Depth-based Subgraph Convolutional Neural Networks

Chuanyu Xu [†], Dong Wang[†], Zhihong Zhang[*†], Beizhan Wang[†], Da Zhou[†], Guijun Ren[‡], Lu Bai[§], Lixin Cui[§], Edwin R. Hancock[¶]
[†] Xiamen University, Xiamen, China
[‡]Capital Markets Analytics, Opera Solutions, LLC, Jersey City, NJ, 07302, USA
[§]Central University of Finance and Economics, Beijing, China
[¶]Department of Computer Science, University of York, York, UK
[*]Corresponding author: zhihong@xmu.edu.cn.

***Abstract*—This paper proposes a new graph convolutional neural architecture based on a depth-based representation of graph structure, called the depth-based subgraph convolutional neural networks (DS-CNNs), which integrates both the global topological and local connectivity structures within a graph. Our idea is to decompose a graph into a family of K-layer expansion subgraphs rooted at each vertex, and then a set of convolution filters are designed over these subgraphs to capture local connectivity structural information. Specifically, we commence by establishing a family of K-layer expansion subgraphs for each vertex of graph by mapping graph to tree procedures, which can provide global topological arrangement information contained within a graph. We then design a set of fixed-size convolution filters and integrate them with these subgraphs (depicted in Figure 1). The idea is to apply convolution filters sliding over the entire subgraphs of a vertex to extract the local features analogous to the standard convolution operation on grid data. In particular, the convolution operation captures the local structural information within the graph, and has the weight sharing property among different positions of subgraph; the pooling operation acts directly on the output of the preceding layer without any preprocessing scheme (e.g., clustering or other techniques). Experiments on three graph-structured datasets demonstrate that our model DS-CNNs are able to outperform six state-of-the-art methods at the task of node classification.**

I. Introduction

Convolutional Neural Networks (CNNs) have gained significantly improvement in problems such as image classification, video recognition and machine translation, where the underlying data representation has a grid-like structure. These architectures efficiently reuse basic operations of convolution and pooling. However, many interesting tasks (such as social networks, biological networks and knowledge graphs) involve data that lies in an irregular domain which can usually be represented in the form of graphs.

Nevertheless, there is an increasing interest in the literature to extend CNNs to non-lattice graphical structures. Advances in this direction are often named as spectral approaches and spacial approaches. spectral approaches draw on a spectral representation of the graphs, i.e. the properties of convolution operators in the graph Fourier domain which are related to the Laplacian matrix of the graph [1], [2], [3]. However, the learned filters depend on the eigenvectors of the graph Laplacian, which depends on the graph structure. Thus, a model trained on a specific structure can not be directly transferred to a different graph with a different Fourier basis.

Spatial approaches [4], [5], [6] define groups of filters directly on the graph and operating on spatially close neighbors to extract local features shared across the graph. One of the challenges of these approaches is to define a fixed-size filter sliding over the graph with different sized neighborhoods and maintains the weight sharing property of CNNs, because the size and ordering of spatially close neighbors is not well definable.

In this paper, we focus on extracting both the global and local structure within a graph by linking the ideas of convolution and graph depth-based representations. We propose a depth-based subgraph convolution neural networks (DS-CNNs) to characterize the topological structure of a graph. This model is motivated by the idea that each node and its neighbors have local features shared across the graph. These features could be extracted with local convolutional filters learned from the graph and can provide a better basis for prediction. In general, the main contributions of our work are summarized as follows:

- **Depth-based subgraph convolution operation**: The depth-based subgraph convolution operation scans a 'tree' of parameters across the input data to extract local features analogous to the standard convolution operation on grid data. These local features could be composed to form multi-scale structures.
- **Weight sharing property**: The same convolution is globally valid across the subgraph, resulting in a significant parameter reduction.
- **Depth-based subgraph pooling operation**: Our depth-based subgraph pooling operation acts on the output of the preceding layer directly without any preprocessing scheme such as clustering.
- **Accuracy** : In our experiments, DS-CNNs significantly outperform several alternative methods for node classification tasks. The code will be made available for public use.

