Motif-matching based Subgraph-level Attentional Convolutional Network for Graph Classification

Hao Peng,[1,2,3] Jianxin Li,[1,2] Qiran Gong,[4] Yuanxing Ning,[1,2] Senzhang Wang,[5] Lifang He [6]
[1]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China
[2]State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China
[3]Key Laboratory of Aerospace Network Security, Ministry of industry and information technology,
School of Cyberspace Science and Technology, Beihang Universty, Beijing 100191, China
[4]Department of Computer Science, Brown University, Providence, RI, USA
[5]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.
[6]Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, USA
{penghao,lijx,ningyx}@act.buaa.edu.cn, qiran_gong@brown.edu , szwang@nuaa.edu.cn , lih319@lehigh.edu

Abstract

Graph classification is critically important to many real-world applications that are associated with graph data such as chemical drug analysis and social network mining. Traditional methods usually require feature engineering to extract the graph features that can help discriminate the graphs of different classes. Although recently deep learning based graph embedding approaches are proposed to automatically learn graph features, they mostly use a few vertex arrangements extracted from the graph for feature learning, which may lose some structural information. In this work, we present a novel motif-based attentional graph convolution neural network for graph classification, which can learn more discriminative and richer graph features. Specifically, a motif-matching guided subgraph normalization method is developed to better preserve the spatial information. A novel subgraph-level self-attention network is also proposed to capture the different impacts or weights of different subgraphs. Experimental results on both bioinformatics and social network datasets show that the proposed models significantly improve graph classification performance over both traditional graph kernel methods and recent deep learning approaches.

Introduction

Graph classification, which aims to identify the class labels of graphs in a dataset, is critically important to many real-world applications in diverse domains. Data from molecular chemistry (Benkö, Flamm, and Stadler 2003), bioinformatics drug discovery (Gonzalez-Diaz et al. 2007), social network (Backstrom and Leskovec 2011), text classification (Peng et al. 2018), malware detection (Anderson et al. 2011), etc., can all be represented as labeled graphs with relationships and interdependencies between objects. In chemistry and bioinformatics drug analysis, for instance, each chemical compound can be represented as a graph where nodes correspond to atoms, and edges signify the presence of chemical bonds between atoms. The task then is to predict the class label of each graph, for instance, the anti-cancer activity, mutagenic or toxicity of a chemical compound.

To solve the graph classification problem, one usually extracts some graph features that help discriminate the graphs of different classes. Traditional technologies include random-walks, subgraphs or sub-tree patterns based graph kernel methods (Vishwanathan et al. 2010; Shervashidze et al. 2011). In general, graphs that share many common graphlets are considered similar. The graph kernel based methods measure the similarity between two graphs with kernel functions corresponding to the inner products of the extracted features (Vishwanathan et al. 2010; Yanardag and Vishwanathan 2015). With the recent success of deep learning techniques, the focus of graph classification techniques has shifted from the graph kernel functions to the spatial-based graph convolutional neural network (GCNN) (Wu et al. 2019). One pioneering spatial-based GCNN model is PSCN (Niepert, Ahmed, and Kutzkov 2016). It utilizes standard convolutional neural network by converting graph-structured data into grid-structured data with multiple sorting functions. However, PSCN only samples a few of vertex arrangements as the grid-structured data to represent a graph, which loses some structural information such as the subgraph structures. In addition, the design of the sorting functions requires strong prior knowledge, which is very difficult in practice.

It is non-trivial to use deep learning techniques for graph classification due to the following two major challenges. First, it is very challenging for deep learning models such as CNNs to handle the complex graph data. Graph data are in the non-Euclidean domain, and each graph has a variable size of unordered nodes and each node in a graph has a different number of neighbors. Therefore, some important operations (e.g., convolutions, recurrences) that are easy to compute for image data, are very hard to conduct for the graph data. Although some recent works including PSCN (Niepert, Ahmed, and Kutzkov 2016), DGCNN (Zhang et al. 2018), NEST (Carl et al. 2018) and DIFFPOOL (Ying et al. 2018) proposed new graph preprocessing paradigms or graph convolutional networks, they cannot fully capture the aggregated features from neighbors and nodes by using a variety of pooling operators. Second, existing deep learning based graph classification models (Zhou et al. 2018; Wu et al. 2019) lack of sufficient study on the diverse impacts of different nodes, graphlets or subgraphs. They consider the impacts between two objects or feature maps generated by convolution kernels are equally important. The existing graph attention models (Veličković et al. 2018; Lee et al. 2018) can only generate node-level embedding. Thus they are difficult to be applied to arbitrary graph classifications due to the required strong prior knowledge to design the attention model. Although GAM (Lee, Rossi, and Kong 2018) is a node-level attentional RNN model which considers local connectivity among nodes, we argue that higher level such as subgraph-level attention is more interpretable in graph classification. Hence existing deep learning based graph classification models generally lack of sufficient interpretability.

To address the above challenges, we propose a Motif-based Attentional Graph Convolutional Neural Network model, named as MA-GCNN, for graph classification. We first propose a motif-matching based subgraph normalization method to better preserve spatial information by converting graph-structured data into a grid-structured data representation. Second, we design subgraph-independent convolutional neural networks to learn different-levels of features for each subgraph without pooling operators. Next, a novel subgraph-level self-attention network is also introduced in the propagation step to learn different impacts or weights of different subgraphs in a graph for classification. Here, we concatenate the feature maps of each channel of each subgraph, learned by different convolution kernels, into a corresponding vector, and then measure different impacts or weights among the vectors by the self-attention mechanism. By leveraging the motif-matching guided graph processing, two layers of subgraph-independent convolution kernels and subgraph level self-attentional network layer, we finally design a novel end-to-end graph classification framework, which is more interpretable via the weighted subgraphs. We conduct extensive experiments on both bioinformatics and social network datasets for graph classification. Compared with both traditional graph kernel based algorithms and recent deep learning approaches, our proposed models achieve significant performance improvement in classification accuracy on ten benchmark datasets. The code of this work is publicly available at *https://github.com/RingBDStack/MA-GCNNs*.

