Multi-Task Representation Learning with Multi-View Graph Convolutional Networks

Hong Huang, *Member, IEEE,* Yu Song, Yao Wu, *Student Member, IEEE,* Jia Shi, Xia Xie, and Hai Jin, *Fellow, IEEE*

Abstract—Link prediction and node classification are two important downstream tasks of network representation learning. Existing methods have achieved acceptable results but they perform these two tasks separately, which requires a lot of duplication of work and ignores the correlations between tasks. Besides, conventional models suffer from the identical treatment of information of multiple views, thus they fail to learn robust representation for downstream tasks. To this end, we tackle link prediction and node classification problems simultaneously via multi-task multi-view learning in this paper. We first explain the feasibility and advantages of multi-task multi-view learning for these two tasks. Then we propose a novel model named as MT-MVGCN to perform link prediction and node classification tasks simultaneously. More specifically, we design a multi-view graph convolutional network to extract abundant information of multiple views in a network, which is shared by different tasks. We further apply two attention mechanisms: view attention mechanism and task attention mechanism to make views and tasks adjust the view fusion process. Moreover, view reconstruction can be introduced as an auxiliary task to boost the performance of the proposed model. Experiments on real-world network datasets demonstrate that our model is efficient yet effective, and outperforms advanced baselines in these two tasks.

Index Terms—Representation Learning, Graph Neural Networks, Multi-task Learning, Data Mining

I. Introduction

As the networks are widespread in the real world, such as academic networks [1], biological networks [2] and social networks [3], [4], learning to make predictions or classifications for networks is appealing to a wide range of concerns. Specifically, we study two fundamental tasks for network analysis in this work: link prediction and node classification. The link prediction task is defined as estimating the existence of edges between node pairs based on network observation, while the node classification task aims to assign different class labels to the nodes.

In recent years, a great deal of efforts have been devoted to solving these two tasks, such as network embedding and graph convolutional networks. Network embedding [5] aims to learn low-dimensional representations for nodes of a network. For example, DeepWalk [6] adopts random walk to generate node sequences as word sequences then utilizes Skip-gram [7] model to get the node representations. Since many network

All the authors are with the National Engineering Research Center for Big Data Technology, Service Computing Technology and System Lab, Cluster and Grid Computing Lab and School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China. E-mail: {honghuang, yusonghust, yaowu, shijia, shelicy, hjin}@hust.edu.cn. Hong Huang is the corresponding author.

embedding methods [8]–[11] can preserve the structure and property of networks, they are very suitable for link prediction and node classification tasks. Apart from network embedding, graph convolutional networks (GCNs) [12], [13] are semi-supervised methods applicable to link prediction and node classification applications. The main idea of GCNs is to design a convolution operation for message passing between the immediate node and its neighbors, then the GCNs can be trained by a task-specific loss like Convolutional Neural Networks (CNNs). Based on that, Kipf & Welling [14] has proposed a widely used convolution operation in the spatial domain that leads to significant results in various downstream tasks. Although existing methods have achieved great results, they usually suffer from one or two main inadequacies:

(1) They are unable to utilize information encoded in multiple views while there usually exists more than one type of proximity between nodes, yielding multiple views for networks. In general, the relationships among nodes in real-world networks are sophisticated and diverse. Taking academic network as an example, we can observe the authors' interactions from three different views including co-authorship, author-citation and text-similarity views [15]. The co-authorship view depicts the cooperation relationship between two authors, the author-citation view reflects an author citing articles by another author and the proximity of two authors in the text-similarity view is defined as the textual similarity of the papers they published. Each view may be isolated and biased, thus we need to take all views into consideration to learn a robust representation.

It is of great significance to promote the collaboration of different views for learning robust representations of nodes for downstream tasks. The relationships between different views are usually complex, making the collaboration of views quite difficult. Existing multi-view models apply naive view combination strategies, such as weighted average [15], add [16] and multi-view matrix factorization [17], lack of sufficient collaboration of views, leading to the sub-optimal learnt representations. Besides, these methods adopt the fusion of views depending entirely on the characteristics of the data, such as view agreement or disagreement [18], which is not able to capture the abundant information carried by multiple views. In order to further confirm our statement, we follow the method [19] to study the agreement level among views on AMiner network (see details in Sec. V-A). Given a pair of views, each node can connect to a unique neighbor set in each view. If the Jaccard coefficient [20] between the two sets of neighbors is large than 0.5 then

Fig. 1. (a) Case study of view agreement on AMiner Network. cita: citation network, co-au: co-authorship network and text: text similarity network. (b) Case study of task correlation on AMiner Network. Correlated represents two nodes on the edge have the same label.

we think the agreement information is carried by the node in these two views, otherwise is disagreement information. Figure 1(a) shows the proportion of nodes carried agreement or disagreement information in each pair of views. As we can see, co-authorship view and text similarity view have noticeable agreement information while other pairs of views are totally different, thus it is inappropriate to simply fuse multiple views relying on agreement or disagreement.

(2) They solve these two tasks separately, which requires a large amount of repetitive work and neglects the rich correlated information between tasks. Generally, training multiple tasks simultaneously can bring more benefits than training a single task independently. First, both link prediction task and node classification task can be regarded as classification problems, with one at edge-level while the other at node-level, thus these two tasks can be optimized simultaneously. Second, these two tasks are relevant in many cases. For example, the authors in the same research filed may cite each other more often than cite other authors in different research fields; academic co-authorships are also more common among scholars in the same field and the scholars with a common research field will have higher text similarity in their papers. As shown in Figure 1(b), if there exists an edge between two nodes (authors), they will have the same label (research filed) with a high probability in each view, which indicates there is an intrinsic correlation between link prediction and node classification tasks, and multi-task learning may boost performance potentially. To this end, we are eager to design an effective model to achieve multi-task learning. It is also reasonable that different views' information contributes unequally to different tasks, thus the fusion of views needs to be decided by both the data and the targeting tasks.

To address the above limitations, we propose a novel multi-task multi-view learning framework for link prediction and node classification, namely MT-MVGCN. Specifically, our framework first takes multi-view graph convolutional networks as the backbone to learn representations of nodes in each view. After that, to encourage the collaboration between different views we design two attention mechanisms: view attention for voting the consensus representation and task-specific attention for getting specific representation for each

task. Then we sum these two representations for each task and train all tasks together. By setting the multi-view graph convolutional networks and view attention shared by tasks, we allow different tasks to transfer information implicitly. Further, we add view reconstruction as an auxiliary task, which can boost the performance of our model in some scenarios. We empirically evaluate our model on four real-world datasets, and the experimental results also indicate the effectiveness of the attention mechanisms and the benefits of multi-task learning.

