and finding a convolutional neighborhood for each selected node. However, in this process, structural information of the graph may be lost and also some redundant information can be introduced. Therefore, the paper [13] introduced a Gaussian mixture model by adding a disordered graph convolutional layer (DGCL) to overcome the above problems. Besides, [15] presented masked self-attentional layers to adptively endow different weights to those neighbor vertices of a vertex, and thereby the performance of the model can be improved. To capture temporal evolution of graph sequences, recursive neural networks are introduced into the graph in [28].

Based on the Spectral Graph Theory[11], a generic framework for processing data on graphs is presented in [16], and the fundamental operations such as filtering, translation, modulation, dilation, and downsampling are generalized to the graph setting. [17] proposed two efficient constructions: Spatial Construction and Spectral Construction. Then, [18] extended [17] to large-scale classification problems with small learning complexity, and proposed unsupervised and new supervised graph estimation strategies. In [19], the spectral graph theoretical of CNNs on graphs is formulated and strictly localized spectral filters are given. More importantly, recursive Chebyshev polynomials are utilized to approximate parameterized polynomial filters such that the complexity and efficiency of learning are significantly reduced. [20] was based on spectral graph convolutional neural networks [19], [17] simplifying to a linear function of first-order and conducted the task of transductive node classification in a large-scale network. The paper [29] learned a residual Laplacian matrix for each graph and the optimal distance metric parameters shared among the data, then graphs of arbitrary structure and size can be input into the CNN.

III. Convolution on Graph

As we have introduced, the graph-structured data is with the irregular structure and completely coordinate-free on vertices and edges. To generalize the idea of common CNNs onto graphs, we would like to review the convolution filter on homomorphic graphs/subgraphs, which is different to the convolution on these gridded images/videos, as shown in Fig. 1. Here we mainly introduce the Spectral filtering of graphs, which is also used in our followed experiments.

For a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ represents a set of vertices with the number $|\mathcal{V}| = n$ and $\mathcal{E}$ represents a set of edges. Let $\mathbf{W}$ denotes the adjacent matrix representing the topology of $\mathcal{G}$, then $W_{ij} = 1$ if there is an edge connection between vertices v_i and v_j, otherwise W_{ij} = 0. Each vertex has a feature or signal denoted as $\mathbf{x} \in \mathbf{R}^d$, and d is the number of feature. The features of all vertices in graph are summarized by $\mathbf{X} \in \mathbf{R}^{n \times d}$.

Laplacian matrix of combinational definition is $\mathbf{L} = \mathbf{D} - \mathbf{W} \in \mathbf{R}^{n \times n}$, where $\mathbf{D} \in \mathbf{R}^{n \times n}$ is the degree matrix with $D_{ij} = \sum_j W_{ij}$, and the normalized definition is $\mathbf{L} = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{I}_n$ is identify matrix. As Laplacian matrix is positive semidefinite, there exists a matrix denoted as $\mathbf{U}$ which is composed of a set of orthogonal eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \cdots \mathbf{u}_n\}$ satisfying $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{\Lambda}$ is a diagonal matrix consisting of eigenvalues $\{\lambda_1, \lambda_2, \lambda_3 \cdots \lambda_n\}$.

In Fourier domain, the eigenvalue represents a specific frequency. The eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \cdots \mathbf{u}_n\}$ are Fourier basis. Then, the graph Fourier transform can be formulated as

$$\hat{\mathbf{X}} = \mathbf{U}^T\mathbf{X} \tag{1}$$

and the inverse transformation is given by

$$\mathbf{X} = \mathbf{U}\hat{\mathbf{X}} \tag{2}$$

In the Fourier domain, the filtering operation of graph is given as

$$\hat{\mathbf{X}}_{out} = \hat{h}(\mathbf{\Lambda})\hat{\mathbf{X}} \tag{3}$$

where $\hat{h}(\cdot)$ is the spectral operator. Equivalently, we transform it to the time domain as

$$\begin{aligned}\mathbf{X}_{out} &= \mathbf{U}\hat{h}(\mathbf{\Lambda})\hat{\mathbf{X}} \\ &= \mathbf{U}\hat{h}(\mathbf{\Lambda})\mathbf{U}^T\mathbf{X} \\ &= \mathbf{U}\begin{pmatrix}\hat{h}(\lambda_1) & & & \\ & \hat{h}(\lambda_2) & & \\ & & \ddots & \\ & & & \hat{h}(\lambda_n)\end{pmatrix}\mathbf{U}^T\mathbf{X} \\ &= \hat{h}(\mathbf{L})\mathbf{X}\end{aligned} \tag{4}$$

The spectral filtering is parameterized as

$$\begin{aligned}\hat{h}_\theta(\mathbf{\Lambda}) &= \begin{pmatrix}\hat{h}_\theta(\lambda_1) & & & \\ & \hat{h}_\theta(\lambda_2) & & \\ & & \ddots & \\ & & & \hat{h}_\theta(\lambda_n)\end{pmatrix} \\ &= \sum_{k=0}\theta_k\mathbf{\Lambda}^k\end{aligned} \tag{5}$$

In order to reduce the complexity of learning, Chebyshev are used to approximate as

$$\hat{h}_{\theta',k}(\mathbf{\Lambda}) = \sum_{k=0}^{K}\theta_k' T_k(\mathbf{\Lambda}) \tag{6}$$

where θ_k' is the Chebyshev polynomial coefficient, $T_k(\mathbf{\Lambda})$ is the Chebyshev approximate polynomial, $T_0(\alpha) = 1$, $T_1(\alpha) = \alpha$ and the recursive formulation is $T_k(\alpha) = 2\alpha T_{k-1}(\alpha) - 2\alpha T_{k-2}(\alpha)$.