Related Work

Existing works for graph classification can be broadly categorized into traditional graph kernel based methods and graph convolutional neural networks based models.

A great deal of research works have focused on designing the suitable graph kernel functions for each graph dataset in terms of the task of classification. Popular methods include graphlets (Shervashidze et al. 2009), random walk and shortest path kernel (Borgwardt and Kriegel 2005), sub-tree kernel (Shervashidze et al. 2011), deep graph kernel (Yanardag and Vishwanathan 2015), graph invariant kernel (Orsini, Frasconi, and De Raedt 2015), multiscale laplacian graph kernel (Kondor and Pan 2016), and graph feature selection (Kong et al. 2013). In general, the graphlet kernel decomposes a graph into graphlets, Weisfeiler-Lehman kernel decomposes a graph into sub-trees, and shortest-path kernel decomposes a graph into shortest-paths. The decomposed sub-structures are then represented as a vector of frequencies where each vector element represents the occurrence time of a given sub-structure in the graph. Thus, the Euclidean space or some other domain-specific reproducing kernel Hilbert space is used to define the dot product between the vectors of frequencies. Kernel based models have demonstrated the ability to capture different levels of features in graph classification. However, although kernel based models can capture explicit and sub-structural similarities at different levels, they are not able to capture implicit similarities.

Recently, graph convolutional neural networks (GCNNs) are proposed and presented promising performance in graph classification. The original idea of defining graph convolution has been recognized as the problem of learning filter parameters that appear in the graph fourier transform in the form of a graph Laplacian (Bruna et al. 2014). (Kipf and Welling 2017) proposed a self-loop graph adjacency matrix and a propagation rule to compute and update the weights in each neural network layer. An optimized GCNNs model is proposed in (Defferrard, Bresson, and Vandergheynst 2016) by utilizing fast localized spectral filters and efficient pooling operations. Considering the weakness of traditional CNNs in spatial hierarchies and rotational invariance, Graph Capsule networks are proposed in (Verma and Zhang 2018; Xinyi and Chen 2019). (Lee, Rossi, and Kong 2018) proposed a RNN based node-level graph attention model GAM to process informative parts of a graph by adaptively visiting a sequence of important nodes. Due to the disadvantage of partial observability of the input graphs, GAM is not suitable for arbitrary graphs, and cannot achieve state-of-art performance. In chemical and bimolecular structures, nodes can represent basic chemical elements, groups or amino acids, while toxic or pharmacological components are often a combination of basic elements at high-level. However, existing attentional deep learning based graph classification models ignored the division of a large graph into subgraphs to improve the model interpretability for downstream applications. RNN autoencoder based graph representation methods adopt random walks, breadth-first search and shortest paths to generate node sequences to learn structural features (Aynaz and Tanya 2018). (Simonovsky and Komodakis 2017) introduced a edge-conditioned convolution operation on graph signal performed in the spatial domain where filter weights are conditioned on edge labels and are dynamically generated for each input sample. (Ying et al. 2018) proposed a differentiable graph pooling to generate hierarchical representations of graphs. (Ivanov and Burnaev 2018) proposed to use the distribution of anonymous walks as a network embedding, sampling walks in a graph to approximate actual distribution with a given confidence. DGCNN (Zhang et al. 2018) used a SortPooling layer which sorted graph vertices in a consistent order to make traditional neural networks trainable on the graphs. Different from the above methods capturing spatial structures through various pooling and convolution operations, our proposal preserves spatial structures by a grid-structured representation which preserves rich semantic information.

Motifs are high-order structures that are crucial in many domains such as bioinformatics, neuroscience and social

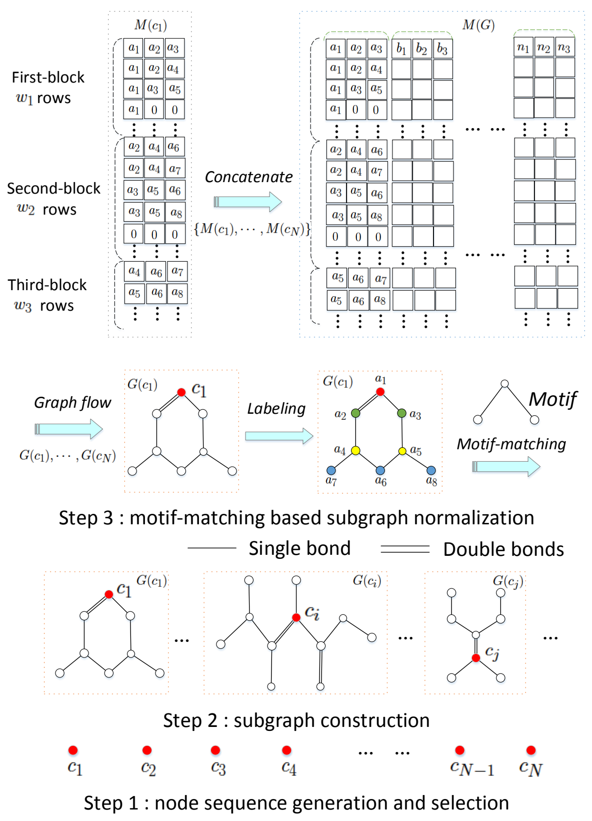


Figure 1: An illustration of motif-matching guided graph processing.

