Node2Grids: A Cost-Efficient Uncoupled Training Framework for Large-Scale Graph Learning

Dalong Yang
Sun Yat-Sen University
Guangzhou, China
ydlkevin@gmail.com

Chuan Chen*
Sun Yat-Sen University
Guangzhou, China
chenchuan@mail.sysu.edu.cn

Youhao Zheng
The University of Sydney
Sydney, Austria
yzhe1017@gmail.com

Zibin Zheng
Sun Yat-Sen University
Guangzhou, China
zhzibin@mail.sysu.edu.cn

Shih-wei Liao
National Taiwan University
Taipei, Taiwan
liao@csie.ntu.edu.tw

ABSTRACT

Graph Convolutional Network (GCN) has been widely used in graph learning tasks. However, GCN-based models (GCNs) are inherently *coupled training frameworks* repetitively conducting the recursive neighborhood aggregation, which leads to high computational and memory overheads when processing large-scale graphs. To tackle these issues, we present Node2Grids, a cost-efficient *uncoupled training framework* that leverages the independent mapped data for obtaining the embedding. Instead of directly processing the coupled nodes as GCNs, Node2Grids supports a more efficacious method in practice, mapping the coupled graph data into the independent grid-like data which can be fed into the uncoupled models as Convolutional Neural Network (CNN). This simple but valid strategy significantly saves memory and computational resources while achieving comparable results with the leading GCN-based models. Specifically, in order to support a general and convenient mapping approach, Node2Grids selects the most influential neighborhood with central node fusion information to construct the grid-like data. To further improve the downstream tasks' efficiency, a simple CNN-based neural network is employed to capture the significant information from the mapped grid-like data. Moreover, the grid-level attention mechanism is implemented, which enables implicitly specifying the different weights for the extracted grids of CNN. In addition to the typical transductive and inductive learning tasks, we also verify our framework on million-scale graphs to demonstrate the superiority of cost performance against the state-of-the-art GCN-based approaches. The codes are available on the GitHub link [1].

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; **Neural networks**.

KEYWORDS

Graph Convolutional Network, Uncoupled Training, Large-Scale Graph Learning

ACM Reference Format:
Dalong Yang, Chuan Chen, Youhao Zheng, Zibin Zheng, and Shih-wei Liao. 2021. Node2Grids: A Cost-Efficient Uncoupled Training Framework for Large-Scale Graph Learning . In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482456

1 INTRODUCTION

Graph Convolutional Network (GCN) [1] has achieved a great success in graph learning tasks, including node classification [2, 3], link prediction [4, 5] and community detection [6, 7]. By applying convolutional operations to gather the embeddings of neighbors layer by layer, GCN-based models (GCNs) significantly gain embedding for the given nodes. However, GCNs are inherently *coupled training frameworks* which repetitively propagate the representations through interactions between neighbors during training, which leads to challenges in practice due to the recursive neighborhood expansion. Specifically, unlike the *uncoupled training frameworks* where the calculations can be resolved into uncoupled units (e.g., an image for CNN model), GCNs gain embedding for a single node considering the repetitive aggregation of a great number of coupled nodes, leading to the inflexibility of the training process.

The original GCN [1] is a full-batch framework, requiring the features and adjacency relations of nodes from the full graph Laplacian available, including the nodes for testing. And the matrix manipulations are operated over the whole graph, which causes high computational and memory overheads. However, for numerous cases, the graph is expanding dynamically rather than in a fixed state, which requires an inductive framework [8] capable of generalizing well to any augmentation graph by a significant model utilizing only training set for learning [9].

*Corresponding author.
[1] https://github.com/Ray-inthebox/Node2Grids

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482456

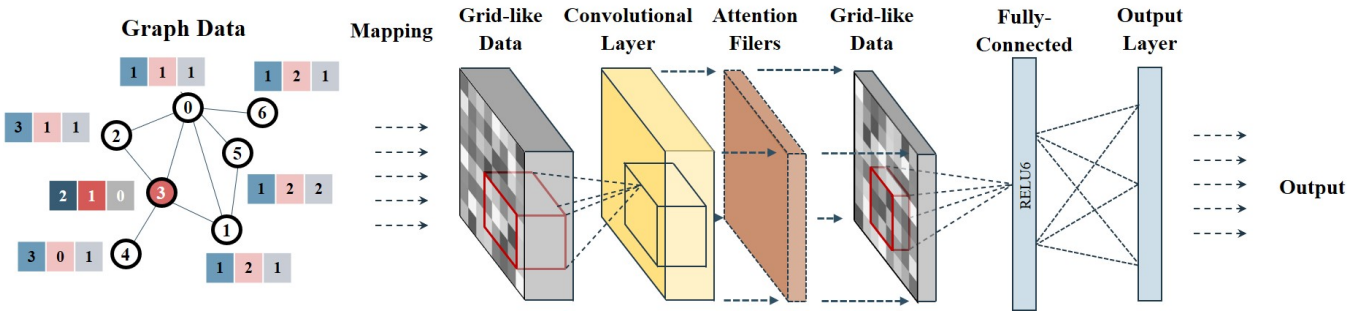


Figure 1: An illustration of the architecture for proposed Node2Grids. In this tiny example, there are seven nodes which have three features. To begin with, the central node is mapped to the Euclidean structured grid with the size of $k \times 1 \times 3$. Then a convolutional layer will be further used to extract the information from the grid-like data. In the next step, the proposed model employs attention filters to learn the weight of each "pixel" (i.e. grid). Finally, the fully-connected layers are applied to gain the output for node classification.

Instead of taking the full-batch as inputs like original GCN, the mini-batch strategy [10] achieves better performance in flexibility when processing large-scale data and inductive learning tasks. Specifically, mini-batch training updates the learnable parameters only based on the sampled nodes, which reduces the size of input data in each iteration [11]. Inspired by these advantages, there are various studies attempting to introduce mini-batch strategy to GCN framework. GraphSAGE [12] proposes an inductive learning architecture and utilizes a fixed number of neighbors for the sampled node to aggregate feature information. Graph Attention Network (GAT) [13] applies the attention mechanism to learn the attentions between central node and its neighbors. FastGCN [9] samples a fixed number of nodes for each graph layer based on the node importance. LGCN [14] builds the sub-graphs for the training nodes by adjacent information, which reduces the batch size for the training process. Cluster-GCN [11] obtains a great performance in large-scale inductive learning problems, sampling a block of nodes from the dense graph through the clustering algorithms. hGANet [15]selects k-most important neighboring nodes for the query, introducing the hard graph attention and channel-wise attention to overcome the limitation of consuming excessive computational resources. Recently, GraphSAINT [16] utilizes the graph sampling methods, constructing the mini-batches by sampling the training graph to improves training efficiency.