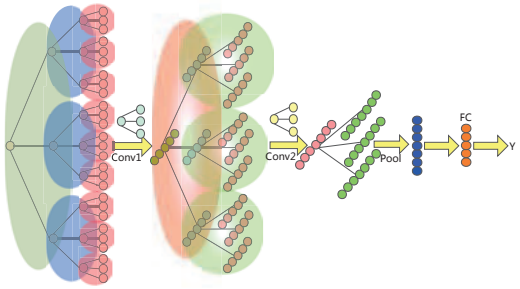

Fig. 1. An illustrative example of our DS-CNNs with $K = 4$ and $m = 3$. The 'Conv' arrow depicts the convolution operation. The subgraph above the 'Conv' arrow represents a convolution kernel, extracting structural features along the tree. Then the extracted features are summarized by pooling operation.

II. Preliminary Concepts

In this section, we introduce the related basics of graph that will be used for developing the work presented in this paper.

A. Graphs

A graph G is a pair of sets (V, E), where $V = \{v_1, ..., v_t\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges, formed by pairs of vertices. Each graph can be represented by an adjacency matrix A of size $t \times t$, where t is the number of vertices in G. In particular, $A_{i,j} = 1$ if there is an edge between vertex v_i and vertex v_j, i.e. v_i and v_j are adjacent, and $A_{i,j} = 0$ otherwise. A walk is a sequence of edges and vertices, where each edge's endpoints are the two vertices adjacent to it. A path is a walk in which all vertices are distinct (except possibly the first and last). We denote $d(u, v)$ as the length of the shortest path between vertex u and vertex v, and denote v' k-hop as the k-neighborhoods of vertex v, i.e. $d(u, v) = k$ for any vertices u of v' k-hop.

III. Proposed DS-CNNs Model

In this section we combine the idea of subgraph convolution with that of using a depth-based representation to develop a novel depth-based subgraph convolution architecture for a graph. Our idea is to decompose a graph into substructures (i.e., subgraphs) spanned from a root vertex to the remaining vertices with a K-layer expansion. More specifically, for each vertex, a neighborhood subgraph consisting of exactly m vertices is extracted by leveraging graph grafting and graph pruning procedures and normalized as a m-ary tree. The leaf nodes of the m-ary tree are further replaced by their own neighborhood m-ary trees. This process is performed recursively until a K-level m-ary tree is constructed for each vertex. We then designed a set of subgraph feature detectors, which can be viewed as convolution with a set of finite support kernels, sliding over the obtained K-level m-ary tree to extract local features as the standard convolution operation. After one layer of convolution over different positions of the subgraph along the tree structure, structural features are extracted, and a new tree is generated. The new tree has a reduced level

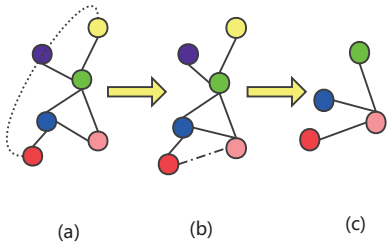

Fig. 2. An illustrative example of graph grafting. Vertices connected in dotted line are the pink vertex's 2-hop, the red vertex has a higher pagerank value than other vertices of pink vertex's 2-hop.

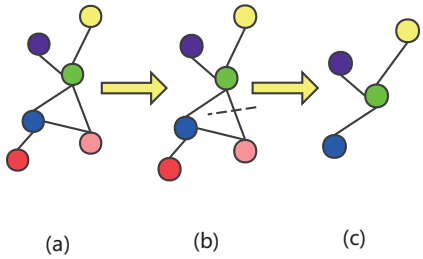

Fig. 3. An illustrative example of graph pruning. The pink vertex has a smaller pagerank value than other vertices of green vertex's 1-hop.

size compared with the input tree, where each parent node and its child nodes in the input layer become a new node in the next layer. The extracted local features produced by the convolution layer are forwarded to the pooling layer and are thereafter packed into one or more fixed-size vectors by taking the max/mean value in each dimension. After the pooling layer, the fixed-size feature vector is subsequently presented to the fully-connected layers (FC) to compute the predicted probability over the class labels. One merit of such an architecture is that each vertex has K-layer expansion subgraphs, and hence both the global topological arrangement information and local connectivity structural information contained within a graph can be learned effectively and efficiently by subgraph convolution.