In summary, our main contributions are listed as follows:

- We explore the feasibility of multi-task multi-view learning for link prediction and node classification tasks, then we develop a novel framework, namely MT-MVGCN, to train these two tasks simultaneously.

- Specifically, we design view attention mechanism to capture the consensus information of different views, meanwhile the task attention mechanism aims to extract the significant information for each task. Moreover, the view reconstruction task is introduced as an auxiliary task, which helps to learn more robust hidden representations.

- Experimental results on several multi-view networks prove our method is effective yet efficient over state-of-the-art baseline approaches.

The rest of this paper is organized as follows. Sec. II reviews the related work. Sec. III formalizes the problems; Sec. IV introduces the proposed model. Sec. V describes the design of experiments and reports the experimental results. Sec. VI presents the conclusion and future work.

II. RELATED WORK

Network Embedding. Our work is related to network embedding, which aims to find a nonlinear function to embed the raw network into a low-dimensional latent space. Nonnegative matrix factorization is widely used for network embedding [21], [22] because it makes all decomposed components non-negative and achieves linear dimension reduction at the same time [23]. Besides, random walk is also used for various models such as DeepWalk [6], node2vec [11], Struc2vec [10] and SDAE [24]. Apart from these methods, there exists some heterogeneous network embedding models. For example, Metapath2vec and HIN2Vec [25], [26] learn network embedding based meta-path for synchronized modeling of structural information and semantic association in heterogeneous networks. Shine [27] extracts potential representations of users from heterogeneous networks and predicts unobserved signs of emotional connections.

Graph Neural Networks. The convolutional neural networks are designed for the data represented as a regular grid in the Euclidean space such as pictures, which is not suitable for non-euclidean data such as networks. Inspired by the success of CNNs, various works [12], [14], [28], [29] attempt to re-define a convolutional operation on graphs. Generally, Graph Neural Networks (GNNs) can be regraded as a sample strategy for neighbors of nodes and update the representations of nodes by a weighted sum of their neighborhoods. Based on there

methods, some improvements or variants are proposed. For example, Graph Attention Network (GAT) [30] introduces attention mechanism to aggregate neighborhoods for nodes and FastGCN [31] aims to accelerate the speed of original GCN via importance sampling. Traditional deep neural network is a stack of non-linear layers thus it is also suitable for mapping original graph structure and properties into a low dimensional space. Some representative works, such as SDNE [9], SiNE [32] and Deepcas [33], provide end-to-end solutions to network problems.

Link prediction and Node classification. Link prediction and node classification are two of the most fundamental downstream tasks on network analysis, have attracted extensive attention from industry and academia [5]. For example, node2vec [11] performs link prediction on a social network, a biological network and an academic network. SiNE [32] demonstrates the excellent performance of signed network embedding on link prediction. The node classification is also widely used for different networks. Many studies [14], [15], [26], [30] also apply their models for node classification, which achieves superior performance. To name a few, in language networks, such as Wikipedia, node classification can infer the Part-of-Speech tags for words [34]. In protein-protein interactions networks, node classification is applied to classify proteins into 50 different biological states [11]. However, our method is the first attempt to solve these two tasks simultaneously, which can fully utilize the correlation between tasks to boost the performance.

Multi-task learning, Multi-view Learning and Attention Mechanisms. Multi-task and Multi-view learning are significant fields for deep learning and great of methods have been proposed in recent years. There are many successful methods and applications in various fields [35]–[37], such as computer vision, data mining and natural language processing. However, only a few of works focus on networks. For multi-view networks, existing methods adopts some naive methods to learn the representations for downstream tasks. For example, MVE [15] uses the weighted average of different views and MINES [16] adds all views together. For multi-task learning, MTGAE [38] designs a shared auto-encoder for all tasks. Although these methods mentioned in related work are effective and efficient for their problems, they are hardly satisfactory for multi-view and multi-task learning simultaneously. However, considering multi-task and multi-view learning at the same time is a trend in recent years [39], [40]. Besides, our work is related to attention mechanisms, which have been making great successes on many problems, including machine translation [41], recommendation [42], image classification [43] and so on. Attention mechanisms [15], [30], [41] aim to learn importance of different parts of the training data so that the models can focus on the most informative parts.

III. PROBLEM DEFINITION

Definition 1: **Multi-view network** Given a multi-view network $G = (V, E_1, E_2, \cdots, E_k)$, where V is the set of nodes and E_i ($1 \leq i \leq k$) is the set of edges observed from view i. Each view is regarded as a single-view network reflecting a single and distinct relationship among nodes, described by edges in E_i.

Obviously, we can utilize various methods to learn low dimension representation $\boldsymbol{Z}_i \in \boldsymbol{R}^d$ (d $\ll |V|$) for view i. After that, we still face the problem of how to fuse representations of multiple views to get the final representation for multi-task learning.

Problem 1: **View Fusion** Assuming we have obtained view representations $\{\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_k\}$, the view fusion process aims to learn a fusion function $\boldsymbol{f}$ to get the final representation $\boldsymbol{Z} = \boldsymbol{f}(\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_k)$.

Problem 2: **Multi-task Learning** Given M related or partially related tasks denoted as $\{T_m\}_{m=1}^M$, multi-task learning aims to use the knowledge contained in all or part of the M tasks to improve the learning of model.

IV. THE MT-MVGCN FRAMEWORK

In this section, we will introduce our model in details. Taking three views as an example, as illustrated in Figure 2, we first design multi-view graph convolutional networks to extract information in different views, then the view attention and task attention are applied to learn representations for link prediction and node classification tasks. Moreover, we introduce view reconstruction as an auxiliary task to improve our model. Overall, since tasks are related, it is reasonable to assume that different tasks share a common representation for the original features. Thus the multi-view graph convolutional networks and view attention mechanism are shared by tasks. Through training data in all tasks, a stronger representation can be learned for each task, and this representation can bring performance improvements. Besides, we allow different tasks to extract task-specific information through task attention mechanism.