Even though the aforementioned mini-batch based frameworks enable enhancing GCN to some extent, the flexibility of these GCN-based models are still restricted due to the essence of coupled training, repeatedly considering the aggregations between neighboring nodes. In fact, the results realized through high-order recursive neighborhood expansion are not cost-effective. On one hand, intuitively, the features of the high-order neighbors are not highly related to the targeted nodes, where the aggregation from high-order neighboring nodes may introduce the redundant information. On the other hand, the inflexibility caused by the recursive neighborhood expansion makes the training process coupled, which leads to high computational and memory overheads. Hence, it makes sense to explore the uncoupled framework with good cost performance.

To overcome the defects of coupled frameworks, we present Node2Grids, a cost-efficient uncoupled architecture for large-scale graph learning. The framework of Node2Grids consists of two parts: (1) mapping the coupled nodes (i.e. graph data) to independent grid-like data (i.e. Euclidean structured data), and (2) employing a simple three-layer neural network based on CNN to capture the characters from the grid-like data. And these two steps are extremely simple but valid to improve the cost performance. Compared with the previous coupled works based on GCNs, Node2Grids adopts more cost-effective and simpler strategies to realize the graph data decoupling, which are illustrated in Section 3 in detail. By the strategy of mapping, the classification loss on a single node is able to be resolved into the individual term on each sample, rather than depending on a number of coupled nodes like GCNs. Additionally, the downstream task enables applying mini-batch training method to the mapped grid-like data. The batch size in the training nodes can be set elastically and the data in the same batch can be computed parallelly, which reduces the computational and memory requirements. We summarize the contributions as follow:

- For the sake of efficient cost performance, Node2Grids supports a convenient graph decoupling framework to avoid the problem of recursive neighborhood expansion. Moreover, the architecture of Node2Grids is designed extremely simply with lower time and memory overheads.
- In order to evaluate the efficiency in terms of time and memory cost, we compare the time and memory complexities of Node2Grids with the leading GCN-based methods, which theoretically proves that the uncoupled Node2Grids has comparably low overheads as the state-of-the-art GCN-based models. Moreover, the superiority in cost performance of Node2Grids is practically verified through comprehensive experiments on both transductive and inductive datasets, including the large-scale graphs.

2 RELATED WORK

Our model is related to previous graph learning techniques including Graph Convolutional Network, some recent developments in large-scale inductive learning tasks and CNN-based frameworks.

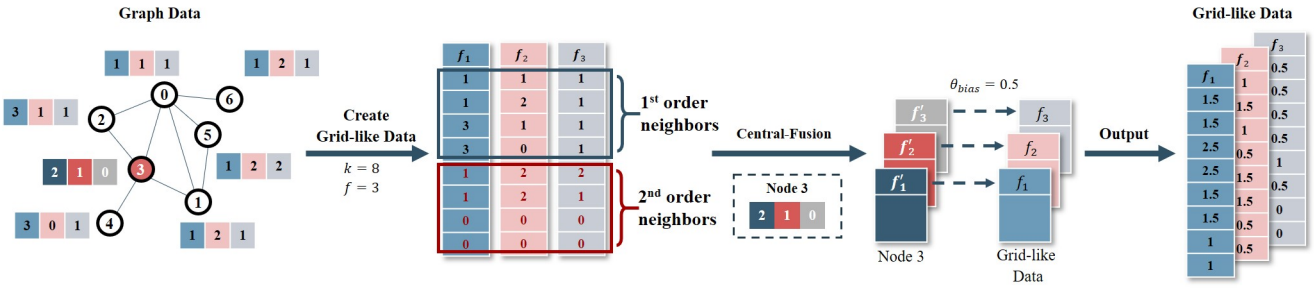


Figure 2: An example of the proposed mapping algorithm with $k = 8$. The input graph has seven nodes with three features (i.e. $f = 3$), and the brown node 3 is the given central node. To begin with, Node2Grids selects the first-order neighbors $N_1 = \{0, 1, 2, 4\}$ and second-order neighbors $N_2 = \{5, 6\}$. Then the N_1 and N_2 are ranked respectively according to the node degree. In grid-like data, each feature of the nodes represents a channel. For each channel, the proposed algorithm fills the features of the nodes in N_1 and N_2 into the grids respectively to create the Euclidean structured data (pads the nodes from N_2 after padding the nodes from N_1). Note that if $k > |N_1| + |N_2|$, the default value of an unfilled grid is zero. In the last step, after creating the grid-like data with the size of $8 \times 1 \times 3$, Node2Grids conducts information fusion (with $\theta_{bias} = 0.5$) between the central node and its neighbors, i.e. the feature of the central nodes is introduced to update the grid-like data.

Graph Convolutional Network. The process of discrete convolution is essentially operations of weighted sum [17]. Based on the spectral domain [18], GCN [1] introduces the convolutional operations on topological graph and reaches state-of-the-art performance on some datasets. Specifically, for a given graph, GCN gains a node representation by aggregating its neighbors' layer by layer. At each layer, the operations can be described as linear transformations and nonlinear activation functions. However, original GCN is a full-batch based and transductive learning model, where the fixed whole graph is utilized as input, causing a great limitation in efficiency and memory usage for large-scale graph learning. What's more, the full-batch based GCN is inapplicable for inductive learning tasks.

Large-scale graph and inductive learning. Hamiltion et al. [12] propose the concept of inductive learning on the large graph, where the nodes from the testing set are unavailable during the training process. In order to process inductive learning problems, GraphSAGE [12] trains the model through a fixed number of neighbors for each given node, where the representations are aggregated from the local neighborhood. At inference time, GraphSAGE employs the trained model to generate embedding for entirely invisible testing nodes. While GraphSAGE still suffers from the high requirements of memory due to the problem of exponential neighborhood growth.