A. The Depth-Based Representation for a Graph

In order to exploit topological information concerning the arrangement of vertices and edges in a graph, we develop a K-layer depth-based representation for a graph from each vertex. Concretely, it comprises two steps: (1) construct a m-ary tree with each vertex by the graph grafting and graph pruning algorithm; (2) The leaf nodes of the i-level m-ary tree are

further replaced by their own neighborhood m-ary trees hence a K-level m-ary tree is recursively constructed for each vertex.

For each vertex, a receptive field with the same size should be constructed. However, the size of each node's 1-hop are different, so we propose graph drafting and graph pruning to normalize each node's neighborhood subgraph to a m-ary tree.

Graph Grafting For node v whose size of 1-hop is less than m, we use graph grafting to choose nodes from node v' i-hop (i>=2) to fill node v' 1-hop. As shown in Figure 2 , besides the pink vertex itself, we still need to incorporate $m = 1$ vertex into the receptive field from node v' i-hop (i>=2). We prior choose nodes from node v' 2-hop, if nodes in 2-hop is not enough, then we choose nodes from 3-hop and so on. If there exist more nodes than we need, we prior choose nodes with higher pagerank values. In this way, the neighborhood subgraph consisting of exactly m vertices is extracted and normalized as an m-ary tree. Then we rank the leaf nodes of the m-ary tree according to the pagerank values of them.

Graph Pruning For node v whose size of 1-hop is more than m, we use graph pruning to choose nodes from node v' 1-hop. As shown in Figure 3, besides the green vertex itself, we need to cut one node so that only $m = 3$ vertices is reserved, we prior cut nodes with smaller pagerank values. In this way, the neighborhood subgraph consisting of exactly m vertices is extracted and normalized as an m-ary tree. Then we rank the leaf nodes of the m-ary tree according to the pagerank values of them.

Mapping Graph to Tree From graph grafting and graph pruning, we normalize each node's subgraph as an m-ary tree, the leaf nodes of each m-ary tree are further replaced by their own neighborhood m-ary trees. In this way, a K-level m-ary tree is recursively constructed for each vertex. Algorithm 1 illustrates the detail of Mapping Graph to Tree algorithm.

Algorithm 1: Mapping Graph to Tree

Input: graph, receptive field size $m+1$, pagerank algorithm, graph grafting, graph pruning, the depth K

Output: normalized neighborhood graph (K-level m-ary tree) for each vertex

```
initialization;
compute pagerank value for each vertex;
construct a m-ary tree with each vertex by the graph grafting and graph pruning algorithm;
for i = 2, i ≤ K − 1 do
    The leaf nodes of the i-level m-ary tree are further replaced by their own neighborhood m-ary trees;
end
return K-level m-ary tree for each vertex;
```

B. Depth-based Subgraph Convolution Operator

In this section, we first list the notation used in the paper, in Table I. Then, we present our depth-based subgraph convolution neural networks for the K-level m-ary tree. Figure 1 shows an example of the whole process with $K = 4$ and $m = 3$. In a manner similar to CNNs on images, our DS-CNNs also contains convolution and pooling operations. Our depth-based subgraph convolution operation extracts structural features along the tree. Then the extracted features are summarized by a depth-based subgraph pooling operation. In this way, our DS-CNNs allows effective structural feature learning.

TABLE I
IMPORTANT NOTATIONS USED IN THIS PAPER AND THEIR DESCRIPTIONS.

Symbol	Definition
$node(s,t)$	the t-th node in level s
$X^{l,p}$	the p-th feature channel in layer l
$X^{l,p}_{s,t}$	the node $(s,t)'$ p-th feature channel in layer l
$H^{l,p}_{s,t}$	$H^{l,p}_{s,t} = \{X^{l,p}_{s,t}, X^{l,p}_{s+1,(t-1)m+1}, ..., X^{l,p}_{s+1,tm}\}$ i.e. the p-th feature channel of node $(s,t)'$ receptive field in layer $l+1$
$W^{l,k,p}$	the filter mapping from the p-th feature channel in layer l to the k-th feature channel
f	the activation function
f_{l-1}	the number of filters in layer $l-1$
$b^{l,k}$	the bias of the k-th filter in layer l
$\odot$	element-wise multiplication

When CNNs are applied to images, a square grid is moved over each image with a particular step size to extract structural features as the output of the convolution. More precisely, a receptive field in the preceding layer becomes a neuron in the next layer after a convolution operation. In this way, the local structure features of images is well captured by the convolution operation. By generalizing CNNs to the K-level m-ary tree obtained in previous step of graph normalization, we scan a subgraph-based window along the tree to extract structural features as the output of our convolution.