Then, the Chebyshev approximate local filter is finally given as

$$\mathbf{X}_{out} = \mathbf{U}\hat{h}_{\theta',k}(\mathbf{\Lambda})\mathbf{U}^T\mathbf{X} = \sum_{k=0}^{K}\theta_k' T_k(\tilde{\mathbf{L}})\mathbf{X} \tag{7}$$

where $\tilde{\mathbf{L}} = \frac{2\mathbf{L}}{\lambda_{max}} - \mathbf{I}_n$, $\mathbf{I}_n$ is identify matrix. In the following experiments, we set $\lambda_{max} = 2$. θ_k' is the Chebyshev polynomial coefficient that needs to learn.

In the training process, the cross-entropy loss function is employed for optimizing the parameters of graph CNNs.

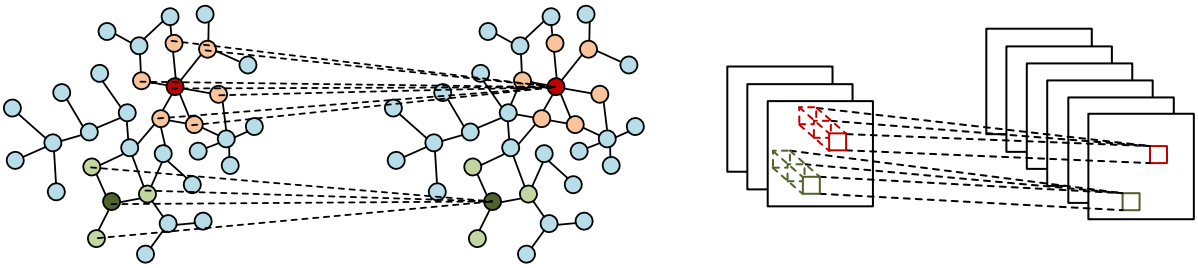
$$J(\theta) = -\sum_i I(y = C_i) ln\{\hat{y}_i(\theta)\} \tag{8}$$

Samples can be divided into C classes and $i \in C$. y is the label of the sample. $I(y = C_i)$ is the indicative function, $I(y = C_i) = 1$ when $y = C_i$, otherwise $I(y = C_i) = 0$. The back-propagation process can be given as

$$\frac{\partial J(\theta)}{\partial \theta} = -\sum_i I(y = C_i)\frac{1}{\hat{y}_i(\theta)}\frac{\partial \hat{y}_i(\theta)}{\partial \theta} \tag{9}$$

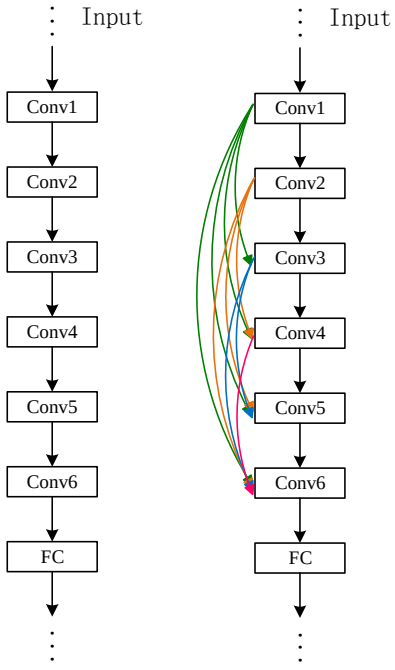
IV. Transforming Classic Network Structures to Graph CNN

In this section, we transform these classic network structures (including ResNet [8], Inception [10] and DenseNet [7]) to graph CNN and show the structures of three different networks. We will detailedly introduce four graph CNNs with different network structures, i.e., plain Graph CNN, G_ResNet, G_Inception, G_DenseNet.



(a) (b)

Fig. 1. Different convolution operations on graph (left) and gridded image (right) samples. (a) shows the 1-localized convolution on graph data, where the red node has 6 adjacencies, and the green node has 3 adjacencies. In the process of convolution, they use the features of the 6 and 3 nodes respectively, in addition to the feature of their own nodes. (b) shows the convolution operation on the regular gridded data (e.g., images and videos). As long as a fixed-size convolution kernel is given, the convolution kernel will slid from left-to-right and top-to-bottom, all vertices can be then convolved once.



(a) Our baseline (b) G_Densenet

Fig. 2. Graph CNN architectures of the baseline and G_Densenet.