A. Multi-View Graph Convolutional Networks

For a multi-view network $G = (V, E_1, E_2, \cdots, E_k)$, we denote adjacency matrix as $\boldsymbol{A}_i$ and feature matrix as $\boldsymbol{X}_i$ for view i. N is the number of nodes. To extract the information encoded in view i, we follow Kipf & Welling [14] to design a Graph Convolutional Network (GCN) with the following propagation rule at l-th layer:

$$\boldsymbol{Z}_i^{(l)} = \sigma(\tilde{\boldsymbol{D}}_i^{-\frac{1}{2}} \tilde{\boldsymbol{A}}_i \tilde{\boldsymbol{D}}_i^{-\frac{1}{2}} \boldsymbol{Z}_i^{(l-1)} \boldsymbol{W}^{(l)}) \qquad (1)$$

Here, we set $\tilde{\boldsymbol{A}}_i = \boldsymbol{A}_i + \boldsymbol{I}_N$ and $\boldsymbol{I}_N$ is identity matrix, $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{\boldsymbol{A}}_{ij}$, $\boldsymbol{W}^{(l)}$ is the weight matrix, $\boldsymbol{Z}_i^{(l)} \in \boldsymbol{R}^{N \times d^{(l)}}$, $\boldsymbol{Z}_i^{(0)} = \boldsymbol{X}_i$ and $\sigma(\cdot)$ is the activation function. In this paper, we choose $tanh$ as activation function in all cases. If the node features are not available the $\boldsymbol{X}_i$ will be an identity matrix.

Note that there is a key point in multi-view GCN that we make the weight matrix $\boldsymbol{W}^{(l)}$ shared by each view. Through this shared architecture[1] we can: (1) project all views into the same semantic space so that the fusion of view representations is more interpretable. (2) make our model scalable for views

---

[1]Note the node set is shared across all views. Moreover, the multi-view GCN can also be applied to such multi-view networks that have a few nodes unobserved in each view. In this case, we just need to pad those unobserved nodes' entries as zero in the adjacency matrix of each view.

Fig. 2. Overview of the proposed MT-MVGCN model. The Multi-view GCN and view attention modules are shared by different tasks. The view reconstruction module is designed as an auxiliary task.

Fig. 3. Illustration of View Attention Mechanism.

and take up less memory due to the parameters shared mechanism. (3) allow different views influence each other mutually and collaborate implicitly. After getting the final output of multi-view GCN for each view denoted as $\{\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_k\}$, we will discuss how to design the fusion function f to generate representation for each task in the next section.

B. Attention based View Fusion

In order to achieve the collaboration between different views, we need to fuse view-specific representations by a flexible way. Instead of concatenating or adding the representations in different views, we can also apply attention mechanism to capture the complex relationships between views and tasks. Concretely, view attention mechanism is proposed to make all views vote the robust representation and task attention mechanism aims to fetch the important information across different views for each task.

1) View Attention Mechanism: Recent research shows that the attention mechanism can be understood as the weighted sum of values, and the query and keys are used to calculate the weight coefficients of the corresponding values. In this work, we still apply the query, keys to calculate the attention score for each view, then we sum view-specific values according to the attention score. However, the scale of the networks are usually relative large thus we must design a time efficient way to calculate attention score for each view. Here we first introduce view attention mechanism and take three views as an example, which is illustrated in Figure 3.

As an initial step, weight matrices $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ define two learnable linear transformations for the node representations of each view respectively, then we get the query vector $\boldsymbol{Q}$ and key vectors $\boldsymbol{K}$ as follows:

$$\boldsymbol{Q}_i = \boldsymbol{Z}_i \boldsymbol{W}_Q, \quad \boldsymbol{Q} = avg[\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{Q}_3] \tag{2}$$

$$\boldsymbol{K}_i = \boldsymbol{Z}_i \boldsymbol{W}_K, \quad \boldsymbol{K} = stack[\boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{K}_3] \tag{3}$$

where the avg is element-wise average, $stack$ operation stacks all key matrices into a new matrix with rank one higher than each key matrix, $\boldsymbol{W}_Q$ and $\boldsymbol{W}_K$ are shared by all views in order to make our model be scalable for any number of views.

Since the output of view attention is the weighted average of value matrices, thus it is of great significance to require the value matrices to preserve original structural information as much as possible. Different from query and key matrices, we design a graph encoder to calculate the value matrices, aiming to further take the higher-order structural information into consideration. The graph encoder is defined by function g and weight matrix $\boldsymbol{W}_V$:

$$\boldsymbol{V}_i = g(\tilde{\boldsymbol{A}}_i, \boldsymbol{Z}_i) = \tilde{\boldsymbol{A}}_i \boldsymbol{Z}_i \boldsymbol{W}_V, \quad \boldsymbol{V} = stack[\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3] \tag{4}$$

After getting query, key, value matrices, we calculate the attention score with the intuition: the query matrix is the average information of views thus it can represent the consensus information of views. If the query matrix $\boldsymbol{Q}$ and the key matrix $\boldsymbol{K}_i$ have a larger inner product through broadcasting, meaning all views believe view i is more informative, then we will assign a larger weight for this view. Following such intuition, we compute the final representation of view attention mechanism as:

$$\boldsymbol{Z}_v = \frac{softmax(\boldsymbol{Q} \cdot \boldsymbol{K}) \cdot \boldsymbol{V}}{\sqrt{d}} \tag{5}$$

where the $softmax$ function is used to normalize all choices, d is the hidden dimension of these matrices. In practice, we may not compute the attention score with matrix multiplication due to the node size limitation in real-world networks. Compared with matrix multiplication, the inner product operation reduces the time complexity by N times, which greatly accelerates the running speed.

Since multiple views can be significantly different from each other, the variance of network data are quite high. To tackle such problem, we extend the Eq. (5) to multi-head attention to stabilize the learning process of view attention, similarly to [41]. Specifically, we utilize H attention heads to execute Eq. (5) independently and simultaneously, and then their results are concatenated, leading to the following output feature representation:

$$\boldsymbol{Z}_v = \|_{h=1}^{H}(\frac{softmax(\boldsymbol{Q}^h \cdot \boldsymbol{K}^h) \cdot \boldsymbol{V}^h}{\sqrt{d^h}}) \tag{6}$$

where $\|$ represents concatenation, and h means the h-th head.