CNN-based graph learning. Convolutional Neural Network [19] is effective and efficient in analyzing Euclidean structured data. In order to introduce CNN to graph learning tasks, it is crucial to transform graph data to grid-like data. For utilizing CNN to extraction, LGCN [14] selects a fixed number of neighboring nodes by ranking the feature values to construct the grid-like data for the given central nodes at each iteration. While this dynamic grid-like data construction during each iteration introduces the extra cost of computation. Even though LGCN leverages the strategy of sub-graphs building and CNN to enhance the model, there is still a drawback in flexibility due to the coupled pre-processing of implementing GCN to aggregate the neighboring characters.

Instead of continual and coupled neighboring aggregation as the GCN-based models (i.e., the coupled training frameworks), the proposed Node2Grids adopts a simple but effective mapping approach to extract the characters of neighborhood (i.e., mapping the coupled nodes to uncoupled grid data). Moreover, Node2Grids applies the efficacious CNN-based network to extract the meaningful information from independent grid-like data. These strategies make the graph processing tasks cost-efficient and flexible, achieving great cost performances in both transductive and inductive learning problems.

3 PROPOSED METHOD

In this section, we illustrate the framework of Node2Grids. The goal of Node2Grids is to construct a flexible framework to tackle the problems that conventional GCN-based models faced. The architecture of Node2Grids is shown in Figure 1. We first introduce the mapping strategy, and then describe the simple CNN-based three-layer neural network as well as the modified loss function we used for node classification tasks.

3.1 Grid-Like Data Generating

In order to apply efficient CNN model (i.e. an uncoupled training method) to the coupled graph data, there are several challenges caused by the gaps between graph data and grid-like data. The graph is coupled with nodes and edges according to the topology, with various numbers of the unordered neighborhood for each node. While the grid-like data comprise a fixed number of grids organized orderly and independently in space. Specifically, in generic graph, the number of each node's neighbors is not constant, yet CNN requires the spatial neighborhood size remains the same. Besides, instead of the ordered neighboring grids in grid-like data, neighbors in graph cannot be sorted directly since there is no naturally recognized rule for node ranking. In this part, we illustrate the details for the mapping approach which transforms the graph data to uncoupled grid-like data. The algorithm of mapping can be categorized

into three strategies, i.e. degree-based selection, neighborhood expansion and Central-Fusion. Notably, the mapping processing can be conducted in parallel.

Degree based selection. We propose a degree-based method for mapping which transforms coupled graph data to independent Euclidean grids. This degree-based strategy has the following benefits: 1) The node degree is a common character in graph data which makes the proposed model adaptive to various kinds of graph data. 2) As an inherent property of graph, node degree reflects the influential ability [20] and the topology of original graph [21] naturally, where a node with higher degree tends to have a greater level of influence. 3) The degrees of nodes are easy to obtain, which saves time for constructing the grids.

Neighborhood expansion. Graphs are generally sparse in the natural world. We hold an opinion that the information included in the first-order neighbors is limited for sparse networks, which is not sufficient for expressing the character of a given node. In addition, the second-order neighbors play important roles in social activities [22]. Therefore, we introduce the second-order nodes to extend the neighborhood for the given central node. To be specific, second-order neighbors are the nodes which can be reached from the given node by two hops, not included in the first-order neighbors and central node. In this case, we consider the secondary transmission of node information to enrich the expression, which achieves obtaining the more expressive Euclidean grids. Notably, considering the cost performance, Node2Grids don't bring in the higher order neighbors which are less related to the central node than the first-order and second-order neighboring nodes, and the search for high-order nodes will cause extra overheads.

Central-Fusion. The proposed model constructs the grid-like data by leveraging the feature of the central node and its neighborhood. Compared with the information carried by adjacent nodes, it is more important to consider the expression of the given the central node. There should be two keys for fully representing the given node. First of all, the central node feature should have more proportion than the neighboring nodes' in the mapped Euclidean structured data. Furthermore, the expression of the central node needs to be contained on the global grid-like data rather than in local locations. Therefore, we introduce the approach of Central-Fusion to disperse the central node feature into the Euclidean grid globally. And we will demonstrate the effectiveness of Central-Fusion by the exploratory experiments in Section 4.5 and Section 4.6.

The proposed model builds a channel of grids for each feature. Supposing the given central node with f features utilizes k neighbors to construct the Euclidean structured grid, we set the dimension of the grid-like data to $k \times 1 \times f$, where f is the number of grid channels (i.e. node features) and $k \times 1$ is the size of grid-like data (i.e. grid size) in each channel. The process of mapping is illustrated in Figure 2. For a given central node, we search for the first-order neighbors N_1 and second-order neighbors N_2 for padding, selecting the top k neighbors from the searched nodes according to their degree value. Then the proposed Node2Grids pads the grid with features of the selected nodes. Note that searching for second-order neighbors increases the time complexity, Node2Grids defines that the nodes in N_1 have the higher priority than N_2 when padding the grids (i.e. The first-order and second-order nodes are ranking through node degree separately, and after the first-order nodes are placed on the grids, we put the second-order nodes if the number of first-order neighbors is less than k), which enables controlling whether to bring in second-order neighbors by adjusting the setup of k according to the connectivity about network. Namely, if the graph is not sparse, the value of k can be set in a proper range to only employ the first-order neighbors, which are sufficient to significantly express the original graph. Notably, if $k > |N_1| + |N_2|$, the default value of an unfilled grid is set to zero. We also conduct the exploration experiments about grid size (i.e. k) in Section 4.5.

At the final step of mapping, the proposed Node2Grids conducts Central-Fusion, i.e. information fusion between the given node and the its neighbors. For the sake of expressing character of the given node around the grid-like data, we fuse current grids' features with the corresponding given node's. Concretely,

$$G = \theta_{bias} * G_c + (1 - \theta_{bias}) * G_n, \tag{1}$$

where $G_n \in \mathbb{R}^{k \times 1 \times f}$ is the grid-like data built by only selected neighbors. We obtain $G_c \in \mathbb{R}^{k \times 1 \times f}$ extended from the central node, in which the grids of each channel are filled with the corresponding features of the central node. And θ_{bias} is the bias coefficient of information fusion, which enable modifying the proportion of central node character. Of course, Node2grids adopts these easy principles to construct the grid-like data for the good efficiency, so the expression of primitive graph may not as well as the complex models with various components.