A. Plain Graph CNN

The plain baseline of graph CNN are mainly inspired by the philosophy of VGG nets [6], as illustrated in Fig. 2(a). We stack 6 convolution layers, each of which is followed with a batch normalization and rectified linear layer. Different from the gridded images/videos, on which local convolution kernels can be defined as multiple lattices with various receptive fields, graph CNN can adopt a K-localized convolution for every vertex in a graph [19]. i.e. K hops from the central vertex. The number of channels in the first three convolution layers is 32, and 64 channels have been set for the last three convolution layers. According to Eq. 7, in the forward propagation of Graph CNN, we denote $\bar{\mathbf{X}}_k = T_k(\tilde{\mathbf{L}})\mathbf{X}$, and the k-th iteration $\bar{\mathbf{X}}_k = 2\tilde{\mathbf{L}}\bar{\mathbf{X}}_{k-1} - \bar{\mathbf{X}}_{k-2}$ with $\bar{\mathbf{X}}_0 = \mathbf{X}$ and $\bar{\mathbf{X}}_1 = \tilde{\mathbf{L}}\mathbf{X}$, where $\mathbf{X} \in \mathbf{R}^{n\times d}$ is the initial feature map of graph data. For the K-localized convolution, $\bar{\mathbf{X}}_K$ is the result of K-th iteration. Here the receptive field K of all convolution layers is simply set to 6. The plain Graph CNN ends with a fully-connected layer with softmax, which dimension equals to the number of classes in the graph datasets.

B. Dense Graph CNN

DenseNet combines features by concatenating them instead of combining features through summation. DenseNet improve flow of information and gradients throughout the network, which makes network easy to train. Based on the plain Graph CNN, we construct Dense graph CNN (G_DenseNet) by connecting each layer to every other layer in a feed-forward fashion, which is shown in Fig. 2(b). The input of each convolutional layer is the output of all previous convolutions, and the output of each convolutional layer must be used as the input of following convolutions. As the number of feature maps per layer gradually increases, the size of $\mathbf{\Theta}_i$ also increases from layer to layer. We can formulate Dense graph CNN as

$$\begin{aligned}
\mathbf{Y}_0 &= \mathcal{F}_{\text{dens}}(\bar{\mathbf{X}}_K, \mathbf{\Theta}_0)\\
\mathbf{Y}_1 &= \mathcal{F}_{\text{dens}}(\bar{\mathbf{X}}_K||\bar{\mathbf{Y}}_{0K}, \mathbf{\Theta}_1)\\
\mathbf{Y}_2 &= \mathcal{F}_{\text{dens}}(\bar{\mathbf{X}}_K||\bar{\mathbf{Y}}_{0K}||\bar{\mathbf{Y}}_{1K}, \mathbf{\Theta}_2)\\
&\cdots\\
\mathbf{Y}_l &= \mathcal{F}_{\text{dens}}(\bar{\mathbf{X}}_K||\bar{\mathbf{Y}}_{0K}||\cdots||\bar{\mathbf{Y}}_{l-1K}, \mathbf{\Theta}_l)
\end{aligned} \tag{10}$$

where $\mathbf{Y}_0, \mathbf{Y}_1, \cdots, \mathbf{Y}_l$ denote the output of different convolution layers, $\bar{\mathbf{X}}_K$, $\bar{\mathbf{Y}}_{0K}$, $\cdots$, $\bar{\mathbf{Y}}_{l-1K}$ correspond to the K-localized convolution of input graph signals. Based on the feature information of all preceding layers, the function $\mathcal{F}_{\text{dnes}}(\cdot)$ represents the dense operation to be learned by convolution layers.

C. Residual Graph CNN

Based on the above plain network, similar to the ResNet [8], we insert shortcuts connections which turn the plain Graph CNN into its residual version. The residual graph CNN is built by stacking two residual graph blocks, each of which consists of the residual part and identity mapping. The residual part consists of three convolutions, each of which is followed by a batch normalization and a rectified linear layer, and the identity mapping has a linear projection to match the same dimensions. The residual graph block is formulated as

$$\mathbf{Y} = \mathcal{F}_{\text{res}}(\bar{\mathbf{X}}_K, \{\mathbf{\Theta}_0, \mathbf{\Theta}_1, \mathbf{\Theta}_2\}) + \bar{\mathbf{X}}_K\mathbf{\Theta}_s, \tag{11}$$

where $\mathbf{\Theta}_0$, $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$ are the parameters of three different convolution layers, respectively. Θ_s is a linear projection matrix of identity mapping. The function $\mathcal{F}_{\text{res}}(\bar{\mathbf{X}}_K, \{\mathbf{\Theta}_0, \mathbf{\Theta}_1, \mathbf{\Theta}_2\})$ represents the residual mapping to be learned by three convolution layers. Each convolution layer is followed by a batch normalization (BN) layer and ReLU activation function. Every convolution of the first residual graph block is 32 channels, the second is 64 channels. The receptive field is set to 6 (i.e., K=6). For simplicity, the residual graph CNN is named as G_ResNet, and the corresponding architecture is illustrated in Fig.3(a).