network analysis. Recent works have explored motifs in clustering (Benson, Gleich, and Leskovec 2016) and graph classification (Carl et al. 2018) tasks. However, existing methods only focus on applying motifs to filter structures of graphs for neural networks optimization, but do not fully exploit motifs to capture local stationary and spatial structures of a graph. The most relevant work to ours is PSCN (Niepert, Ahmed, and Kutzkov 2016) model. As a standardized process from graph to convolutional neural networks, PSCN contains node sequence selection, graph normalization and shallow convolution. Compared to our methods, PSCN ingores some structural information when converting the graph-structured data into grid-structured representation.

Motif-matching based Graph Processing

We first introduce how to transform the graph-structured data to a grid-structured representation. As shown in Figure 1, the Motif-matching based graph processing includes node sequence generation and selection, subgraph construction, and motif-matching based subgraph normalization.

We denote a graph as $G = (V, E)$, where V denotes the node set and $|V| = n$, E denotes the edge set and $|E| = m$. $d(v, u)$ denotes the length of the shortest path between two nodes v and u in G. Here, we treat the double, triple and higher bonds of the edges in bioinformatics as two, three and more single bonds in the graph. So, we can compute a *closeness centrality* $C_v = (n-1)/\sum_{u\in V, u\neq v} d(v, u)$ for each node v on G in parallel. Specifically, we use the two-hop paths as the motif structure, because two-hop paths have symmetry and are suitable for various matches. As the distributions of edges in graphs are usually unbalanced, some more complex motifs, such as triangles, may be rarely matched. In fact, dense subgraphs can be partitioned into multiple two-hop paths, and thus can be approximately reconstructed. The two-hop paths motif can be represented by an array which is suitable for convolution operations.

For the first step of node sequence generation and selection, we sort all nodes in a graph by their *closeness centrality* in descending order, and then select the top-N nodes as central nodes of the graph. The selected top-N nodes are considered as representative objects. As shown in Step 1 of Figure 1, the red nodes from c_1 to c_N represent the selected central nodes. If the number of nodes in the graph is less than N, the remaining central nodes will be padded with zeros. Only top-N nodes are selected to reduce the computational complexity of feature learning.

Next, we extract a subgraph $G(c_i)$ for each central node $c_i, i \in [1, N]$, as shown in Step 2 of Figure 1. Each subgraph is extracted in the order of *first*, *second* and *third* order neighbors based on breadth first search (BFS) and the *closeness centrality*. Here, we limit the number of nodes in the subgraph not larger than K. The idea of selecting N subgraphs on G and limiting the total number of nodes of each subgraph is to maximize the coverage of the original graph structure with N subgraphs. The design of the subgraph is similar to the receptive field of the sliding window in CNN models. For instance, we extract the subgraph $G(c_1)$ for the central node c_1.

Finally, we convert each subgraph $G(c_i)$ into a corresponding central matrix $M(c_i)$ by motif-matching. We first initialize the central matrix with zeros. For each node $\bar{u}$ in subgraph $G(c_i)$, we calculate the length of shortest path between node $\bar{u}$ to the central node c_i of the subgraph. Therefore, we can sort the nodes in the subgraph $G(c_i)$ based on their shortest distances and *closeness centrality*. Based on the sorted node index, we serialize all nodes in the subgraph to $\{a_1, a_2, a_3, \cdots, a_k\}$. For instance, we serialize the nodes in $G(c_1)$ to $\{a_1, a_2, \cdots, a_8\}$, as shown in Step 3 of Figure 1. Next, we fill the central matrix by the matched two-hop paths motif in parallel. Each row of the central matrix $M(c_i)$ is filled with matched nodes in the two-hop paths. As the number of nodes in each subgraph is small, we can also perform a fast motif matching algorithm (Sun et al. 2012) when there are a large number of subgraphs. Here, we keep the closest node to the central node in the first column (the red nodes in the first column of the central matrix $M(c_1)$ in Step 3 of Figure 1). Meanwhile, the central matrix $M(c_i)$ can be divided into three blocks according to the shortest distance from the matched nodes to the center node c_i. The first block contains the central node, such as the $a1$ in the first block of $M(c_1)$ shown in the Step 3. Similarly, the second and third blocks contain the two-hop and three-hop neighbor nodes of the center node, respectively.

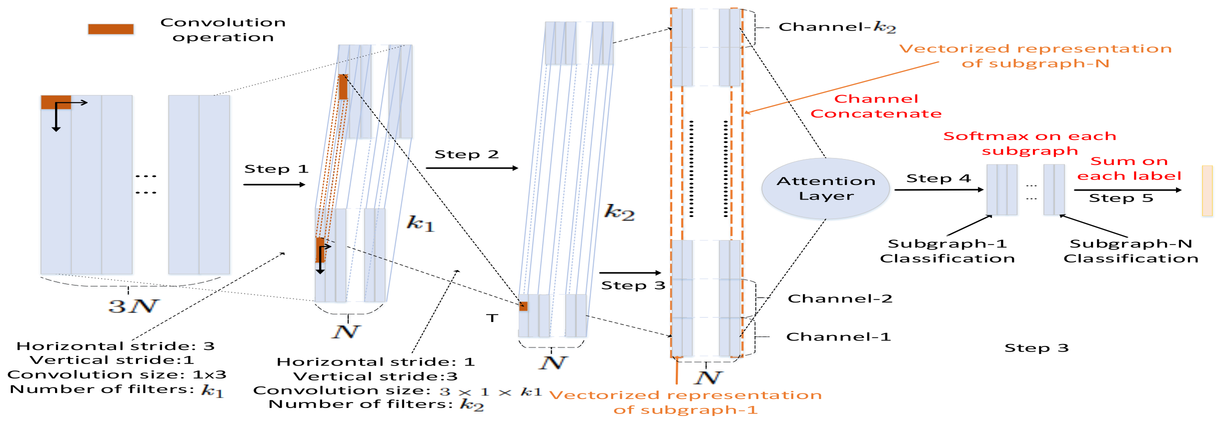


Figure 2: An illustration of the subgraph-level self-attention deep convolutional neural network.