The convolutional activation $X^{l,k}_{s,t}$ for node (s,t), feature k and layer l is given by

$$X^{l,k}_{s,t} = f(\sum_{p=1}^{f_{l-1}}(\sum_{j=1}^{m+1} W^{l,k,p}_{j} H^{l-1,p}_{s,t,j}) + b^{l,k}) \qquad s \le K - l + 1$$

The activation $X^{l,k}$ for k-th feature channel in layer l can be expressed more concisely using tensor notation as

$$X^{l,k} = f(\sum_{p=1}^{f_{l-1}}(W^{l,k,p} \odot H^{l-1,p}) + b^{l,k})$$

C. Depth-based Subgraph Pooling Operator

Another important operation proposed by CNNs is pooling. Reducing the dimensionality of the input data allows the convolution filters to have a large receptive field and at the same time decrease the number of parameters. One of the most common methods for pooling graph is the multiscale clustering of the grid and then a pooling operation over each cluster. Instead, our pooling operation acts directly on output of the preceding layer without any preprocessing scheme. The pooling activation $X^{l+1,k}_{s,t}$ for node (s,t), feature k and layer $l+l$ is given by

$$X^{l+1,k}_{s,t} = f(W^{l+1,k} \cdot pool(H^{l,k}_{s,t}) + b^{l+1,k})$$

A maximum pooling function $pool_{max}$ can be found by taking the maximum value over a region and an average pooling function $pool_{ave}$ can be obtained by taking the mean value over a region:

$$pool_{max}(R_k) = max_{i \in R_k} a_i$$

$$pool_{avg}(R_k) = \frac{1}{|R_k|} \sum_{i \in R_k} a_i$$

D. Applying DS-CNNs to Node Classification

For the purpose of node classification, each node could be represented by a K-level m-ary tree constructed through Algorithm 1. After multiple layers of applying the depth-based subgraph convolution and pooling operation, multiple features which carry different structural information become the final representation X_N of the input node. Then, the final node representation X_N is passed to a fully connected layer and outputs a conditional probability distribution $\mathbb{P}(Y|X)$, which can be obtained by applying the softmax function. This process can be formulated as below:

$$\mathbb{P}(Y|X) = \text{softmax}(f(W^d \odot X_N))$$

Moreover, our depth-based convolutional representation for graph is invariant with respect to the node index (rather than the node position), which means our activations of two isomorphic input graphs will be the same. We prove it as follows.

Theorem 1. *The depth-based convolutional activations of two isomorphic input graphs will be the same.*

Proof. We prove this theorem by contradiction.
Assume two graphs G_1 and G_2 are isomorphic but their depth-based convolutional activations are different. At least a pair of nodes u, w, where u, v belongs to the resulting node sequence of graph G_1 and G_2 respectively and will have the same position in the resulting node sequence. The activations of u and v in layer l are different. The depth-based convolutional activations of two nodes ca be written as

$$X_u^{l,k} = f(\sum_{p=1}^{f_{l-1}} (W_u^{l,k,p} \odot H_u^{l-1,p}) + b_u^{l,k})$$

$$X_v^{l,k} = f(\sum_{p=1}^{f_{l-1}} (W_v^{l,k,p} \odot H_v^{l-1,p}) + b_v^{l,k})$$

Note that

$$W_u^{l,k,p} = W_v^{l,k,p} = W^{l,k,p}$$

$$b_u^{l,k} = b_v^{l,k} = b^{l,k}$$

Graphs that are isomorphic (the same except for vertex labels) become identical after canonical graph labelling, so

$$H_u^{l-1,p} = H_v^{l-1,p} = H^{l-1,p}$$

by isomorphism, allowing us to rewrite the activation as

$$X_u^{l,k} = f(\sum_{p=1}^{f_{l-1}} (W^{l,k,p} \odot H^{l-1,p}) + b^{l,k})$$

$$X_v^{l,k} = f(\sum_{p=1}^{f_{l-1}} (W^{l,k,p} \odot H^{l-1,p}) + b^{l,k})$$

Which implies that $X_u^{l,k} = X_v^{l,k}$ and presents a contradiction and completes the proof. □