D. Inception Graph CNN

For the Inception graph CNN named as G_Inception, we stack two Inception graph blocks that both followed by one convolution layer 3(b). Each Inception graph block is composed of four tributaries. Each tributary of the first three tributaries contains two convolutions and the last tributary contains one convolution. According to Eq. 7, we choose convolution layers with different receptive fields at different tributaries for better capturing various information of graph data. For the j-th convolution of the i-th tributary, $\mathbf{\Theta}_{ij}$ represents the corresponding parameter matrix. The Inception graph block can be given as

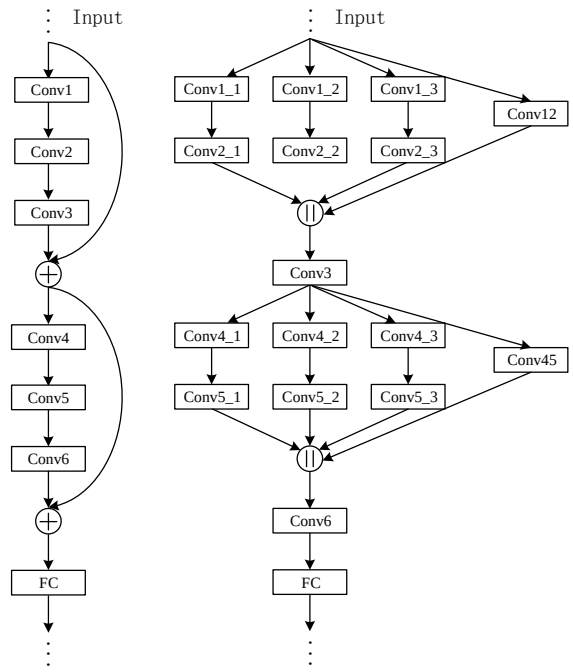
$$\begin{aligned}\mathbf{Y} =& \mathcal{F}_{\text{inc}}(\bar{\mathbf{X}}_{K1}, \{\mathbf{\Theta}_{00}, \mathbf{\Theta}_{01}\}) \\ &|| \ \mathcal{F}_{\text{inc}}(\bar{\mathbf{X}}_{K2}, \{\mathbf{\Theta}_{10}, \mathbf{\Theta}_{11}\}) \\ &|| \ \mathcal{F}_{\text{inc}}(\bar{\mathbf{X}}_{K3}, \{\mathbf{\Theta}_{20}, \mathbf{\Theta}_{21}\}) \\ &|| \ \mathcal{F}_{\text{inc}}(\bar{\mathbf{X}}_{K4}, \{\mathbf{\Theta}_{30}\}),\end{aligned} \tag{12}$$

where the symbol "||" represents concatenation [30] among convolution layers, Each convolution layer is also followed by a batch normalization (BN) layer and ReLU activation function. The function $\mathcal{F}_{\text{inc}}(\cdot)$ denotes how to represent the graph information with a prescribed receptive field in each path of inception block, and receptive fields are not exactly the same in different tributaries.

In the graph Inception block, the receptive fields of convolution layers in the first three paths can be set as $K_1 = 3$, $K_2 = 6$ and $K_3 = 9$, respectively. And the rest path only includes one convolution layer, and we choose $K_4 = 6$ as the size of receptive field. Such a designed Inceptive structure can take multi-scale information of graph signal into account. The convolution layers in the first Inception graph block is with 32 channels, the second is with 64 channels. The receptive field is set to 6 of both "Conv3" with 32 channels and "Conv6" with 64 channels followed by two graph Inception block.

V. Experiments

In the section, we evaluate the performance of four different graph CNN structures on several public benchmark datasets including bioinformatics and social network datasets. They are all undirected graphs and their global properties are summarized in Table I [31]. We first introduce the datasets and experimental setups, and then report and analyze the experimental results, after which a further discussion will be held about different parameters of Graph CNN.



(a) G_ResNet (b) G_Inception

Fig. 3. Graph CNN architectures of G_ResNet and G_Inception.

A. Datasets

Bioinformatics datasets. MUTAG [32] is a nitro compounds dataset including 188 samples and divided into 2 classes. PTC [33] consists of compounds labeled according to carcinogenicity on rodents with 19 vertex labels. NCI109 [34] is a balanced dataset of chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell, and they contain 4110 and 4127 chemical compounds, respectively. ENZYMES [35] is a dataset of 600 protein tertiary structures obtained from the BRENDA enzyme database. The ENZYMES dataset contains 6 enzymes.

Social network datasets. These social network datasets come from [36]. We use the number of neighbors of each node as the label of the node. COLLAB is a scientific collaboration dataset containing ego-networks of different researchers from three subfields of Physics. The task can then determine whether the ego-collaboration network belongs to any of three classes: High Energy Physics, Condense Mater Physics and Astro Physics. IMDB-BINARY and IMDB-MULTI are movie collaboration datasets. Each graph represents a movie. Each vertex represents an actor that appears in the movie. IMDB-BINARY are constructed from Action and Romance genres. IMDB-MULTI contain three classes: Comedy, Romance and Sci-Fi. The task is to predict which genre a graph belongs to.