In general, different hops can have different block sizes, such as $\{a_1, a_2, a_3\}$, $\{a_1, a_2, a_4\}$ and $\{a_1, a_3, a_5\}$ in the first block, $\{a_2, a_4, a_6\}$, $\{a_2, a_4, a_7\}$, $\{a_3, a_5, a_6\}$, $\{a_3, a_5, a_8\}$ in the second block and $\{a_4, a_6, a_7\}$, $\{a_5, a_7, a_8\}$ in the third block of $M(c_1)$. Considering the different scales of graphs, we fix the row numbers of the first, second and third blocks as w_1, w_2 and w_3, respectively. Note that we preserve all the nodes and their relationships in a subgraph to a corresponding center matrix according to the above rules without repetitive matching. Next, we concatenate the N central matrices $M(c_i), i \in [1, N]$ into a large combined matrix $M(G)$ following the sequence in Step 1, such as the the combination of $M(c_1)$, $M(c_2), \cdots, M(c_N)$ shown in Step 3 of Figure 1. Therefore, for a graph G, it can be finally represented as a combined matrix $M(G)$. We note that each element in the matrix $M(G)$ refers to a vertex in the graph G. We name the above procedure as subgraph normalization.

To sum up, given a set of graphs, the following steps are applied on each graph: (1) select a fixed-length sequence of central nodes from the graph; (2) assemble a fixed-size subgraph for each central node in the selected sequence; (3) normalize each extracted subgraph into a central matrix; and (4) concatenate the central matrices into a combined matrix.

Subgraph-level Attentional Convolutional Neural Networks

After transforming a graph G to a matrix $M(G)$, we next use deep convolution neural networks and attention networks to learn the features of the graph from different-levels.

The size of the input combined matrix $M(G)$ is $3N \times (w_1 + w_2 + w_3)$, where N is the number of selected central nodes, 3 is the length of the motif, and $(w_1 + w_2 + w_3)$ is the sum of the rows of the three matrix blocks. As each central matrix represents a subgraph, in order to enhance the interpretability of the model, we can design a model that ensures the independence of the feature representations among subgraphs. In the first convolution layer, the size of the convolution kernel is 1×3, and the convolution slides in both horizontal and vertical directions. The horizontal direction stride is 3, which equals to the length of the motif. The vertical direction stride is 1. We use k_1 convolution kernels to generate a $k_1 \times N \times (w_1 + w_2 + w_3)$ feature map, where each vertical feature map also characterizes the learned representation of the corresponding subgraph. In the second convolution layer, the size of convolution kernel is set to $3 \times 1 \times k_1$. To guarantee the independence of the subgraph feature representations, the horizontal stride is 1 and the vertical stride is 3 as shown in the step (1) of Figure 2. To preserve the spatial information of the graph, we do not employ any pooling operations. We denote $T = (w_1 + w_2 + w_3)/3$, and the size of output feature map of the second convolution layer is $N \times k_2 \times T$ with k_2 convolution kernels, as shown in the step (2) of Figure 2. Therefore, we can directly use the above two layer convolutional neural networks to learn different-levels and subgraph-independent feature representations for each graph, and add two layers of full-connected network for graph classification. We name the combination of Motif-matching guided Graph processing, two layers of subgraph-independent Convolutional Neural Networks and two layers of fully-connected networks as the M-GCNN. Since we have the independent representations of subgraph level features, we design a subgraph level self-attention module to measure the different impacts of the subgraphs in graph classification.

Different from traditional softmax networks, we implement self-attentional layers to capture the influences of different subgraphs on classification task. As shown in step (2) of Figure 2, we take the feature map with the size of $1 \times k_2 \times T$ as one channel of output feature map. In order to characterize the interactions among subgraphs, we first concatenate all channels of feature map into a $N \times (k_2 \cdot T)$ matrix, where each dimension in N is the vectorized representation of a subgraph, as shown in the step (3) of Figure 2. For two subgraphs $G(c_i)$ and $G(c_j)$, the corresponding vectorized features are $\vec{h}_{c_i}$ and $\vec{h}_{c_j}$, where the size of the vector is denoted as $F = k_2 \cdot T$. Then, we add a $W \in R^{F' \times F}$ weight matrix and a $\vec{a}^T \in R^{2F'}$ weight vector to learn the mutual influence among subgraphs from the vectorized features. More specifically, the attentional layer contains F' hidden neurons, and the attention mechanism is implemented by a feed forward neural network. Considering the negative values of dot product, similar to GAT model (Veličković et al. 2018), we also apply the LeakyReLU (with negative input slope $\alpha = 0.2$) function. Therefore, the coefficient of subgraph $G(c_j)$ on $G(c_i)$ computed by the attention mechanism can be computed by:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\vec{a}^T[W\vec{h}_{c_i} \parallel W\vec{h}_{c_j}]))}{\sum_{k \in N, k \neq i} \exp(\text{LeakyReLU}(\vec{a}^T[W\vec{h}_{c_i} \parallel W\vec{h}_{c_k}]))}, \quad (1)$$