3.2 Grid-Like Data Processing

Convolutional Neural Network (CNN) is an efficient and effective approach for the Euclidean structured data processing. Inspired by these advantages, Node2Grids applies CNN to conduct downstream tasks rather than GCN. At the training phase, we apply the 1-D convolutional kernels to extract the significant information from the uncoupled grid-like data. Specifically, Node2Grids employs a simple and flexible three-layer neural network, consisting of a convolutional layer and two fully-connected layers, to conduct the node classification tasks. Moreover, we utilize the attention mechanism to learn the weights of each grid in grid-like data around all channels.

Grid-level attention mechanism. For the general Euclidean structured data such as images, "pixels" in different grids play different roles in expressing the "image" [23]. For example, the colored pixels of an image are generally more essential than the blank ones. In other words, there exist different biased weights for "pixels" in different space. Similarly, different neighbors have different influences on the central node [13]. Thus, it's significant to learn the grid-level attention (i.e. attentions on the neighbors with different level influences) for the mapped grid-like data in Node2Grids. To this end, we propose the attention mechanism which utilizes learnable attention filters to implicitly specify the weights within the grid-like data. For the sake of better specifying the weights, Node2Grids conducts attention mechanism on the extracted grid-like data of CNN, rather than executing it at the beginning to change the original mapped data. Notably, all channels of the grid-like data share the same filter parameters due to the factor that Node2Grids focuses on attentions about neighbors rather than the features (the channels), which reduces the parameters greatly. Instead of prior frameworks'

Table 1: Summary of time and memory complexities against the leading GCN-based models, where the summary of GCN-based models are from Chiang's work [11]. In this table, L is the number of GCN layers, A is the adjacency matrix of full graph, $||A||_0$ is the number of nonzeros in the adjacency matrix, N and N_t are the number of nodes from the dataset and the training set respectively. F is the number of features and the number of output features is fixed in all layers for simplicity. For SGD-based methods, b is the batch size and r is the number of sampled nodes for the predicted node. For Node2Grids, h is the number of attention filters and k is the size of grid-like data. As for memory complexity, we consider the parameters and the intermediate variables of the models for simplicity.

	GCN [1]	GraphSAGE-GCN [12]	Cluster-GCN [11]	Node2Grids
Time complexity	$O(L\|\|A\|\|_0F + LNF^2)$	$O(r^L NF^2)$	$O(L\|\|A\|\|_0F + LNF^2)$	$O(khN_tF + kN_tF^2)$
Memory complexity	$O(LNF + LF^2)$	$O(br^L F + LF^2)$	$O(bLF + LF^2)$	$O(bkF + kF^2 + kh)$

Table 2: Summary of the datasets used in the experiments.

Dataset	#Nodes	#Features	#Classes	#Training Nodes	#Validation Nodes	#Test Nodes	Task
Cora	2,708	1,433	7	140	500	1,000	Transductive
Citeseer	3,327	3,703	6	120	500	1,000	Transductive
Pubmed	19,717	500	3	60	500	1,000	Transductive
PPI	56,944	50	121	44,906	6,514	5,524	Inductive
Amazon2M	2,449,029	100	47	1,196,615	39,323	1,213,091	Inductive

numerous attention operations on all neighboring nodes, a handy attention mechanism is implemented by Hadamard product on the structured grid-like data for Node2Grids. Moreover, we will prove the effectiveness of the grid-level attention mechanism by the experiments in Section 4.6.

CNN-based network architecture. After the process of mapping, the uncoupled Euclidean structured representations are obtained for nodes, where the general convolutional kernels can be applied to extract the advanced information. Compared with previous works applying GCN to aggregate the neighboring features, a simple CNN-based three-layer network architecture is employed to process the grid-like data, which enables speeding up the training process and saving memory to enhance the scalability. To be specific, we apply a convolutional layer and two fully-connected layer in our architecture. Notably, the second fully-connected layers is used as the classifier. And the layer-wise propagation rule of Node2Grids is formulated as:

$$\begin{aligned} x_l &= \mathrm{Conv}(G), \\ x'_l &= x_l + (\frac{1}{h}\sum_{t=1}^{h} F_{att}) \circ x_l, \\ \tilde{x}_l &= g(x'_l), \end{aligned} \tag{2}$$

where $G \in \mathbb{R}^{k\times 1\times f}$ is the mapped Euclidean structured data, in which $k\times 1$ represents the grid size in each channel and f represents the number of channels. In this case, G contains the fusion character between the central node and its neighbors. $\mathrm{Conv}(\cdot)$ is a regular 1-D CNN that extracts the significant information from grid-like data, in which the kernel size is n_{ker} and the stride is set to 1, without any padding strategy. Besides, Node2Grids employs the Hadamard product $\circ$ to realize attention mechanism. The $F_{att} \in \mathbb{R}^{(k-n_{ker}+1)\times 1\times f}$ is a single attention filter, where the number of learnable parameters is $(k-n_{ker}+1)\times 1$ (i.e. all channels of grid-like data share the same filters' parameters). Note that the multi-head attention mechanism is applied, and the h refers to the number of attention heads (i.e. the number of attention filters). In addition, $g(\cdot)$ is the function of fully-connected layers and $\tilde{x}_l$ is the output of the three-layer network.

3.3 Modified Loss Function

The learnable parameters in attention filters have a great impact on the output. In order to enhance the ability to resist disturbance, the greater coefficient is set for l_2 regularization in attention filters. For making the loss function adapted to attention mechanism, we modify it by multi-value l_2-regularization. The modified loss function of Node2Grids is defined as:

$$L_{total} = L(y, \bar{y}) + \lambda \sum w^2 + \lambda_{att} \sum w_{att}^2, \tag{3}$$

where y and $\bar{y}$ are predicted labels and real labels of nodes respectively, w and w_{att} are the learnable parameters in neural networks and attention filters respectively. In addition, λ and λ_{att} are the l_2-regularization coefficients for the neural networks and attention filters respectively. $L(y, \bar{y}) + \lambda \sum w^2$ is the general loss function with l_2-regularization. And $\lambda_{att} \sum w_{att}^2$ is the formula of l_2-regularization for attention filters to prevent over-fitting. Notably, λ_{att} is set to be larger than λ, which allows a stronger constraints to the parameters updating for attention filters.