B. Experimental Setups

We train four different graph CNNs, which structures has been described in Section IV. Each dataset is divided into 10 groups, of which nine are used to train the network and the remaining one is used for testing. We carry out 10-fold cross validation, and its average results can be used as the final accuracy rates. The prediction accuracy is expressed in form

TABLE I
SUMMARY OF GRAPH DATASETS USED IN OUR EXPERIMENTS.

Dataset	Num graphs	Classes	Node labels	Avg nodes	Avg edges
MUTAG [32]	188	2	7	17.93	19.79
PTC [33]	344	2	19	14.29	14.69
NCI109 [34]	4127	2	38	29.68	32.13
ENZYMES [35]	600	6	3	32.63	62.14
COLLAB [36]	5000	3	-	74.49	2457.78
IMDB-BINARY [36]	1000	2	-	19.77	96.53
IMDB-MULTI [36]	1500	3	-	13.0	65.94

of "standard $\pm$ deviation" in graph classification benchmark datasets. For each cross-validation, we train 300 epochs using Momentum Optimizer and the learning rate is set to 0.01. In addition, the momentum is 0.9 and the decay rate is with 0.95. We use cross entropy as loss function. For the baseline Graph CNN, G_ResNet and G_DenseNet, the receptive field of each convolution layer is set to 6. For G_Inception, it can merge multi-scale information of graph signals. Therefore, in the G_ Inception block, for each of the two convolutions, we set 3, 6 and 9 as the receptive fields, respectively. And for the rest only one convolution, We choose the middle 6 as its receptive field. To prevent overfitting, we add a dropout layer followed the fully connected layer, which can randomly discard half of the neurons.

C. Results and comparisons

We compare the proposed Graph CNNs (i.e., baseline Graph CNN, G_ResNet, G_DenseNet and G_Inception) with several state-of-the-art approaches on seven graph datasets, such as random walk kernel (RW) [37], the graphlet kernels (GK) [38], the Weisfeiler-Lehman subtree kernels (WL) [39], Feature-Based (FB) [17], Deep Graphlet (DGK) and DWL [40], the convolutional neural network (PSCN) [12], the shift aggregate extract network (SAEN) [2] and the dynamics based features (DyF) [31]. The performance of four Graph CNNs and comparisons with several state-of-the-art methods are shown in table II. Overall, the experimental performance of the our four Graph CNN structures is superior to other existing methods. Comparing with kernel based method [37], [38], [39], [40], feature based method [17], [31], convolution neural network (PSCN) [12] and shift aggregate extract network (SAEN) [2], all our graph CNN structures with spectral filtering method achieve the state-of-the-art performance on all datasets and obtain a significantly improvement on MUTAG, PTC, ENZYMES and IMDB-BINARY datasets in contrast to the second best performance. G_ResNet achieves state-of-the-art on two social network datasets: 79.90% vs 72.87% [31] on IMDB-BINARY and 54.43% vs 50.55% [39] on IMDB-MULTI. G_Inception can significantly outperform these state-of-the-art algorithms on two datasets: 95.00% vs 92.63% [12] on MUTAG and 67.50% vs 53.43% [40] on ENZYMES. G_DenseNet shows much improvement than other methods on two bioinformatic datasets: 73.24% vs 60.00% [12] on PTC and 80.66% vs 80.32% [40] on NCI109 and 83.16% vs 80.61% [31] on COLLAB. Our graph CNNs are able to render very impressive results. The performance pf graph recognition can be further improved to some extent when we deepen these three graph CNN and the results are reported in the following discussion.

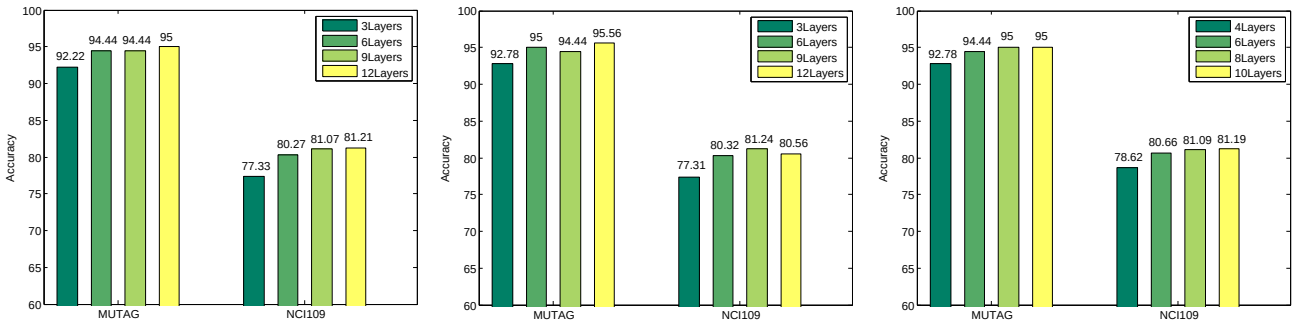
When comparing the performance of the baseline Graph CNN, G_ResNet, G_Inception and G_DenseNet, different Graph CNN models have the certain advantages on various types of graph datasets. The structures of ResNet, Inception and DenseNet were proposed initially to boost the performance of deep convolution networks from different aspects, such as residual learning, considering information in multiple receptive fields, densely employing multi-level representations. For better show how to transform these CNN structures to Graph CNNs, the proposed G_ResNet, G_Inception and G_DenseNet can still outperform our baseline network, which is a simple 6-layer graph CNN framework. For example, our G_Inception model on the ENZYMES dataset can significantly outperform the baseline network, 67.50% vs 64.83%. This indicates that our G_Inception model can introduce greater data diversity in multiple receptive fields and improve the network performance of graph data. We achieve much better performance with the G_DenseNet framework then the baseline network on PTC dataset, e.g., 73.24% vs 71.76%. It demonstrates that the G_DenseNet perform very well on the graph recognition problem by simultaneously considering different level information. The accuracies of G_ResNet have been improved on IMDB-BINARY and IMDB-MULTI datasets, which also show its capability by transforming the ResNet structure to Graph CNN.