E. Learning Filters

We assume that each convolution layer l is followed by a pooling layer $l+1$. According to the back propagation algorithm says that in order to compute the sensitivity for a unit at layer l, we should first sum over the sensitivities of the next layer corresponding to units that are connected to the node of interest in the current layer l. We multiply each of those connections by the associated weights defined at layer $l+1$. We then multiply this quantity by the derivative of the activation function evaluated at the current layer's pre-activation inputs,Z. In the case of a convolutional layer followed by a pooling layer, we can upsample the pooling layer's sensitivity map $\delta^{l+1,k}$ to make it the same size as the convolutional layer's map and then just multiply the upsampled sensitivity map from layer $l+1$ with the activation derivative map at layer l element-wise. The 'weights' defined at a pooling layer map are all equal to $W^{l,k}$, so we just scale the previous step's result by $W^{l,k}$ to finish the computation of $\delta^{l,k}$. So we can get:

$$\delta^{l,k} \triangleq \frac{\partial E}{\partial Z^{l,k}}$$

$$\delta^{l,k} = \frac{\partial E}{\partial Z^{l+1,k}} \cdot \frac{\partial Z^{l+1,k}}{\partial X^{1,k}} \cdot \frac{\partial X^{l,k}}{\partial Z^{1,k}}$$

$$\delta^{l,k} = f'(Z^l) \odot (\text{up}(W^{l+1,k} \delta^{l+1,k}))$$

$$\delta^{l,k} = W^{l+1,k}(f'(Z^l) \odot \text{up}(\delta^{l+1,k}))$$

where up is the Upsampling function and E is the loss energy. Finally, the gradients for the kernel weights are computed using back propagation:

$$\frac{\partial E}{\partial W^{l,k,p}} = \sum_{i,j} (\delta^{l,k})_{i,j} (P^{l-1,p})_{i,j}$$

where $(P^{l-1,p})_{i,j}$ is the patch in $X^{l-1,p}$ that was multiplied element-wise by $W^{l,k,p}$ during convolution. we can compute the bias gradient by simply summing over all the entries in $\delta^{l,k}$:

$$\frac{\partial E}{\partial b^{l,k}} = \sum_{i,j} (\delta^{l,k})_{i,j}$$

IV. Experiments and Comparisons

In this section, we discuss the merits and limitations of the proposed DS-CNNs model. A comprehensive experimental study on a variety of data sets is conducted in order to compare our proposed model DS-CNNs with six state-of-art works in node classification.

A. Node Classification

To demonstrate the effectiveness of the proposed approach on node classification, we conduct experiments on two citation network data sets and one communication data set, i.e. Cora, Pubmed datasets [7] and Email-Eu [8]. Each citation dataset consists of scientific papers (represented by nodes), citation links (represented by edges), and subjects (represented by labels). Table. II summarizes the extent and properties of the three data sets. For node classification, six alternative algorithms are selected as baselines. We will briefly introduce these methods in turn.

TABLE II
DATASET STATISTICS OF NODE CLASSIFICATION TASK.

Dataset	Type	Nodes	Edges	Classes	Features
Cora	Citation network	2,708	5,429	7	1433
Pubmed	Citation network	19,717	44,338	3	500
Email-Eu	Communication network	1005	25,571	42	-

Datasets The Cora data set [7] contains 2,708 machine learning articles categorized into seven possible machine learning subject classes. Each article is represented by a binary $0/1$-valued word vector where each feature corresponds to the presence or absence of a term drawn from a dictionary. The dictionary contains 1,433 unique entries. This graph contains 5,429 citation edges. We treat the citation links as undirected edges and construct a binary, symmetric adjacency matrix.

The Pubmed data set [7] consists of 19,717 scientific papers from the Pubmed database on the subject of diabetes. Each paper is classified into one of three classes. This citation network that links the papers consists of 44,338 links. Each paper is represented by a Term Frequency Inverse Document Frequency (TFIDF) vector drawn from a dictionary with 500 terms. As with the CORA corpus, we construct an adjacency-based DS-CNNs that treats the citation network as an undirected graph.