3.4 The Analysis of Time and Memory Complexities

Time and space complexities compared against the leading GCN-based models are summarized in Table 1. For Node2Grids, the time complexity of convolutional and the fully-connected layers is $O(kF^2)$. And the attention operations have the time complexity with $O(khF)$. As for memory complexity, the parameter overhead of convolutional and the fully-connected layers is $O(kF^2)$. The attention filters have the $O(kh)$ parameters. And Node2Grids should spend $O(bkF)$ memory cost for the intermediate variables in each batch. Hence, for each epoch, the overall time and memory complexity are $O(khN_tF + kN_tF^2)$ and $O(bkF + kF^2 + kh)$ respectively. Notably, from Table 1, the N_t of $O(khN_tF + kN_tF^2)$ refers to the number of training nodes instead of all the nodes from the whole dataset like N and A as GCN-based frameworks. In brief, the cost of Node2Grids only relates to the size of the current training nodes with their k neighbors rather than the recursive high-order neighborhood, which reduces the time and memory overheads.

4 EXPERIMENTS

In this section, we evaluate the effectiveness and efficiency of the proposed Node2Grids in two classes of node classification tasks, i.e. transductive and inductive learning tasks. The statistics of the datasets are summarized in Table 2. For transductive learning problems, we follow the recent studies conducting the experiments over the three benchmark datasets, i.e. Cora, Citeseer and Pubmed datasets [24]. For the inductive learning tasks, we verify the proposed Node2Grids on Protein-protein interaction (PPI) [25] and Amazon2M [11].

4.1 Baseline Methods

DeepWalk [29] is a representative network embedding method, using the random walk to sample nodes and obtains the embedding for the given node by these sampled nodes. Chebyshev [30] and GCN [1] is the leading approaches for graph learning in spectral domain. Based on GCN, hGANet [15] is the state-of-the-art model by introducing a modified attention mechanism. GraphSAGE [12], GAT [13] and LGCN [14] is the mini-batch based strategies, which can conduct the inductive learning tasks. Notably, GraphSAGE-GCN [12] is a framework extending GCN to inductive learning tasks. And GraphSAGE-LSTM [12] is the GraphSAGE-based model employing LSTM as aggregators. ClusterGCN [11] is the state-of-the-art GCN-based framework for inductive large graph learning.

4.2 Experimental Setups

In this section, we describe the experimental setups in the proposed model in both transductive and inductive learning tasks. And the experiments are conducted on the configuration of GTX 1080 Ti graphics card.

Transductive learning task. In stead of the full-batch based strategy, the proposed Node2Grids only leverages partial nodes from the training set with batch size elastically set. For Cora, Citeseer and Pubmed datasets, the batch sizes are 15, 30 and 8 respectively. We apply RMSprop optimizer [26] to train the model, with the learning rate as 0.008 and weight decay as 0.0015. The dropout strategy [27] is applied in the first fully-connected layer, with the rate of 0.6. Note that the attention mechanism is introduced, there are 50 attention heads with the extra coefficients for l_2 regularization in attention filters set to 0.0008, 0.025 and 0.07 for Cora, Citeseer and Pubmed datasets respectively.

Inductive learning task. We conduct the inductive learning experiments on protein-protein interaction (PPI) dataset and Amazon2M. The batch size is set to 2,000 and 10,000 in each iteration for PPI and Amazon2M datasets. The Nadam optimizer [28] is employed, in which the learning rate is set to 0.001 and the value of coefficient λ is set to 5×10^{-7} for l_2 regularization. In addition, The dropout [27] rate is set to 0.5 in both the convolutional layer and the first fully-connected layer. And there are also 50 attention heads applied with λ_{att} set to 1×10^{-6} for l_2 regularization in attention filters.

4.3 Tranductive Learning Tasks

4.3.1 ***Comparisons of Effectiveness.*** In transductive learning problem, The results are reported by mean classification accuracy with the standard deviation over 100 runs as the previous works in transductive tasks [1, 13–15] for ensuring consistency. From the results summarized in Table 3, the proposed Node2Grids outperforms the current leading GCN-based models, especially achieving a great improvement over the original GCN by margins of 2.2%, 2.9% and 0.8% on the datasets of Cora, Citeseer and Pubmed respectively.

4.3.2 ***Comparisons of Efficiency.*** On transductive learning tasks, we also verify the high efficiency of the proposed model by comparison with previous GCN-based approaches. In addition to the three benchmark datasets (i.e., Cora, Citeseer and Pubmed datasets), we also conduct experiments on the large-scale simulation graphs. Notably, there are two factors having impacts on the time cost, which are (1) time spending for each epoch, (2) the convergence speed. To this end, we report the average time by the training process from startup to obtain a valid testing model over 30 runs. For the sake of fairness, we take into account the time of generating the grid-like data in Node2Grids, and these three models do calculations on both training set and validation set in each epoch.

We implement the compared models by running the released codes from their GitHub pages, and use the default configurations for these three datasets. The results of efficiency comparison on real-world networks are summarized in Table 4, where the LGCN*sub* is LGCN model with the sub-graph generating algorithm to speed up the training process. From the results shown in Table 4, GCN achieves better efficiency on Cora and Citeseer datasets due to the parallel computations on small graphs. However, for Pubmed dataset, it's harder to process a large number of nodes for GCN, with efficiency greatly reduced. As for LGCN*sub*, , it spends more time because of the coupled calculation on the relatively large sub-graph for validation set. The performance demonstrates the significant time efficiency and effectiveness of the proposed Node2Grids model, especially in Pubmed which is a large graph with 19,717 nodes.