where $\parallel$ denotes the concatenate operation. Following traditional attention mechanism, we perform S independent attention computations, and employ *averaging* strategy to evaluate influences. As shown in step (4) of Figure 2, the output nonlinearity uses a softmax for final classification:

$$\vec{h'}_{c_i} = \sigma(\frac{1}{S}\sum_{s=1}^{S}\sum_{t \in N, t \neq i} \alpha_{it}^{s} W^{s} \vec{h}_{c_t}), \quad (2)$$

where σ is the sigmoid function, and $\vec{h'}_{c_i}$ is the output probability distribution of $G(c_i)$. Finally, we sum all output probabilities according to the class label, as shown in step (5) of Figure 2. Finally, the class label with the maximum probability is selected as the class graph G belongs to. We name the above Motif-matching guided Graph processing, two layers of subgraph-independent Convolutional Neural Networks and self-attentional layers as the **MA-GCNN**.

The subgraph-level self-attention module is designed because different subgraphs contain different partial structural information of the entire graph. Intuitively, different subgraphs may contribute differently on the classification. Through the attentional layers, we obtain the probability distribution on the class labels for each subgraph. Moreover, the probability distribution on class labels of different subgraphs can help study the importance of the subgraphs on the classification of the entire graph. Finally, we ensemble the classification results of all the subgraphs by applying sum operation as the final probability distribution of the graph.

Experiments

We conduct experiments on multiple public benchmark datasets in the areas of bioinformatics and social network. We compare the propsoed M-GCNN and MA-GCNN models against both state-of-the-art graph kernel based methods and recent deep learning based approaches.

Datasets and Settings

We use the following five bioinformatics datasets **MUTAG**, **PTC**, **PROTEINS**, **D&D** and **NCI1**. MUTAG is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds (Debnath et al. 1991) with 7 discrete node labels, namely C, N, O, F, I, Cl, Br, and 4 discrete edge labels, including aromatic, single, double, and triple bonds. The classes indicate whether the compound has a mutagenic effect on a bacterium. PTC (Toivonen, Srinivasan, and Helma 2003) is a dataset of 344 organic molecules marked according to their carcinogenicity on male and female mice and rats. It has 19 discrete labels in nodes. PROTEINS is a graph collection obtained from (Borgwardt, Cheng, and Vishwanathan 2005) where nodes are secondary structure elements and edges indicate neighborhood in the amino-acid sequence or in 3D space with 61 discrete labels. The graphs are classified as enzyme or non-enzyme. D&D is a dataset of 1178 protein structures (Dobson and Doig 2003) with 82 discrete labels, and is also classified into enzymes and non-enzymes. NCI1 dataset is chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell lines (Wale, Watson, and Karypis 2008), and contains 4110 samples. In order to test the effectiveness of our models on unlabeled graphs, we also select five social network datasets (Yanardag and Vishwanathan 2015), including **IMDB-BINARY** (IMDB-B), **IMDB-MULTI** (IMDB-M), **REDDIT-BINARY** (RE-B), **REDDIT-MULTI-5K** (RE-M-5K) and **REDDIT-MULTI-12K** (RE-M-12K). Note that we use node degree as the attribute in these datasets, and it can easily incorporate continuous features. Both IMDB-BINARY and IMDB-MULTI are movie collaboration datasets with 1000 and 1500 graphs, respectively. The task is to identify which genre an ego-network graph belongs to. Each graph in REDDIT-BINARY, REDDIT-MULTI-5K and REDDIT-MULTI-12K datasets corresponds to an online discussion thread and the nodes denote users. There is an edge between two users if at least one of them responds to the others comment. The task in these datasets is to predict which subreddit a given discussion thread graph belongs to.

The common parameters of training the models are set as MOMENTUM = 0.9, Dropout = 0.5, learning rate = 0.001, and $L2$ norm regularization weight decay = 0.01. We set $F1 = 128, F2 = 64$ in M-GCNN model and $F' = 16, S = 8$ in MA-GCNN model. For each dataset, the parameters N, K, w_1, w_2, w_3 and training epochs are set based on the following principles: (1) N is set to the average number of nodes for all the graphs in a given graph dataset. (2) The numbers of nodes K in the subgraph are set to 10 and 20 in bioinformatics and social network datasets, respectively. (3) The numbers of w_1, w_2, w_3 are set based on the subgraph connectivity information. Considering the number of training sample and downward trend of the objective function, we adjust the batch size from 45 to 450 to get the best accuracy. We employ the cross-entropy loss function which is widely used in classification tasks. All of our experiments are evaluated using the 10-fold cross-validation method. In order to speedup the computation of the *closeness centrality*, we open up to 500 multi-threaded parallel processing.

Baseline Methods

We compare our models with both traditional graph kernel based methods and recent deep learning based graph classification approaches. Graph kernel based baselines include **Graphlet Kernel (GK)** (Shervashidze et al. 2009), **Shortest-Path Kernel (SP)** (Borgwardt, Cheng, and Vishwanathan 2005), **Weisfeiler-Lehman Sub-tree kernel (WL)** (Shervashidze et al. 2011) and **Deep Graph Kernels (DGK)** (Yanardag and Vishwanathan 2015). For deep learning based approaches, the following eight state-of-the-art GCNNs are compared: **PATCHY-SAN (PSCN)** (Niepert, Ahmed, and Kutzkov 2016), **Dynamic Edge CNN (ECC)** (Simonovsky and Komodakis 2017), **Deep Graph**

Table 1: Comparison of average classification accuracy and standard deviation on 5 bioinformatics datasets (%).