The Email-Eu data set [8] was generated using email data from a large European research institution. There is an edge (u, v) in the network if person u sent person v at least one email. The e-mails only represent communication between institution members. The data set also contains "ground-truth" community memberships of the nodes. Each individual belongs to exactly one of 42 departments at the research institute. Note that the vertices of the Email-Eu-Core has no vertex information, so we only take the structural information of the vertices as the input.

Baseline Methods We compare our proposed method DS-CNNs with six state-of-the-art methods for node classification. These methods are used for comparisons are (1) $\ell 1$-regularized logistic regression (l1logistic), (2) $\ell 2$-regularized logistic regression (l2logistic), (3) exponential diffusion kernels-on-graphs (KED) [7], (4) Laplacian exponential diffusion kernels-on-graphs (KlED) [7], (5) diffusion convolutional neural networks (DCNNs) [9], (6) GraphSAGE [4]. For the 'l1logistic' and 'l2logistic' methods, we use node features alone as the input of logistic regression. This means that graph structure information is not considered, and the regularization parameter is fine tuned by the validation set. For 'KED' and 'KlED' , we take the graph structure as input, which means the node feature information is not considered. Similar to previous work [9], we chose parameters for various baseline methods as follows: a) the penalty for l1logistic and l2logistic is chosen from $\{10^{-4}, 10^{-3}, ..., 10^{3}, 10^{4}\}$, b) the parameter α for 'KED' and 'KlED' is chosen from $\{10^{-6}, 10^{-5}, ..., 10^{2}\}$, c) the parameter $H = 2$ for DCNNs because it results in the best classification accuracy, d) GraphSAGE provide a variety of approaches to aggregating features within a sampled neighborhood and we choose GraphSAGE-mean because it almost results in the best accuracy. For each baseline method, we report the results for the parameters which give the best classification accuracy.

TABLE III
THE DETAILS OF SOME PARAMETERS FOR NODE CLASSIFICATION.

Dataset	Conv2	Conv3	FC	lr	L_2	dropout
Cora	32	32	64	10^{-6}	10^{-2}	0.3
Pubmed	32	64	32	10^{-6}	10^{-2}	0.8
Email-Eu	32	32	64	10^{-4}	10^{-2}	0.8

Experimental Set-up For all datasets, we normalize each node as a 3 level 3-ary tree. We train a five-layer DS-CNNs, where the first layer is input layer, the second and third layers are the convolutional layer, the fourth layer is the fully-connected layer, and the final layer is output layer. We use the Adam optimization algorithm [10] for gradient descent. All weights are randomly initialized from a normal distribution with mean zero and variance 0.01. We choose the ReLU as the activation function. This model was implemented in Python using tensorflow [11]. A 10-fold cross-validation strategy is employed to evaluate the classification performance. Specifically, the entire sample is randomly partitioned into 10 subsets and then we choose one subset for test and use the remaining 9 for training, and this procedure is repeated 10 times. The final accuracy is computed by averaging the accuracies from each of the random subsets. The other detail of parameters setting for each dataset are listed in Table. III, in which Conv2 denotes the number of filters in layer 2, Conv3 denotes the number of filters in layer 3, FC denotes the number of neurons in FC layer, lr denotes the learning rate, L2 denotes the L2 regularization and dropout denotes the drop out.

Results Discussion Table. IV reports the average classification accuracy of the different algorithms on node classification. The boldfaced values are the best result. Our proposed five-layer DS-CNNs outperforms each of the competing methods for all datasets studied and the improvement is in the range from 1.45% to 13.09% on the Cora dataset, from 1.06% to 6.48% on the Pubmed dataset and from 3.02% to 6.33% on the Email-Eu dataset respectively. On the Cora dataset, l1logistic and l2logistic give the worst performance. This

TABLE IV
STUDY OF NODE CLASSIFICATION: CLASSIFICATION ACCURACY (IN MEAN ± STD). A COMPARISON OF THE PERFORMANCE BETWEEN SIX BASELINE METHODS AND OUR PROPOSED DS-CNNS ON THREE NODE CLASSIFICATION DATASETS. THE DS-CNNS OFFERS THE BEST PERFORMANCE. - MEANS THE MODEL IS NOT SUITABLE FOR THE DATA SET.