D. Discussion

Deeper graph CNN: Here we explore how deeper graph CNNs impact the performance of graph recognition. For G_ResNet and G_Inception networks, the number of convolution layers can be set to: 3 layers, 6 layers, 9 layers and 12 layers, respectively. Considering that each previous result in G_DenseNet will be used as the input to the convolution layer behind. In this way, the number of parameters in the last few layers is huge, and the amount of calculations also increases dramatically. Therefore the number of convolution layers is set to: 4 layers, 6 layers, 8 layers and 10 layers for G_DenseNet. We choose a small dataset MUTAG with 188 nodes and a large dataset NCI109 with 4127 nodes described above. All experiments of exploring deeper graph CNN are

TABLE II
COMPARISON OF GRAPH RECOGNITION PERFORMANCES WITH DIFFERENT GRAPH CNN MODELS AND SEVERAL STATE-OF-THE-ARTS ON GRAPH DATASETS.

Dataset	MUTAG	PTC	NCI109	ENZYMES	COLLAB	IMDB-B	IMDB-M
RW [37]	83.72 ± 1.50	57.85 ± 1.30	49.75 ± 0.60	24.16 ± 1.64	69.01 ± 0.09	64.54 ± 1.22	34.54 ± 0.76
GK [38]	81.66 ± 2.11	57.26 ± 1.41	62.60 ± 0.19	26.61 ± 0.99	72.84 ± 0.28	65.87 ± 0.98	43.89 ± 0.38
WL [39]	80.72 ± 3.00	56.97 ± 2.01	80.22 ± 0.34	53.15 ± 1.14	77.79 ± 0.19	72.86 ± 0.76	50.55 ± 0.55
FB [17]	84.66 ± 2.01	55.58 ± 2.30	62.43 ± 1.13	29.00 ± 1.16	76.35 ± 1.64	72.02 ± 4.71	47.34 ± 3.56
DGK [40]	82.66 ± 1.45	57.32 ± 1.13	62.69 ± 0.23	27.08 ± 0.79	73.09 ± 0.25	66.96 ± 0.56	44.55 ± 0.52
DWL [40]	82.94 ± 2.68	59.17 ± 1.56	80.32 ± 0.33	53.43 ± 0.91	-	-	-
PSCN [12]	92.63 ± 4.21	60.00 ± 4.82	-	-	72.60 ± 2.15	71.00 ± 2.29	45.23 ± 2.84
SAEN [2]	84.99 ± 1.82	57.04 ± 1.30	-	-	75.63 ± 0.31	71.26 ± 0.74	49.11 ± 0.64
DyF [31]	88.00 ± 2.37	57.15 ± 1.47	66.72 ± 0.20	33.21 ± 1.20	80.61 ± 1.60	72.87 ± 4.05	48.12 ± 3.56
Our baseline	93.89 ± 6.31	71.76 ± 7.58	80.51 ± 2.67	64.83 ± 5.45	82.96 ± 0.86	79.70 ± 3.66	54.40 ± 4.88
G_ResNet	94.44 ± 5.56	73.24 ± 8.05	80.27 ± 2.56	66.83 ± 7.47	82.64 ± 0.99	$\mathbf{79.90 \pm 3.96}$	$\mathbf{54.53 \pm 4.25}$
G_Inception	$\mathbf{95.00 \pm 4.61}$	72.94 ± 6.28	80.32 ± 1.73	$\mathbf{67.50 \pm 5.54}$	82.58 ± 1.28	78.40 ± 3.72	54.53 ± 4.71
G_DenseNet	94.44 ± 4.30	$\mathbf{73.24 \pm 6.64}$	$\mathbf{80.66 \pm 2.49}$	66.83 ± 4.86	$\mathbf{83.16 \pm 1.00}$	79.20 ± 4.19	54.40 ± 4.70