Methods	MUTAG	PTC	PROTEINS	D&D	NCI1
SP	87.28±0.55(9)	58.24±2.44 (9)	75.07±0.54 (10)	78.86± 0.26 (5)	73.47±0.11 (12)
WL	83.78±1.46 (12)	57.97±0.49 (10)	74.68 ±0.49 (11)	79.78±0.36 (3)	84.55±0.36 (1)*
GK	81.66±2.11 (13)	57.26±1.41 (11)	71.67±0.55 (12)	78.45±0.26 (7)	62.28±0.29 (13)
DGK	87.44±2.72 (8)	60.08±2.55 (7)	75.68±0.54 (8)	78.50±0.22 (6)	80.31±0.46 (8)
PSCN	92.63±4.21 (3)	62.29±5.68 (6)	75.89± 2.76 (7)	77.12±2.41 (10)	78.59±1.89 (9)
ECC	89.44±3,2 (6)	–	–	74.10±3.11 (12)	83.80±2.56 (2)
DGCNN	85.83±1.66 (11)	58.59±2.47 (8)	75.54±0.94 (9)	79.37±0.94 (4)	74.44±0.47 (11)
GCAPS-CNN	–	66.01±5.91 (4)	76.40±4.17 (5)	77.62±4.99 (9)	82.72±2.38 (4)
CapsGNN	86.67±6.88 (10)	–	76.28±3.63 (6)	75.38±4.17 (11)	78.35±1.55 (10)
AWE	87.87±9.76 (7)	–	–	71.51±4.02 (13)	–
S2S-N2N-PP	89.86±1.1 (5)	64.54±1.1 (5)	76.61±0.5 (3)	–	83.72±0.4 (3)
NEST	91.85±1.57 (4)	67.42±1.83 (3)	76.54±0.26 (4)	78.11±0.36 (8)	81.59±0.46 (6)
M-GCNN	92.78±3.56 (2)	70.30±3.59 (2)	78.19±1.93 (2)	81.37±1.11 (2)	80.91±2.17 (7)
MA-GCNN	93.89±5.24 (1)*	71.76±6.33 (1)*	79.35±1.74 (1)*	81.48±1.03 (1)*	81.77±2.36 (5)
Gain	1.26	4.34	2.74	1.70	-

Convolution Neural Network (DGCNN) (Zhang et al. 2018), **Graph Capsule CNN (GCAPS-CNN)** (Verma and Zhang 2018), **CapsGNN** (Xinyi and Chen 2019), **Anonymous Walk Embeddings (AWE)** (Ivanov and Burnaev 2018), **Sequence-to-sequence Neighbors-to-node Previous predicted (S2S-N2N-PP)** (Aynaz and Tanya 2018) and **Network Structural ConvoluTion (NEST)** (Carl et al. 2018). DGCNN enhances the pooling network by solving the underlying graph structured tasks. GCAPS-CNN combines the advantages of spectral domain GCNN and capsule networks, which further explores the permutation invariant for graph data. For other baselines, we follow the same experiment and model settings as mentioned in their papers.

Results and Analysis

In this section, we discuss the experimental results in terms of classification accuracy, standard deviation and model running time. All the reported results of comparison methods come from the original papers ("–" means not available).

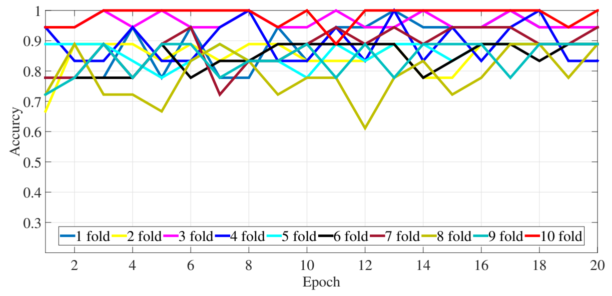
Bioinformatics Graph Classification. Table 1 shows the accuracy and standard deviations of different algorithms on the five bioinformatics datasets. One can see that M-GCNN achieves higher accuracy and lower standard deviation over MUTAG, PTC, PROTEINS and D&D datasets. Even though only the motif-matching guided graph processing and two layers of subgraph-independent convolutional neural networks are utilized, M-GCNN surpasses all the baseline methods in terms of accuracy with only one exception on NCI1. Although WL performs best on NCI1, it does not perform well on other datasets, especially on PTC and MUTAG. The performance improvement verifies that the proposed motif-matching based subgraph normalization methods better capture the spatial information through converting the graph-structured data into the grid-structured data. This table also shows that subgraph-level self-attention optimized model MA-GCNN achieves the highest accuracy on the four datasets. Compared with M-GCN, MA-GCNN further improves the classification performance for all the datasets. On MUTAG, four out of ten cross-validations achieve the 100% accuracy, and the average accuracy is 93.89%. The illustration of the accuracy in 10-fold validation for the MA-GCNN model over the MUTAG is shown in Figure 3(a), and the 1-st, 3-rd, 4-th and 10-th folds achieve the accuracy of 100%. For PTC, MA-GCNN also achieves the highest accuracy 71.77%, and outperforms the best baseline NEST by 4.35%. The illustration of the 10-fold accuracy for the MA-GCNN model in testing in PTC is shown in Figure 3(b). For MUTAG and PTC, the proposed models are trained with 20 and 100 epochs, respectively. From Figure 3, one can see that with a few epochs of training, MA-GCNN can achieve the highest testing accuracy. MA-GCNN achieves up to 2.74% and 1.70% performance improvement over S2S-N2N-PP (Aynaz and Tanya 2018) and WL (Shervashidze et al. 2011) methods on PROTEINS and D&D, respectively. For NCI1, MA-GCNN achieves better performance than seven baselines. Compared with PSCN model that is most relevant to our proposal, M-GCNN and MA-GCNN achieve 2.32% and 3.18% performance improvement in terms of average accuracy on NCI1, respectively.