Model	Cora	Pubmed	Email-Eu
l1logistic	71.63 ± 0.71	87.68 ± 0.89	-
l2logistic	71.81 ± 0.69	86.54 ± 0.93	-
KED	81.92 ± 0.91	83.15 ± 0.64	70.28 ± 0.87
KlED	83.27 ± 0.76	84.11 ± 0.77	71.54 ± 0.81
DCNN	82.52 ± 2.11	88.57 ± 1.34	-
GraphSAGE	82.68 ± 1.83	88.41 ± 1.25	73.59 ± 1.72
DS-CNNs	**84.72 ± 2.28**	**89.63 ±1.67**	**76.61 ±2.33**

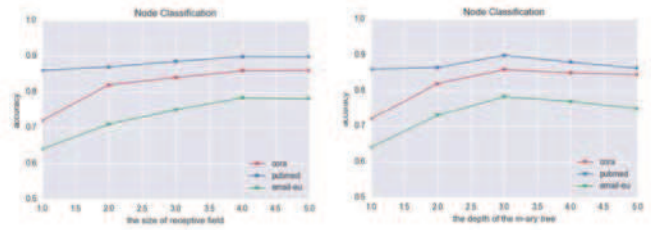

Fig. 4. Impact of the receptive field size and the depth of the m-ary tree on performance for node classification

may be explained by the fact that the logistic regression models only take the node features as input and neglect graph structure information. KED and KLED both take graph structure as input (e.g. node features are not used) and show inferior performance to our DS-CNNs. This indicates that our DS-CNNs is able to extract graph structure features. On the Pubmed dataset, we observed that those methods which incorporate node features outperform those methods that do not, i.e., l1logistic and l2logistic are superior to both KED and KLED in terms of accuracy. Furthermore, our DS-CNNs still maintains the best classification accuracy. Our DS-CNNs outperforms GraphSAGE-mean (taking the elementwise mean value of feature vectors) suggesting that assigning different importance to different nodes within a subgraph while dealing with different sized neighborhoods may be beneficial. Based on these results, it is demonstrated that our proposed method DS-CNNs integrates the merits of using both the global topological and local connectivity structures within a graph. Thus, it performs better than the traditional methods.

To investigate the effect of different receptive field size of $m+1$ and the depth K of the m-ary tree on node classification performance of proposed method DS-CNNs, we test several groups of $m+1$ and K. We report the results in Figure 4, in which the two x-axes give the varying of $m+1$ and K respectively, and the y-axis gives the classification accuracies of our DS-CNNs method. The lines of different colours represent the results on different datasets. The classification accuracies tend to become greater with increasing values of $m+1$ and K. This is because the greater values of $m+1$ and K, the more global topological and local connectivity information within a graph of our DS-CNNs method can be captured.

V. CONCLUSION AND FUTURE WORK

In this paper, we have shown how to construct depth-based subgraph convolution network for a graph. The convolution process makes use of both the global topological arrangement information and local connectivity structures within a graph. Experimental results on node classification show our DS-CNNs is superior to a number of baseline methods.

Our future plan is to extend the work as follows. In prior work we have developed methods for characterizing graphs using the commute time [12] and the heat kernel [13]. For an undirected graph, both of these methods encapsulate the path length distribution between vertices. It would be interesting to use the commute time or heat kernel as a means of node ordering.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (Grant No.61402389), the Fundamental Research Funds for the Central Universities in China (no. 20720160073) and Health joint fund of the Provincial Department of science and technology No.2015J01534. The first two authors contribute equally to this work.

REFERENCES

[1] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.
[2] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2449–2457.
[3] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.
[4] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *CoRR*, vol. abs/1706.02216, 2017.
[5] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 2014–2023.
[6] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.
[7] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, p. 93, 2008.
[8] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, "Local higher-order graph clustering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 555–564.
[9] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.
[10] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
[12] H. Qiu and E. R. Hancock, "Clustering and embedding using commute times," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, 2007.
[13] B. Xiao, E. R. Hancock, and R. C. Wilson, "Graph characteristics from the heat kernel trace," *Pattern Recognition*, vol. 42, no. 11, pp. 2589–2606, 2009.