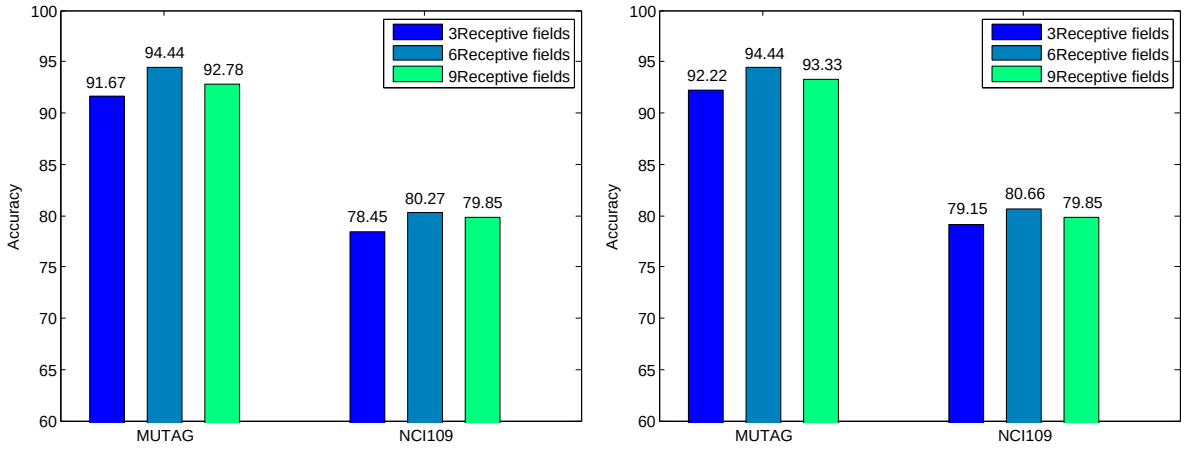
(a) G_ResNet (b) G_Inception (c) G_Densenet

Fig. 4. Performance comparison of G_ResNet, G_Inception, G_DenseNet with different number of convolution layers. Experiments are conducted on two datasets : MUTAG (a small dataset with 188 nodes) and NCI109 (a large dataset with 4127 nodes).

performed on these two datasets, and the experimental results are reported in Fig. 4.

For the G_ResNet model, we stack 3-layer convolution for each residual graph block. In order to build deeper G_ResNet, we stack 6-layer, 9-layer, 12-layer convolution by 2, 3, 4 residual graph block, respectively. The Fig. 4(a) shows the comparisons of different G_ResNet with different number of network layers on MUTAG and NCI109 datasets. For MUTAG dataset, the results of G_ResNet models, which are with 3-layer convolution, the 6-layer convolution, the 9-layer convolution, and the 12-layer convolution, can achieve 92.22%, 94.44%, 94.44% and 95%, respectively. For NCI109 dataset, the performance of different G_ResNet models are 77.33%, 80.27%, 81.07% and 81.21%, respectively. In the G_Inception network, we further observe the changes of classification performance by stacking different number of inception blocks. Stacking 3-layer convolution means that there is only one Inception graph block in the G_Inception model. Stacking 6-layer, 9-layer, 12-layer convolution means that there are 2,3,4 Inception graph block in G_Inception, respectively. As can be illustrated in the Fig. 4(b), the performance of 3-layer G_Inception model is 92.78% on MUTAG dataset. The G_Inception with 12-layer convolution is better than with the 3-layer convolution, e.g., 95.56% vs 92.78%. For NCI109 dataset, the results with different deeper G_Inception models are 77.31%, 80.32%, 81.24% and 80.56%, respectively. The G_Inception network with 9-layer convolution achieves the highest performance 81.24% and improves the average accuracy by 3.93% than the G_Inception model with the 3-layer convolution. Similar to comparisons of G_ResNet and G_Inception, we explore the performance of G_DenseNet with 4-layer, 6-layer, 8-layer and 10 layer network, respectively. As reported in Fig. 4(c), the G_Inception network with the 12-layer convolution can outperform the G_Inception with 3-layer convolution by 2.22%. On the NCI109 dataset, The G_Inception with 12-layer convolution is better than with the 3-layer convolution, e.g., 81.19% vs 78.62%. We can observe that as the number of convolution layers increases, the classification accuracy increases. Another reasonable explanation is that deeper Graph CNN models abstract higher-level presentations that will also help improve the performance of graph recognition.

Extend the receptive field of convolution layer. For G_ResNet and G_DenseNet, we also explore the performance of different receptive fields. The structure of G_Inception has been combined with a variety of receptive fields information, so we don't conduct this discussion on it. In different G_Inception networks, the size of receptive field is set to: 3, 6 and 9, respectively. We think that the selection



(a) G_ResNet (b) G_Densenet

Fig. 5. Performance comparison of G_ResNet and G_Densenet with different receptive fields on two datasets : MUTAG (a small dataset with 188 nodes) and NCI109 (a large dataset with 4127 nodes). The size of receptive field can be set to 3, 6 and 9, respectively.

of receptive fields, which is important for graph CNN, can consider different local structural information. The experiment is performed on the same two datasets: MUTAG and NCI109, and the comparisons of different receptive fields are reported in Fig. 5. With the 3, 6 and 9 receptive fields of G_ResNet, we can observe accuracies of 91.67%, 94.44%, 92.78% on MUTAG dataset, and 78.45%, 80.27%, 79.85% on NCI109 dataset (see Fig. 5(a)). When the receptive field is set to 6, the performance of G_ResNet is highest on both datasets. As can be reported in Fig. 5(b), we can observe that different G_DenseNet models with 3, 6, 9 receptive fields can gain 92.22%, 94.44%, 93.33% on MUTAG dataset, and 79.15%, 80.66%, 79.85% on NCI109 dataset. The performance is also highest when the receptive field is set to 6 in the G_DenseNet model. It demonstrates that extracted representations of graph signals are overly complex and too small to represent useful features, when the size of the receptive field is too large. If the receptive field is too small, local structural features may not be extracted, and the performance of graph CNN will be reduced.