2) Task Attention Mechanism: Similar to the view attention mechanism, task attention mechanism is also computed according to Eq. (7).

$$\boldsymbol{Z}_t = \|_{h=1}^{H}(\frac{softmax(\boldsymbol{Q}_t^h \cdot \boldsymbol{K}_t^h) \cdot \boldsymbol{V}^h}{\sqrt{d^h}}) \tag{7}$$

Where the $\boldsymbol{V}$ is also shared with view attention mechanism, but the main difference between Eq. (6) and Eq. (7) is that the $\boldsymbol{Q}_t$ and $\boldsymbol{K}_t$ are replaced by two trainable variables for each task. Due to $\boldsymbol{Q}_t$ and $\boldsymbol{K}_t$ do not know the information of views, they are trained totally by the specific task so that the task can gather useful information through optimizing these two variables. In other words, this strategy enables different tasks to assign customized weights to different views, thus only the information is believed serviceable will be chosen by each task. Besides, we should notice that view attention result can be shared by all tasks and task attention results are unique across tasks.

After we compute the results of view attention mechanism and task attention mechanism for m-th task, denoted as $\boldsymbol{Z}_v^m$ and $\boldsymbol{Z}_t^m$ respectively. We integrate the results by a hyper parameter α ranging from $0 \sim 1$ to obtain the final representation:

$$\boldsymbol{Z}^m = \alpha \boldsymbol{Z}_t^m + (1 - \alpha)\boldsymbol{Z}_v^m \tag{8}$$

C. Multi-task Learning

In this work, we study two significant tasks associated with relational: link prediction and node classification. For link prediction task, cosine similarity implies the distance between the pair of nodes in vector space, thus the cosine similarities between node pairs are estimated as features to make predictions. Given an edge e, we predict the existence of e in each view; y is the label of edge e, in which the k-th dimension is set as 1 if e exists in k-th view else set as 0, which comes down to a multi-label classification problem. As there exists a huge amount of node pairs that share no edges, predicting edge existence between all possible node pairs is unrealistic. Therefore, we only predict the existence of edges that appear in at least one view. For node classification task, we use the learnt representations as features and the node labels are already provided. For m-th task, $\boldsymbol{Z}^m$ is the final feature representation, y^m is the ground truth label, we first predict the label using a nonlinear layer: $p^m = f(\boldsymbol{Z}^m \boldsymbol{W}^m + b^m)$, f is activation function (eg. sigmoid or softmax function), $\boldsymbol{W}^m$ is the weight matrix and b^m is bias. The cross-entropy is applied for prediction loss, thus the overall loss is formulated as:

$$L = -\sum_{m=1}^{M} \lambda_m \sum_{j=1}^{N_b} [y_j^m \log p_j^m + (1 - y_j^m) \log (1 - p_j^m)] \tag{9}$$

where λ_m is a hyper-parameter determines the importance of m-th task, M is the number of tasks, and N_b is the batch size.

D. MT-MVGCN++

The attention mechanisms in MT-MVGCN employ graph encoder to preserve the structural information when computing the value vectors in Eq. (4). However, the structural information can be preserved mainly because the adjacency matrix contains abundant information rather than the original information is preserved. In addition, graph encoder may lead the features to be over smoothed [44]. Here, we introduce an auxiliary task: view reconstruction, which aims to decode the values back to the original feature space to constrain the encoder preserving enough original features. As [45] proved, the reconstruction criterion can smoothly capture the data manifolds thus preserve the original information. More specifically, we utilize the final representations of tasks to reconstruct a representation of each view: $\widetilde{\boldsymbol{Z}}_i = \tilde{g}(\tilde{\boldsymbol{A}}_i, \boldsymbol{Z}') = \tilde{\boldsymbol{A}}_i \boldsymbol{Z}' \boldsymbol{W}_D^i$, where $\boldsymbol{Z}' = \sum_{m=1}^{M} \boldsymbol{Z}^m$.

In this case, view reconstruction can be regarded as an unsupervised task to be trained with other tasks together. The objective function of view reconstruction is mean square error:

$$loss = \frac{1}{k} \frac{1}{N} \sum_{i=1}^{k} \left\| \boldsymbol{Z}_i - \widetilde{\boldsymbol{Z}}_i \right\|_F^2 \tag{10}$$

where k is the number of views, N is the node size. To this end, the objective function is the sum of prediction loss and reconstruction loss:

$$L = -\sum_{m=1}^{M} \lambda_m \sum_{j=1}^{N_b} [y_j^m \log p_j^m + (1 - y_j^m) \log (1 - p_j^m)]$$
$$+ \lambda_{M+1} \frac{1}{k} \frac{1}{N} \sum_{i=1}^{k} \| \boldsymbol{Z}_i - \widetilde{\boldsymbol{Z}}_i \|_F^2$$

$$(11)$$

E. Implementation

In practice, we utilize Tensorflow [46] for an efficient GPU-based implementation of the proposed models, then the parameters can be optimized efficiently and automatically with gradient descent and back propagation algorithm. Due to the sparsity of network data, we use sparse-dense matrix multiplication for Eq. (1), as described in [14]. Through stacking multiple GCN layers, as described in Eq. (1), we construct a multi-view GCN module to extract the information in each view. Then the outputs of multi-view GCN, i.e. $\{\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_k\}$, will be fused by view attention module and task attention module, and yield the hidden representation $\boldsymbol{Z}^m$ for m-th task. After that, we perform multiple tasks simultaneously to calculate the overall loss. Finally, the parameters can be updated through minimizing the overall loss. The reference code is available at https://github.com/yusonghust/MT-MVGCN

V. EXPERIMENT

TABLE I
STATISTICS OF THE DATASETS

Dataset	# views	# nodes	# edges	# labels	Tasks
YouTube	4	5,108	3,263,045	/	link prediction reconstruction
Twitter	4	12,741	3,154,719	/	link prediction reconstruction
Flickr	2	34,881	3,290,030	169	link prediction node classification reconstruction
Aminer	3	8,438	2,433,356	8	link prediction node classification reconstruction

TABLE II
CONFIGURATIONS OF THE PROPOSED MODELS

Datasets	YouTube	Twitter	Flickr	AMiner
GCN-Layers	3	3	2	1
GCN-Hiddensize	64	64	32	32
α	0.5	0.5	0.5	0.5
$\lambda_{linkpred}$	1.0	1.0	1.0	1.0
$\lambda_{nodecls}$	-	-	0.1	0.001
$\lambda_{reconstruction}$	0.01	0.01	0.01	0.01

A. Datasets

We select four real-world multi-view network datasets. Link prediction will be evaluated in all datasets and node classification will be evaluated in the last two datasets.