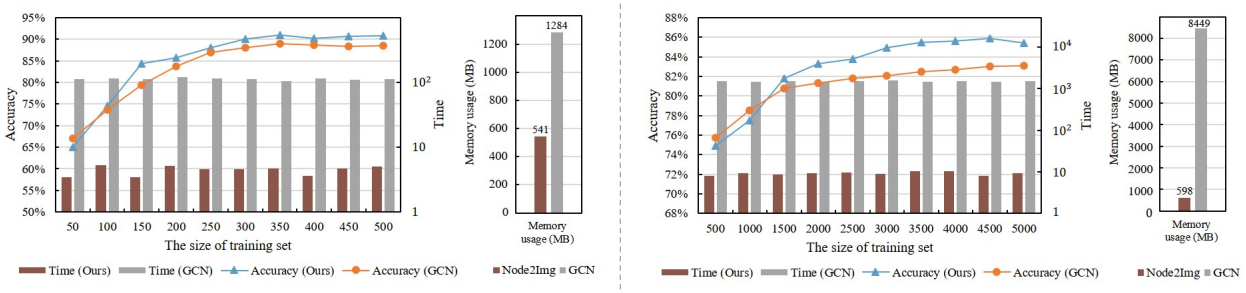
In order to further evaluate the scalability of Node2Grids, we conduct experiments on large-scale simulation networks. We generate nodes with labels based on the planted-l-partition model [31], where the feature distributions are assigned on the basis of Madelon benchmark networks [32]. Specially, we refer to Pubmed dataset

Table 3: The Summary of results for transductive learning experiments in terms of average node classification accuracy, on the dataset of Cora, Citeseer and Pubmed.

Method	Cora	Citeseer	Pubmed
DeepWalk [29]	67.2%	43.2%	65.3%
Chebyshev [30]	75.7%	64.7%	77.2%
GCN [1]	81.5%	70.3%	79.0%
GAT [13]	83.0%	72.5%	79.0%
LGCN [14]	83.3%	73.0%	79.5%
hGANet [15]	83.5%	72.7%	79.2%
Node2Grids (Ours)	**83.7 ± 0.2%**	**73.2 ± 0.4%**	**79.8 ± 0.5%**

Table 4: Results of comparison in time cost and node classification accuracy with GCN and LGCN (using the sub-graph strategy for LGCN, i.e LGCN*sub*), on the datasets of Cora, Citeseer and Pubmed. The time cost is averaged from the start to the end of the training over 30 runs. Notably, the time cost consists of both mapping and training process in the proposed Node2Grids model.

	Cora		Citeseer		Pubmed	
GCN	Time	Accuracy	Time	Accuracy	Time	Accuracy
	2.8s	81.5%	**4.6s**	70.3%	20.9s	79.0%
LGCN*sub*	Time	Accuracy	Time	Accuracy	Time	Accuracy
	53.2s	83.3%	27.6s	73.0%	57.6s	79.5%
Node2Grids (Ours)	Time	Accuracy	Time	Accuracy	Time	Accuracy
	5.3s	**83.7 ± 0.2%**	5.9s	**73.2 ± 0.4%**	**7.3s**	**79.8 ± 0.5%**



(a) N = 0.1M (M = 10^6) (b) N = 1M (M = 10^6)

Figure 3: The results of experiments on large simulation networks, where N represents the size of the dataset. The average node classification accuracy and time cost over 30 runs are reported in these experiments. In addition, the memory usages are reported on 0.1M and 1M graphs, with 500 and 5000 training batch size on these two graphs respectively.

to set average node degree and class number to 6 and 3 respectively. Additionally, each node belongs to a single class with 500 features. Two magnitudes of networks (with size of 0.1M and 1M) are employed during the experiments. And the compared model of GCN adopts two-layer convolutional networks in these simulation networks experiments. From the results reported in Figure 3, it can be observed that Node2Grids outperforms in efficiency with nearly 1/15 and 1/100 of the time cost in GCN on these two graphs respectively, while having greatly comparable performance in prediction accuracy. The memory costs are also reported in Figure 3, which takes into account the model parameters and all hidden representations for a batch. Compared with GCN, the memory usages of Node2Grids are only 1/14 of GCN on 1M network. Furthermore, both time cost and memory requirement for Node2Grids do not increase numerously when increasing the size of dataset, due to the strong parallel ability and few intermediate variables for the flexible training phase. Obviously, the gaps of memory usage as well as the time cost between these two models will further increase as the dataset expands. In conclusion, Node2Grids is a significantly low-overhead and highly parallel model, which demonstrates a superiority in processing large-scale transductive graph.

Table 5: The summary of results in terms of micro-averaged F1 score for inductive learning tasks, on the dataset of PPI.

Method	PPI
GraphSAGE-GCN [12]	0.500
GraphSAGE-LSTM [12]	0.612
LGCN [14]	0.772
GAT [13]	0.973
Node2Grids (Ours)	**0.977 ± 0.002**

Table 6: The summary of results in terms of average micro-averaged F1 score, time cost and memory usage for comparisons between GraphSAGE-GCN, Cluster-GCN and Node2Grids, on the million-scale inductive dataset of Amazon2M.

Method	Time	Memory	F1 score
GraphSAGE-GCN	**527.2s**	6,251MB	0.880±0.005
Cluster-GCN	1521.2s	5,107MB	0.884±0.003
Node2Grids (Ours)	567.5s	**3,221MB**	**0.885±0.003**

4.4 Inductive Learning Tasks

In inductive learning problems, we follow the study of [12] to report the micro-averaged F1 score (with standard deviation) of the nodes in unseen testing set over 10 runs. In addition to explore on the median size datasets, we also verify the scalability of the proposed model on the million-scale graph.

4.4.1 ***Median size dataset.*** We conduct the median size trial on PPI dataset. From Table 5, it can be observed that the proposed Node2Grids performs better than previous methods, especially gaining a great progress over the representative GCN-based mini-batch approaches. (i.e., GraphSAGE-GCN and LGCN, by margins of 0.477 and 0.205 respectively).

4.4.2 ***Million-scale dataset.*** In order to further evaluate the scalability of the proposed Node2Grids, we also conduct experiments on the Amazon2M network, which is the largest public dataset with more than two million nodes. The data format and the codes of compared GCN-based models provided by OGB Team [2] are used. To explore the models' scalability on large training data, we repartition the dataset to allocate more than one million nodes (the original partition with only 0.2M training size) to the training set, i.e., allocate the first one million nodes in the index from the test set to the training set, and evaluate on the rest nodes from the original test set.