Social Network Graph Classification. Table 2 shows the classification results of different algorithms on the five social network datasets. We employ the normalized node degree as the node attribute for social network datasets. Compared with the nodes in the bioinformatics graphs, the nodes in a social network have richer node attributes. One can see that both M-GCNN and MA-GCNN models achieve higher accuracy compared with all the baselines. Overall, MA-GCNN model achieves the highest accuracy over all the five datasets, and it outperforms M-GCNN. For IMDB-B and IMDB-M, MA-GCNN achieves the highest accuracy up to 77.20% and 53.77%, respectively. For REDDIT-BINARY (RE-B), REDDIT-MULTU-5K (RE-M-5K) and REDDIT-MULTU-12K (RE-M-12K), MA-GCNN achieves the highest accuracy up to 89.44%, 56.18% and 48.14%, respectively. MA-GCNN achieves 2.75%, 0.69%, 0.92%, 1.44% and 1.52% performance improvements in terms of average accuracy compared with the best baselines, with smaller standard deviations on the five datasets, respectively.

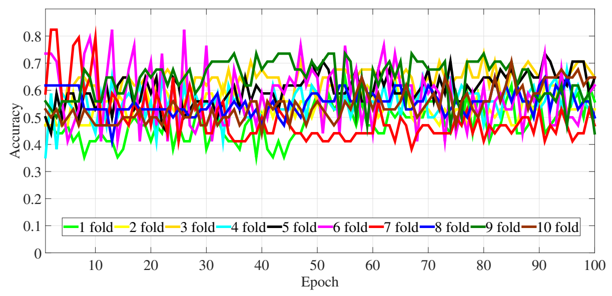
Both M-GCNN and MA-GCNN models show much more promising results against both the recently developed state-

Table 2: Comparison of average classification accuracy and standard deviation on social network datasets (%).

Methods	IMDB-B	IMDB-M	RE-B	RE-M-5K	RE-M-12K
WL	73.40±4.63 (5)	49.33±4.75 (7)	81.10±1.90 (8)	49.44±2.36 (7)	38.18±1.30 (8)
GK	65.87±0.98 (12)	43.89±0.38 (12)	77.34±0.18 (10)	41.01±0.17 (12)	31.82±0.08 (10)
DGK	66.96±0.56 (11)	44.55±0.52 (11)	78.04±0.39 (9)	41.27±0.18 (11)	32.22±0.10 (9)
PSCN	71.00±2.29 (9)	45.23±2.84 (10)	86.30±1.58 (7)	49.10±0.70 (8)	41.32±0.32 (7)
DGCNN	70.03±0.86 (10)	47.83±0.85 (9)	76.02±1.73 (11)	48.70±4.54 (9)	–
GCAPS-CNN	71.69±3.40 (8)	48.50±4.10 (8)	87.61±2.51 (5)	50.10±1.72 (6)	–
CapsGNN	73.10±4.83 (7)	50.27±2.65 (6)	–	52.88±1.48 (4)	46.62±1.90 (3)
AWE	74.45±5.83 (3)	51.58±4.66 (4)	87.89±2.53 (4)	54.74±2.93 (3)	41.51±1.98 (6)
S2S-N2N-PP	73.8±0.7 (4)	51.19±0.5 (5)	86.50±0.8 (6)	52.28±0.5 (5)	42.47±0.1 (5)
NEST	73.26±0.72 (6)	53.08±0.31 (2)	88.52±0.64 (2)	48.61±0.46 (10)	42.80±0.28 (4)
M-GCNN	75.10±3.14 (2)	52.19±2.66 (3)	88.06±1.29 (3)	55.62±2.19 (2)	47.35±1.31 (2)
MA-GCNN	77.20±2.96 (1)*	53.77±3.11 (1)*	89.44±1.18 (1)*	56.18±1.48 (1)*	48.14±1.93 (1)*
Gain	2.75	0.69	0.92	1.44	1.52



(a) MUTAG



(b) PTC

Figure 3: Illustration of the 10-fold accuracies for the MA-GCNN model in testing.

of-art deep learning approaches and graph kernels methods. The improvements in both bioinformatics datasets and social network datasets demonstrate the effectiveness of the motif-matching based subgraph normalization method, subgraph-independent convolutional neural networks and the subgraph-level self-attention layers.

Time Consumption Analysis. Compared with other deep learning based models, the proposed M-GCNN and MA-GCNN models consume more time as they need to first convert graphs to grid-structured representations. Table 3 shows the time consumption of M-GCNN and MA-GCNN models on the ten graph datasets. The second column refers to the preprocessing time of converting graphs to grid-structured representations. For MUTAG, PTC and PROTEINS datasets, our two proposals need less than hour training time to reach the highest accuracy. For social structure datasets, except for REDDIT-BINARY-5K (RE-M-5K) and REDDIT-BINARY-12K (RE-M-12K), the training time of the proposed models is around 3 hours. For the large scale graph RE-M-12K, the training time of our models is about 1 day. As the computation of closeness centrality is time consuming, the preprocessing step in our models is parallelized with different multi-threads. Compared with the significant performance improvement in terms of classification accuracy, the proposed models do not significantly increase the training time consumption.