- **YouTube Dataset** For YouTube network [47], four views are constructed, representing friendship, number of common friends, number of common subscribers between two users and number of common favorite videos respectively.
- **Twitter Dataset** The twitter network [48] contains four views, including re-tweeting, reply, mention and friendship. We apply the similar method [8] to reconstruct these three views to make them denser due to the sparsity of original network.
- **Flickr Dataset** The network built from Flickr dataset [49] includes two views. Friendship view is the contact network among the blog owners. Tag-similarity view is a network with each node connecting to its top ten nearest neighbors, and the similarity is calculated based on the user's tags. The community memberships are used as node labels.
- **AMiner Dataset** The AMiner network [1] has three views: author-citation, co-authorship and text similarity. The edge weight in co-authorship view is the number of publications cooperated by each pair of authors; the edge weight in citation view is defined as the number of literature published by one author and cited by another author; the text similarity view depicts top ten nearest neighbors for each node, and the similarity is calculated based on the title and abstract by TF-IDF [50]. We only preserve authors in eight research fields as [25] as nodes and research fields are treated as node labels.

The details of datasets are listed in Table I.

B. Baselines

We choose four types of baselines: single-task single-view(STSV) based, single-task multi-view(STMV) based, multi-task single-view based(MTSV) and multi-task multi-view(MTMV) based.

- **DeepWalk [6]** DeepWalk uses random walk to construct node sequences then applies skip-gram model to learn network representation.
- **GCN [14]** GCN is a semi-supervised method based on graph convolutions.
- **GAT [30]** GAT introduces attentions mechanism for graph convolutional operations.
- **DW-con** DW-con concatenates representations learned by DeepWalk of all views to generate the final representation.
- **MVE [15]** MVE adopts attention mechanism to assign weights for the weighted sum of view representations.
- **RGCN [51]** RGCN allows multiple relationships among two nodes of a graph. Views corresponding to distinct relationships are encoded differently.
- **HIN2Vec [26]** Multi-view network is a type of heterogeneous network thus HIN2Vec is adopted to learn network representation.
- **MTGAE [38]** Multi-task graph auto-encoder defines a set of nonlinear transformations to capture graph structure for multi-task learning.

TABLE III
Quantitative results on the link prediction task for different datasets.

Category	Methods	YouTube		Twitter		Flickr		AMiner	
		AP	AUC	AP	AUC	AP	AUC	AP	AUC
STSV	DeepWalk	0.697	0.617	0.631	0.717	0.737	0.740	0.732	0.764
	GCN	0.708	0.779	0.629	0.566	0.759	0.768	0.789	0.831
STMV	DW-con	0.701	0.719	0.633	0.566	0.707	0.716	0.751	0.775
	MVE	0.723	0.627	0.642	0.546	0.669	0.676	0.709	0.736
	HIN2Vec	0.798	0.776	0.653	0.625	0.802	0.818	0.626	0.636
MTSV	MTGAE	0.782	0.798	0.649	0.619	0.783	0.808	0.786	0.833
MTMV	GCN-con	0.786	0.775	0.661	0.710	0.824	0.927	0.816	**0.857**
	MT-MVGCN	0.800	0.804	**0.669**	0.703	**0.847**	**0.955**	**0.837**	0.858
	MT-MVGCN++	**0.807**	**0.824**	0.672	**0.719**	**0.861**	0.949	0.829	0.844

- **GCN-con** A variant of MT-MVGCN, which concatenates the results of multi-view GCN to get the final representation.
- **MT-MVGCN** The proposed model for multi-task learning with information of multiple views.
- **MT-MVGCN++** A variant of MT-MVGCN, which introduces an auxiliary task: view reconstruction.

Note that we study the attention mechanisms in section V-E, so here we mainly compare our models with some representative baselines. Besides, there are some multi-view matrix factorization methods, which will not be compared with our model since they can not be extended to large-scale networks.

C. Experimental Setup

For the proposed model, the parameter settings are listed in Table II for different datasets. For DeepWalk, the number and the length of random walks for each node is set as 10 and 80 respectively. The window size of the skip-gram model is 10. For RGCN, HIN2Vec, MTGAE and MVE we use the default parameter settings in the original paper. For all baselines, we set the hidden dimension of representation same as our model. We train all models using Adam [52] optimizer with learning rate of 0.001. We also use dropout with drop rate of 0.5 and early stopping to prevent over-fitting. For two attention mechanisms and GAT, we set the number of heads as 4. We calculate the average precision score (AP), area under curve score (AUC) for link prediction and accuracy score, precision score and f1 score for node classification. As graph data has a large variance, we follow the method [14] to repeat each method for ten times, and the average metrics are reported. For single-view based methods, the best result among different views is reported. As node features are not available for our datasets the feature matrix will be an identity matrix.

D. Experimental Results

In this section, the experimental results of link prediction and node classification are presented. For link prediction the percentage of training data is 50%, and for node classification, the percentage of training data varies from 30% to 70%. The link prediction results are listed in Table III and node classification results are listed in Table IV. From the experimental results, we have the following observations and analysis:

- For link prediction task, MTSV based methods perform better than STSV based method on Flickr and AMiner datasets. In contrast, MTSV based method has similar results with STSV based methods on YouTube and Twitter datasets. This phenomenon can prove that multi-task learning is applicable and useful in many scenarios. First, as the node labels are not available for YouTube and Twitter datasets, the MTGAE model is only trained by link prediction task, thus it is not surprising that MTGAE is close to the performance of the STSV based methods. Second, link prediction task is co-training with node classification task on Flickr and AMiner datasets. Thus we can see that multi-task learning boosts the performance on these two datasets.
- For link prediction task, the results of MT-MVGCN++ are more satisfactory than MT-MVGCN on YouTube and Twitter datasets while the opposite is true on the other two datasets. On the one hand, this result is due to the fact that we introduce an auxiliary task for the first two datasets, which can also prove that multi-task learning is reasonable and useful. On the other hand, for the last two datasets, link prediction task is already co-trained with node classification task, thus the auxiliary task even makes the performance degrade slightly since it may introduce extra noises for the link prediction task.
- For node classification task, we can see that multi-task based baselines have no obvious improvement but HIN2Vec and RGCN outperform single-view based baselines. However, our methods always maintains significantly performance improvement with training percentage varying from 30% to 70%.
- As we can see, the average performance of STMV based methods is close to STSV based but the performance of MTMV based methods is better than MTSV based method. The reason may be that multi-task learning can make better use of information of multiple views. Besides, we also find that some naive view fusion methods, such as concatenation (DW-con) and average (MVE), fail to make collaboration the between different views, thus these methods achieve relatively poor performance. Lastly, the meta-path based heterogeneous network representation learning method has achieved satisfactory results but still not as good as our proposed method. Since there is only one type of