On this dataset, Node2Grids is compared with GraphSAGE-GCN and Cluster-GCN in terms of effectiveness, time and memory overheads, where the time and memory costs are calculated as Section 4.3. The number of convolutional layers for these two GCN-based models is set to three. For the sake of fairness, the batch size of these three models is set to 1/120 of the training set and the preprocessing time is calculated. From Table 6, we can find that Node2Grids enjoys the better effectiveness and memory cost than the leading GCN-based models, where the memory cost is almost 1/2 and 3/5 of GraphSAGE-GCN and Cluster-GCN respectively, with the comparable F1 score and time efficiency. Even though GraphSAGE-GCN spends slightly less time than Node2Grids because of the few preprocessing, this coupled framework suffers from the exponential memory growth due to the recursive neighborhood expansion, requiring much more memory for training. In brief, the superiority of cost performance for Node2Grids is practically verified further through these million-scale experiments.

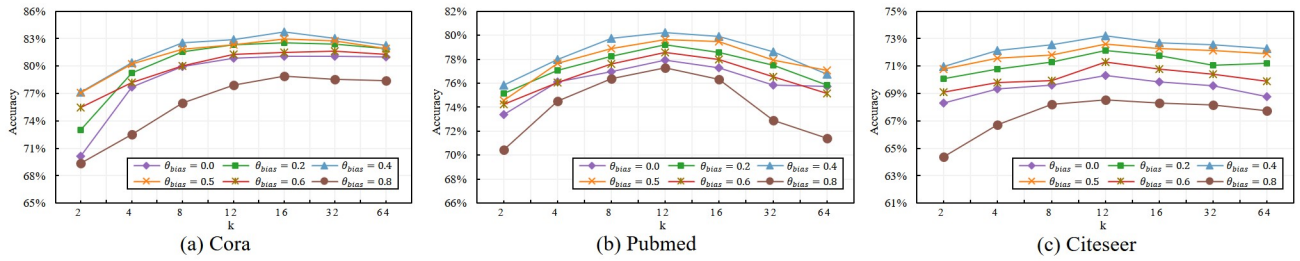
[2]https://ogb.stanford.edu/docs/nodeprop/

4.5 The Study of Central-Fusion and Grid Size

In this section, we explore the influence of hyper-parameter k and θ_{bias} on the datasets of Cora, Citeseer and Pubmed respectively. The Node2Grids utilizes central node fusion (Central-Fusion) to make sure that the character of the given central node can be expressed over the grid-like data globally, in which the θ_{bias} represents the level of central node fusion. (i.e. the greater θ_{bias}, the more partition of central node character in the Euclidean grid). Besides, We select k neighbors based on the value of degree and construct k-D grid-like data in each channel for the given node. Intuitively, the hyper-parameter k represents the level of introducing neighbors' information. We report node classification accuracy to evaluate the effect of k and θ_{bias}. The results are summarized in Figure 4.

For the effect of Central-Fusion, it can be obviously discovered that this strategy achieves positive influence by comparing the situation of $\theta_{bias} = 0$ to a significant value of θ_{bias}. From Figure 4, the model performs best on these three datasets when $\theta_{bias} = 0.4$. Particularly, Node2Grids performs worse when the θ_{bias} is greater or less than the applicable value, which indicates that it exists a reasonable proportion for central node fusion. In other words, deficiency or redundant of the central node character will result in the deviation of expressing the raw graph. As for inductive learning tasks on PPI and Amazon2M datasets, there also exist an applicable setup for θ_{bias} with values of 0.55 and 0.5 respectively.

On Cora, Citeseer and Pubmed datasets, these three networks are sparse with the average node degrees of 4, 5 and 6 respectively. It can be seen from the Figure 4 that k is set appropriately with values of 16, 12 and 12 for Cora, Citeseer and Pubmed respectively. Notably, these best setups for k bring in proper second-neighboring character to enrich the grid-like data. At a word, the proposed Node2Grids model requires a reasonable grid size (i.e. k) to build the grid-like data. When k is too small, it's difficult for the inadequate neighbors to reflect the origin graph. While k is too large, there are two aspects of decreasing the performance for Node2Grids: on one hand, the large k means selecting various of second-order neighbors for mapping, resulting in introducing the redundant second-order information which causes a negative impact on expression of the closer first-order neighbors; On the other hand, the model may pads too much zero in the grid-like data, which compromises the performance of subsequent extraction task. Moreover, we also conduct experiments to explore the setting of k for PPI and Amazon2M datasets. Note that the best setup for k is 16 on both PPI and Amazon2M graphs which is smaller than their average node degrees (with the values of 31



(a) Cora (b) Pubmed (c) Citeseer

Figure 4: The results of exploring the influence of θ_{bias} and k on the datasets of Cora, Citeseer and Pubmed respectively. The node classification accuracy is reported in these experiments. The x-coordinate and y-coordinate represent k and mean accuracy respectively. And the different colored lines represent the different occasions for θ_{bias}.

Table 7: The results of ablation experiments. The mean node classification accuracy over 30 runs are reported on this table. The $\surd$ means taking the corresponding strategy, while the $\times$ is the opposite.

#Second-Order Neighbor	#Central-Fusion	#Attention Mechanism	Cora	Citeseer	Pubmed
$\times$	$\times$	$\times$	67.6%	63.8%	40.7%
$\surd$	$\times$	$\times$	78.9%	69.5%	69.0%
$\times$	$\surd$	$\times$	77.7%	64.1%	45.7%
$\times$	$\times$	$\surd$	73.2%	69.1%	66.1%
$\surd$	$\surd$	$\times$	83.1%	72.4%	76.3%
$\surd$	$\times$	$\surd$	80.9%	70.3%	78.1%
$\times$	$\surd$	$\surd$	78.2%	70.2%	76.0%
$\surd$	$\surd$	$\surd$	**83.7%**	**73.2%**	**79.8%**

and 25 respectively), indicating that the Euclidean grid can be built without introducing the second-order neighbors if the network is not sparse.

4.6 Ablation Study

For further exploring the effectiveness of the proposed strategies, we conduct the ablation studies to justify the contribution of each component of the proposed Node2Grids. Thus, we ablate the strategies of introducing second-order neighbors, Central-Fusion and attention mechanism. Table 7 shows the quantitative comparisons of our ablated variants, where the ablation experiments are conducted on the datasets of Cora, Citeseer and Pubmed.