VI. Conclusions and Future Work

In this work, we have given a comprehensive analysis of when work matters by transforming classical network structures to graph CNN, particularly for the basic graph recognition task. Inspired by the basic ideas of ResNet, DenseNet and Inception network, we have constructed different Graph CNN architectures, including the plain graph CNN, G_ResNet, G_Inception, G_DenseNet. By constructing the G_Inception network, we focus on considering different receptive fields of graph signals in the process of convolution, while G_DenseNet is responsible for capturing different-level representations of graph CNN. The G_ResNet is benefit to constructing deeper graph networks with the help of residual learning. Extensive experimental results clearly demonstrated that effective of the proposed G_ResNet, G_Inception, G_DenseNet models for the problem of graph recognition. In the future, we will further extent the Graph CNN architectures for generic understanding tasks, e.g., image restoration, social network analysis, visual understanding, etc.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant 61073094 and U1233119). This research was partly supported under Australian Research Council Discovery Projects funding scheme (project DP140102270).

References

[1] H. Wang, J. Wu, X. Zhu, and C. Zhang, "Time-variant graph classification," *arXiv preprint arXiv:1609.04350*, 2016. 1

[2] F. Orsini, D. Baracchi, and P. Frasconi, "Shift aggregate extract networks," *arXiv preprint arXiv:1703.05537*, 2017. 1, 6, 7

[3] K. M. Borgwardt, H.-P. Kriegel, S. Vishwanathan, and N. N. Schraudolph, "Graph kernels for disease outcome prediction from protein-protein interaction networks." *Pacific Symposium on Biocomputing Pacific Symposium on Biocomputing*, pp. 4–15, 2007. 1

[4] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Technical Report 1999-66, 1999. 1

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105. 1, 2

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1, 2, 4

[7] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269. 1, 2, 3

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. 1, 2, 3, 4

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324. 1, 2

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9. 1, 2, 3

[11] F. R. Chung, *Spectral graph theory*. American Mathematical society, 1997, no. 92. 1, 2

[12] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International Conference on Machine Learning*, 2016, pp. 2014–2023. 1, 2, 6, 7

[13] B. Wu, Y. Liu, B. Lang, and L. Huang, "DGCNN: disordered graph convolutional neural network based on the gaussian mixture model," *arXiv preprint arXiv:1712.03563*, 2017. 1, 2

[14] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 2, 2017, p. 3. 1

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017. 1, 2

[16] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013. 1, 3

[17] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *International Conference on Learning Representations*, 2013. 2, 3, 6, 7

[18] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015. 2, 3

[19] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852. 2, 3, 4

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations*, 2016. 2, 3

[21] A. T. Lopes, E. de Aguiar, A. F. De Souza, and T. Oliveira-Santos, "Facial expression recognition with convolutional neural networks: coping with few data and the training sample order," *Pattern Recognition*, vol. 61, pp. 610–628, 2017. 2

[22] D. T. Nguyen, W. Li, and P. O. Ogunbona, "Human detection from images and videos: A survey," *Pattern Recognition*, vol. 51, pp. 148–175, 2016. 2

[23] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576. 2

[24] M. Babaee, D. T. Dinh, and G. Rigoll, "A deep convolutional neural network for video sequence background subtraction," *Pattern Recognition*, vol. 76, pp. 635–649, 2018. 2

[25] M. Lin, Q. Chen, and S. Yan, "Network in network," *International Conference on Learning Representations*, 2014. 2

[26] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2015, pp. 5353–5360. 2

[27] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015. 2

[28] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *arXiv preprint arXiv:1612.07659*, 2016. 2

[29] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," *arXiv preprint arXiv:1801.03226*, 2018. 3

[30] W. Wang, X. Li, J. Yang, and T. Lu, "Mixed link networks," *arXiv preprint arXiv:1802.01808*, 2018. 5

[31] L. G. Gomez, B. Chiem, and J.-C. Delvenne, "Dynamics based features for graph classification," *arXiv preprint arXiv:1705.10817*, 2017. 5, 6, 7

[32] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991. 5, 6

[33] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma, "Statistical evaluation of the predictive toxicology challenge 2000–2001," *Bioinformatics*, vol. 19, no. 10, pp. 1183–1193, 2003. 5, 6

[34] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 347–375, 2008. 5, 6

[35] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005. 5, 6

[36] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1365–1374. 5, 6

[37] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines*. Springer, 2003, pp. 129–143. 6, 7

[38] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial Intelligence and Statistics*, 2009, pp. 488–495. 6, 7

[39] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. 23, pp. 2539–2561, 2011. 6, 7

[40] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1365–1374. 6, 7