TABLE IV
QUANTITATIVE RESULTS IN TERMS OF CLASSIFICATION **ACCURACY/PRECISION/F1** SCORES ON DIFFERENT DATASETS

Category	Methods	Flickr			Aminer		
		30%	50%	70%	30%	50%	70%
STSV	DeepWalk	0.898/0.668/0.699	0.901/0.755/0.731	0.913/0.738/0.723	0.932/0.899/0.765	0.936/0.912/0.899	0.944/0.913/0.921
	GCN	0.907/0.690/0.720	0.909/0.772/0.749	0.911/0.737/0.723	0.876/0.810/0.830	0.879/0.843/0.821	0.883/0.812/0.827
	GAT	0.901/0.675/0.706	0.913/0.780/0.758	0.918/0.752/0.738	0.889/0.816/0.835	0.899/0.871/0.852	0.907/0.845/0.858
STMV	DW-con	0.883/0.633/0.666	0.883/0.719/0.693	0.884/0.673/0.656	0.875/0.806/0.826	0.876/0.844/0.823	0.877/0.805/0.820
	MVE	0.882/0.631/0.664	0.891/0.734/0.710	0.893/0.684/0.668	0.878/0.800/0.821	0.880/0.850/0.829	0.879/0.809/0.824
	RGCN	0.911/0.701/0.730	0.917/0.789/0.768	0.922/0.780/0.767	0.947/0.906/0.917	0.957/0.940/0.931	0.960/0.930/0.936
	HIN2Vec	0.910/0.698/0.727	0.911/0.776/0.754	0.914/0.766/0.753	0.961/0.934/0.941	0.962/0.951/0.943	0.961/0.935/0.941
MTSV	MTGAE	0.895/0.661/0.692	0.898/0.749/0.725	0.903/0.712/0.696	0.854/0.768/0.791	0.865/0.825/0.802	0.875/0.810/0.825
MTMV	GCN-con	0.896/0.663/0.695	0.906/0.765/0.743	0.914/0.750/0.736	0.875/0.811/0.830	0.875/0.847/0.826	0.876/0.811/0.826
	MT-MVGCN	0.906/0.688/0.718	**0.922/0.800/0.780**	0.923/0.778/0.765	0.969/0.948/0.954	0.971/0.961/0.954	0.973/0.953/0.958
	MT-MVGCN++	**0.917/0.716/0.745**	0.921/0.798/0.777	**0.935/0.800/0.788**	**0.973/0.952/0.958**	**0.977/0.970/0.965**	**0.984/0.974/0.976**

node in the multi-view network, leading to the semantic information of the meta-path greatly reduced.

- For both tasks, the proposed methods outperform all compared baselines on all datasets which indicates our methods can be better extend to link prediction and node classification tasks. We can find that although GCN-con is based on multi-task multi-view learning, the performance is still under our methods because our models adopt two attention mechanisms to make use of information from different views rather than combining all views naively. More specifically, view attention mechanism and task attention mechanism complement each other to better extract the information in each view. Besides, we observe that by simultaneously training view reconstruction task with MT-MVGCN, MT-MVGCN++ achieves better performance in most instances.

E. Ablation Study of Attention Mechanisms

There are two important attention mechanisms in the proposed methods, the view attention mechanism and task attention mechanism. In this subsection, we evaluate the contributions of these two attention mechanisms to the final results. For each dataset, we first remove the view attention mechanism or task attention mechanism from MT-MVGCN model, namely as MT-MVGCN-NVA and MT-MVGCN-NTA respectively. Apart from that, we assign equal weights for views without attention mechanisms, named as MT-MVGCN-EQU. Then we study the link prediction and node classification results with different training ratios. And the results are presented in Figure 4.

Overall, it is obvious that view attention mechanism and task attention mechanism are essential parts of the proposed model. For all datasets, there is a clear performance decline after we remove two attention mechanisms with the variation of the proportion of training data. This phenomenon shows that, rather than simply assigning equal weight to each view, the attention mechanisms make the model focus on the most informative views. Apart from that, link prediction task is more sensitive to the attention mechanisms than node classification task. As we can see, the performance of link prediction is

TABLE V
RESULTS OF ABLATION STUDIES OF MULTI-TASK LEARNING FOR THE PROPOSED MODEL. WE PERFORM EACH TASK SEPARATELY, NAMELY 'SINGLE' AND PERFORM ALL TASKS VIA MULTI-TASK LEARNING, NAMELY 'MULTI'. TIME COST IS THE AVERAGE TIME COST PER EPOCH AND IT IS THE SUM OF TASKS FOR SINGLE-TASK LEARNING.

Dataset		Flickr		AMiner	
Task	Metric	Single	Multi	Single	Multi
Link prediction	AP	0.820	**0.847**	0.851	**0.855**
	AUC	0.857	**0.955**	0.892	**0.897**
Node classification	ACC	**0.925**	0.922	0.975	**0.977**
	Precision	0.789	**0.800**	0.963	**0.970**
	F1	0.772	**0.780**	0.958	**0.965**
Time cost	Second	212	**111**	0.369	**0.284**

marked retrogressive after the removing of attention mechanisms while node classification accuracy can still keep relative stable in Flickr dataset and only degrade on AMiner dataset.

And we still notice that the importance of task attention mechanism is beyond view attention mechanism. First, there is a larger performance decline under the condition of removing task attention mechanism. We can see that AUC and AP metrics are declined by 5% on average if we remove task attention rather than view attention. Second, for link prediction task, the metrics of MT-MVGCN-NVA declines apparently on all datasets but still not sharply as MT-MVGCN-NTA. For node classification task, we even find that there are little change without view attention on Flickr dataset. By comparison, it is so clear for the declining in performance without task attention especially on AMiner dataset.

F. Ablation Study of Multi-task Learning

Although our experimental results have demonstrated the effectiveness of multi-task learning, we here study the difference between training link prediction and node classification task separately and training them together for the proposed model on Flickr and AMiner datasets. We set the train data percentage as 0.5 and keep other experimental settings the same including hardware[2]. The results are shown in Table V.

[2]A shared device with Intel(R) Xeon(R) CPU E5-2680, a Tesla P100 GPU and 250Gb Memory.