Table 3: Time consumption of the proposed methods (mins).

Datasets	Preprocess	M-GCNN	MA-GCNN
MUTAG	1.4	3.3	4.6
PTC	0.8	4.1	6.9
PROTEINS	7.2	15.5	32.0
D&D	31.8	56.4	108.9
NCI1	4.7	48.6	72.1
IMDB-B	8.3	49.8	84.2
IMDB-M	5.6	48.1	66.8
RE-B	126.3	186.8	282.9
RE-M-5K	1284.7	462.1	588.8
RE-M-12K	354.4	954.6	1410.7

Conclusion

In this paper, we propose a novel motif-matching based subgraph normalization method to preserve spatial information of graph. By integrating motif-matching based graph processing, subgraph-independent convolutional networks and subgraph-level self-attention layers, the proposed MA-GCNN model is able to learn more discriminative features for real-world graphs to facilitate the task of graph classification. Extensive evaluations show that MA-GCNN achieves new state-of-the-art performance in both bioinformatics datasets and social network datasets. In the future, we plan to integrate more complex motif structures, such as three-hop paths or other graphlet into our models, and apply the subgraph-level self-attention graph convolution network to other downstream tasks such as the discovery of drug targets and molecular geometry optimization.

Acknowledgments

The corresponding author is Jianxin Li. This work is supported by the NSFC program (No. 61872022, 61421003, 61602237), NSF of Guangdong Province (2017A030313339), and SKLSDE-2018ZX16.

References

Anderson, B.; Quist, D.; Neil, J.; Storlie, C.; and Lane, T. 2011. Graph-based malware detection using dynamic analysis. *Journal in computer Virology*.

Aynaz, T., and Tanya, B.-W. 2018. Learning graph representations with recurrent neural network autoencoders. In *DLDay*.

Backstrom, L., and Leskovec, J. 2011. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of WSDM*.

Benkö, G.; Flamm, C.; and Stadler, P. F. 2003. A graph-based toy model of chemistry. *J. Chem. Inf. Comput. Sci.*

Benson, A. R.; Gleich, D. F.; and Leskovec, J. 2016. Higher-order organization of complex networks. *Science*.

Borgwardt, K. M., and Kriegel, H.-P. 2005. Shortest-path kernels on graphs. In *Proceedings of ICDM*. IEEE.

Borgwardt, K. M.; Cheng, S.; and Vishwanathan. 2005. Protein function prediction via graph kernels. *Bioinformatics*.

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

Carl, Y.; Mengxiong, L.; Vincent W., Z.; and Jiawei, H. 2018. Node, motif and subgraph: Leveraging network functional blocks through structural convolution. In *ASONAM*.

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

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

Dobson, P. D., and Doig, A. J. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*.

Gonzalez-Diaz, H.; Vilar, S.; Santana, L.; and Uriarte, E. 2007. Medicinal chemistry and bioinformatics-current trends in drugs discovery with networks topological indices. *Current topics in medicinal chemistry*.

Ivanov, S., and Burnaev, E. 2018. Anonymous walk embeddings. In *Proceedings of ICML*.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICML*.

Kondor, R., and Pan, H. 2016. The multiscale laplacian graph kernel. In *Proceedings of NIPS*.

Kong, X.; Yu, P. S.; Wang, X.; and Ragin, A. B. 2013. Discriminative feature selection for uncertain graph classification. In *Proceedings of SDM*.

Lee, J. B.; Rossi, R. A.; Kim, S.; Ahmed, N. K.; and Koh, E. 2018. Attention models in graphs: A survey. *arXiv preprint arXiv:1807.07984*.

Lee, J. B.; Rossi, R.; and Kong, X. 2018. Graph classification using structural attention. In *Proceedings of KDD*.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *Proceedings of ICML*.

Orsini, F.; Frasconi, P.; and De Raedt, L. 2015. Graph invariant kernels. In *Proceedings of IJCAI*.

Peng, H.; Li, J.; He, Y.; Song, Y.; and Yang, Q. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of WWW*, 1063–1072.

Shervashidze, N.; Vishwanathan, S. V. N.; Petri, T. H.; Mehlhorn, K.; and Borgwardt, K. M. 2009. Efficient graphlet kernels for large graph comparison. *AISTATS*.

Shervashidze, N.; Schweitzer, P.; van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *JMLR*.

Simonovsky, M., and Komodakis, N. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of CVPR*.

Sun, Z.; Wang, H.; Wang, H.; Shao, B.; and Li, J. 2012. Efficient subgraph matching on billion node graphs. *The VLDB Endowment*.

Toivonen, H.; Srinivasan, A.; and Helma, C. 2003. Statistical evaluation of the predictive toxicology challenge. *Bioinformatics*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In *Proceedings of ICML*.

Verma, S., and Zhang, Z. L. 2018. Graph capsule convolutional neural networks. *arXiv*.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph kernels. *JMLR*.

Wale, N.; Watson, I. A.; and Karypis, G. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019. A comprehensive survey on graph neural networks. *arXiv*.

Xinyi, Z., and Chen, L. 2019. Capsule graph neural network. In *Proceedings of ICLR*.

Yanardag, P., and Vishwanathan, S. 2015. Deep graph kernels. In *Proceedings of KDD*.

Ying, R.; You, J.; Morris, C.; Ren, X.; Hamilton, W. L.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of NIPS*.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of AAAI*.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; and Sun, M. 2018. Graph neural networks: A review of methods and applications. *arXiv*.