As we can see, each strategy boots the framework independently. Compared with the completely ablated model, the accuracy is improved by margins of 11.3%, 5.7% and 28.3% on the Cora, Citeseer and Pubmed datasets respectively by individually applying the strategy of introducing the second-order neighbors, which indicates the importance of considering the second-order node information in the sparse graphs. Meanwhile, the enhancement brought by separately implementing Central-Fusion, which is 10.1%, 0.3%, 5.0% respectively, implies the effectiveness of fusing the central node feature with the neighborhood's as Formulation (1). Likewise, the lift through applying the attention mechanism also suggests that assigning weights to the grids is warranted.

The experiments also obtain more positive impacts while using multiple strategies. Particularly, the intact model gains significant progress over the completely ablated framework by margins of 16.1%, 9.4% and 39.1% on the dataset of Cora, Citeseer and Pubmed. This demonstrates the effectiveness of the three strategies to the proposed architecture.

5 CONCLUSION

This paper presents a cost-efficient uncoupled training framework to analyze the coupled graph data, which can be applied to process the large-scale graph especially. We verify the superiority of effectiveness and efficiency of this simple uncoupled training framework through extensive experiments on both transductive and inductive learning tasks. Moreover, we also verify the effectiveness of the proposed strategies applied in Node2Grids during the experiments. Though this uncoupled framework in our work is designed simply, Node2Grids still shows comparable significant cost performance against the state-of-the-art GCN-based models. In the future, it is an interesting exploration to further design the varied mapping approaches with the corresponding neural network structures in this uncoupled training direction.

6 ACKNOWLEDGMENTS

This work is supported by the following fundings, i.e. the Key-Area Research and Development Program of Guangdong Province (2020B010165003), the National Natural Science Foundation of China (62176269, 11801595), the Guangdong Basic and Applied Basic Research Foundation (2019A1515011043), the Natural Science Foundation of Guangdong (2018A030310076) and the Tencent Wechat Rhino-bird project (2021321).

REFERENCES

[1] Thomas N. Kipf, Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

[2] Abu-El-Haija, Sami and Kapoor, Amol and Perozzi, Bryan and Lee, Joonseok. 2018. N-gcn: Multi-scale graph convolution for semi-supervised node classification. *arXiv* preprint *arXiv*:1802.08888.

[3] Wenting Zhao, Zhen Cui, Chunyan Xu, Chengzheng Li, Tong Zhang, Jian Yang. 2019. Hashing Graph Convolution for Node Classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 519-528.

[4] Muhan Zhang, Yixin Chen. 2018. Link prediction based on graph neural networks. 2018. In *Advances in Neural Information Processing Systems (NIPS)*. 5165-5175.

[5] Lei Kai, Meng Qin, Bo Bai, Gong Zhang, Min Yang. 2019. GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. 388-396.

[6] Di Jin, Ziyang Liu, Weihao Li, Dongxiao He, Weixiong Zhang. 2019. Graph convolutional networks meet Markov random fields: Semi-supervised community detection in attribute networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 33: 152-159.

[7] Fanghua Ye, Chuan Chen, Zibin Zheng, Rong-Hua Li, Jeffrey Xu Yu. Discrete Overlapping Community Detection with Pseudo Supervision. 2019. In *2019 IEEE International Conference on Data Mining (ICDM)*. 708-717.

[8] Michalski, Ryszard S. 1983. A theory and methodology of inductive learning. Machine learning. *Springer*. 83-134.

[9] Jie Chen, Tengfei Ma, Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations (ICLR)*.

[10] Hinton Geoffrey, Nitish Srivastava, Kevin Swersky. 2012. Overview of mini-batch gradient descent. In *Neural Networks for Machine Learning*. 575.

[11] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 257-266.

[12] Will Hamilton, Rex Ying, Jure Leskovec. 2017. Inductive representation learning on large graphs, In *Advances in neural information processing systems*. 1024-1034.

[13] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Pietro Lio, Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations (ICLR)*.

[14] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1416-1424.

[15] Hongyang Gao, Shuiwang Ji. Graph representation learning via hard and channel-wise attention networks. 2019. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 741-749.

[16] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations (ICLR)*.

[17] Shuangbiao Liu, QianWang, Geng Liu. 2000. A versatile method of discrete convolution and FFT (DC-FFT) for contact analyses. *Wear*. 243(1-2): 101-111.

[18] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. In *IEEE Transactions on Knowledge and Data Engineering*. 30(9): 1616-1637.

[19] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*. 1097-1105.

[20] Xiaohui Zhao, Fangai Liu, Jinlong Wang and Tianlai Li. 2017. Evaluating influential nodes in social networks by local centrality with a coefficient. In *ISPRS International Journal of Geo-Information*. 6(2):35.

[21] Hongsuda Tangmunarunki, Ramesh Govinda, Sugih Jami, Scott Shenke, Walter Willinge. 2002. Network topology generators: Degree-based vs. structural. In *ACM SIGCOMM Computer Communication Review*. 32(4): 147-159.

[22] Chen C, Zhang B, Lu Q, et al. 2015. A Consensus Algorithm Based on Nearest Second-Order Neighbors' Information[J]. *IFAC-PapersOnLine*. 48(11): 545-550.

[23] Jim Adams, Ken Parulski, and Kevin Spaulding. 1998. Color processing in digital cameras. In *IEEE micro*. 18(6): 20-30.

[24] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine*. 29(3): 93-93.

[25] Marinka Zitnik and Jure Leskovec. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*. 33(14): i190–i198.

[26] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *Coursera: Neural Networks for Machine Learning*.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*. 15(1): 1929–1958.

[28] Timothy Dozat. Incorporating Nesterov Momentum into Adam. 2016. In *ICLR Workshop*. (1):2013–2016.

[29] Bryan Perozzi, Rami Al-Rfou, Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. 701-710.

[30] Michaël Defferrard, Xavier Bresson, Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems (NIPS)*. 3844-3852.

[31] Santo Fortunato. 2010. Community detection in graphs. *Physics reports*. 486(3-5): 75-174.

[32] Guyon, Isabelle. 2003. Design of experiments for the NIPS 2003 variable selection benchmark.