(a) **YouTube and Twitter**

(b) **Flickr**

(c) **AMiner**

Fig. 4. Comparison of performance with different attention mechanisms. MT-MVGCN-NVA: MT-MVGCN without view attention mechanism, MT-MVGCN-NTA: MT-MVGCN without task attention mechanism, MT-MVGCN-EQU: MT-MVGCN with equal weights for views.

As we can see, the multi-task learning based results still outperform the single-task based results on two datasets. For link prediction task, the multi-task learning achieves better performance by 3% on average, which means the information of node labels may be helpful for link prediction task. Besides, we still find that the influence varies for different datasets but it always positive. For node classification, we conclude that multi-task learning only brings a little influence due to the tiny difference between training tasks separately and simultaneously. Since it will not lead to the performance degradation, therefore training node classification with other tasks can be time efficient. As the graph convolutional networks tend to over-fitting resulting in unsatisfactory performance, we deduce that through multi-task learning the model is forced to learn more robust feature representations so the results are better

on the whole. First, the model must learn general features for different tasks so the possibility of over-fitting is reduced. Second, different tasks can not only provide some useful information to each other, but also introduce some noise to each other to prevent over-fitting.

There is a key point that multi-task learning is very time efficient especially for the large networks. Flickr network is several times the size of AMiner network, thus we can observe that multi-task learning saves nearly double the time cost on Flickr dataset but for the AMiner dataset it is not so obvious. As a result, we can conclude that multi-task learning is time efficient and avoids a lot of repetitive work compared with single-task learning.

G. Visualization of Attention Mechanisms

To validate the effectiveness of the view and task attention mechanisms, we explore the learned weight for each view of different datasets. Now that we have touched upon attention heads, we also show where the different heads are focusing. The visualizations of view attention weights, link prediction task attention weights and node classification task attention weights are shown in Figure 5, 6 and 7 respectively. As we can see, the weights of multiple views have noticeable difference between each other, which means that the view attention mechanism try to assign proper importance to each view. For example, the third view of YouTube dataset is more valued and the second view of Twitter view has relative large weight than other views. Moreover, we also observe that dividing the attention mechanism into multiple heads and forming multiple sub-spaces allows it to focus on different aspects of information. Compared with view attention mechanism, the task attention mechanism assigns different weight to each view since different tasks tend to utilize distinct view information to achieve better performance. Intuitively, view attention mechanism are shared by all tasks so it would like to extract consensus information while task attention mechanism prefers to extract task-specific information, leading to different weights learned by this two kinds of attention mechanisms. Overall, our model can achieve better performance via two kinds attention mechanisms capturing consensus information as well as task-specific information.

H. Parameter Sensitivity

In this section, we study the parameter sensitivity of the proposed model. There are three important parameters, the balance between two attention mechanisms α, the importance of node classification task $\lambda_{nodecls}$, and the importance of view reconstruction task $\lambda_{reconstruction}$. As we keep the importance of link prediction task as 1.0, we only need to vary the importance of other two tasks. The training percentage is set as 0.5. We choose Flickr and AMiner datasets as examples and the results are shown in Figure 8.

*1) **The influence of** α:* We first analyze how the balance of two attention mechanisms affects the experimental results. For both tasks we can see that the performance initially raises with the α increasing. The link prediction accuracy raises a little while the AUC raises sharply on both datasets. For

(a) **YouTube**   (b) **Twitter**   (c) **Flickr**   (d) **AMiner**

Fig. 5. Visualization of view weights learned by view attention mechanism of each dataset

(a) **YouTube**   (b) **Twitter**   (c) **Flickr**   (d) **AMiner**

Fig. 6. Visualization of view weights learned by link prediction task attention mechanism of each dataset

(a) **Flickr**   (b) **AMiner**

Fig. 7. Visualization of view weights learned by node classification task attention mechanism of AMiner dataset and Flickr dataset

large, the link prediction results are not very satisfactory. This phenomenon may due to the reason that if the importance of node classification task is too low the link prediction task can not obtain enough information; while the importance is too high the model will ignores the link prediction task, thus it is not surprised that the performance decline. Compared with Flickr dataset, there is no obvious change in link prediction performance when $\lambda_{nodecls}$ is small on AMiner dataset. However, Flickr dataset is faced with the performance degradation in link prediction task if the parameter is not properly set. For node classification task, the influence is basically the same as link prediction task. With the increasing of importance the model start focus more on node classification gradually thus the classification accuracy continues growing.

3) The influence of $\lambda_{reconstruction}$: Finally we study the impact of $\lambda_{reconstruction}$, which controls the contribution of reconstruction task to our model. As we can see, the auxiliary task is pretty significant for our model especially for Flickr dataset. The performance for both tasks raises evidently with the weight of view reconstruction increasing at start. Then there is a slightly degradation of performance if the weight of view reconstruction sustained increases. In contrast, it seems that link prediction task is not sensitive to the auxiliary task on AMiner dataset, but it has no negative impact on performance as long as the weight is set appropriately. All in all, our model boosts the performance in most cases through the view reconstruction as it helps to preserve structural information of different views.

node classification task, the performance on AMiner dataset is more sensitive to the attention mechanisms than its on Flickr dataset. Then with the increasing of α, the performance of both tasks keeps relatively stable without decline for two tasks. In addition, corresponding to section V-E, the task attention mechanism is key factor of the proposed models. If the weight of task attention mechanism is too low the performance is also poor. On the whole, we conclude that keeping the importance of two attention mechanisms roughly equal is a favourable setting for the model.

2) The influence of $\lambda_{nodecls}$: Then we discuss the influence of node classification task, with the importance of it varying from 0.0001 to 2.0. As observed in Figure 8, there is little difference in the effect on the two datasets. For Flickr dataset, when the hyper-parameter is too small or too

(a) **Flickr**

(b) **AMiner**

Fig. 8. Parameter Sensitivity w.r.t. the balance of two attention mechanisms α, the importance of node classification task $\lambda_{nodecls}$ and the importance of view reconstruction task $\lambda_{reconstruction}$.

I. Discussion

In this subsection, we mainly discuss the spotlights of our models and try to explain why our models perform better than experimental baselines.

Compared with existing multi-view based methods, such as average, concatenate and add, our models achieve great results due to the advanced view fusion strategy. First, we propose an attention-based fusion method which can capture the complex relationships between views. We not only allow all views to vote for the robust representation, but also allow the specific task to focus on the most informative views. Second, as mentioned in [53], average fails to capture repeat features in graph, thus average methods and something like that may also fail to fuse different views information since it is common for repeating features among views. Third, add and concatenate methods are enough to preserve the original information, but may introduce extra noise and lack sufficient collaboration of views, making the results sub-optimal. Finally, previous attention based methods like MVE [15] adopts simple attention mechanism to take an average of all views, which is not interpretable and effective as our models. In contrast, by applying two multi-head attention mechanisms our models can also compute diverse attentions independently on different sub-spaces in parallel.

Compared with the existing multi-task based method, i.e. MTGAE, our models include shared layers and task specific

layers, which brings great flexibility to multi-task learning. MTGAE [38] is composed of a symmetrical graph auto-encoder shared by all tasks so it is hard to extract specific information for a certain task. The advantage of our models lies in that view attention is shared by tasks and task attention is specific to each task, expected to capture more complex relationships between network information and tasks. In addition, the auxiliary task also makes great contributions to multi-task learning.

VI. Conclusion

In this paper, we study the problem of link prediction and node classification tasks by multi-task learning with multi-view graph convolutional networks. More specifically, our model considers utilizing information of multiple views for multiple tasks through two attention mechanisms. Besides, we enhance our model by introducing an auxiliary task which can make the network structural information better preserved. In the experiments, we show that our model is significantly and consistently superior to the start-of-the-art baselines. We also study the contribution of two attention mechanisms and the parameter sensitivity. In the future, we plan to explore two directions based on the current work: one is how to make the model learn the importance of different tasks by itself rather than controlling it by hyper-parameters. The other is applying our model to more realistic scenarios and problems.

REFERENCES

[1] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 990–998.

[2] T. Li, R. Wernersson, R. B. Hansen, H. Horn, J. Mercer, G. Slodkowicz, C. T. Workman, O. Rigina, K. Rapacki, H. H. Stærfeldt *et al.*, "A scored human protein–protein interaction network to catalyze genomic interpretation," *Nature methods*, vol. 14, no. 1, p. 61, 2017.

[3] R. Van Noorden, "Online collaboration: Scientists and the social network," *Nature news*, vol. 512, no. 7513, p. 126, 2014.

[4] D. M. Romero, B. Uzzi, and J. Kleinberg, "Social networks under stress," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 9–20.

[5] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 27th Conference on Advances in neural information processing systems*, 2013, pp. 3111–3119.

[8] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of World Wide Web Conference*, 2015, pp. 1067–1077.

[9] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.

[10] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 385–394.

[11] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[12] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 30th Conference on Advances in neural information processing systems*, 2016, pp. 3844–3852.

[13] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[15] J. T. Meng, J. Shang, X. Ren, M. Zhang, and J. Han, "An attention-based collaboration framework for multi-view network representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. ACM, 2017, pp. 1767–1776.

[16] Y. Ma, Z. Ren, Z. Jiang, J. Tang, and D. Yin, "Multi-dimensional network embedding with hierarchical structure," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 387–395.

[17] H. Zhao, Z. Ding, and Y. Fu, "Multi-view clustering via deep matrix factorization." in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 2017, pp. 2921–2927.

[18] S. Sun, "A survey of multi-view machine learning," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2031–2038, 2013.

[19] Y. Shi, F. Han, X. He, X. He, C. Yang, J. Luo, and J. Han, "mvn2vec: Preservation and collaboration in multi-view network embedding," *arXiv preprint arXiv:1801.06597*, 2018.

[20] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of jaccard coefficient for keywords similarity," in *Proceedings of the international multiconference of engineers and computer scientists*, vol. 1, no. 6, 2013, pp. 380–384.

[21] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 203–209.

[22] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 459–467.

[23] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proceedings of the 15th Conference on Advances in neural information processing systems*, 2001, pp. 556–562.

[24] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 1145–1152.

[25] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2017, pp. 135–144.

[26] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1797–1806.

[27] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, and Q. Liu, "Shine: Signed heterogeneous information network embedding for sentiment link prediction," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 592–600.

[28] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proceedings of the 31st Conference on Advances in Neural Information Processing Systems*, 2017, pp. 3697–3707.

[29] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1263–1272.

[30] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[31] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," in *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[32] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, 2017, pp. 327–335.

[33] C. Li, J. Ma, X. Guo, and Q. Mei, "Deepcas: An end-to-end predictor of information cascades," in *Proceedings of the 26th international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 577–586.

[34] C. Tu, W. Zhang, Z. Liu, M. Sun *et al.*, "Max-margin deepwalk: Discriminative learning of network representation." in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2016, pp. 3889–3895.

[35] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *Proceedings of the 13th European conference on computer vision*. Springer, 2014, pp. 94–108.

[36] A. Trivedi, P. Rai, H. Daumé III, and S. L. DuVall, "Multiview clustering with incomplete views," in *NIPS Workshop*, vol. 224, 2010.

[37] P. Liu, X. Qiu, and X. Huang, "Adversarial multi-task learning for text classification," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 1–10.

[38] P. V. Tran, "Multi-task graph autoencoders," in *Workshop on Relational Representation Learning, NIPS 2018, Montréal, Canada.*, 2018.

[39] C.-T. Lu, L. He, W. Shao, B. Cao, and P. S. Yu, "Multilinear factorization machines for multi-task multi-view learning," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 701–709.

[40] X. Zhang, X. Zhang, H. Liu, and X. Liu, "Multi-task multi-view clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3324–3338, 2016.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31th Conference on Advances in neural information processing systems*, 2017, pp. 5998–6008.

[42] D. Khattar, V. Kumar, V. Varma, and M. Gupta, "Hram: A hybrid recurrent attention machine for news recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 1619–1622.

[43] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *Proceedings of the Twenty-eighth Conference on Advances in neural information processing systems*, 2014, pp. 2204–2212.

[44] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the Thirty-Second AAAI Conference*, 2018.

[45] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[46] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, 2016, pp. 265–283.

[47] R. Zafarani and H. Liu, "Social computing data repository at asu [http://socialcomputing. asu. edu]. tempe, az: Arizona state university, school of computing," *Informatics and Decision Systems Engineering*, 2009.

[48] M. De Domenico, A. Lima, P. Mougel, and M. Musolesi, "The anatomy of a scientific rumor," *Scientific reports*, vol. 3, p. 2980, 2013.

[49] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 817–826.

[50] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242. Piscataway, NJ, 2003, pp. 133–142.

[51] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proceedings of the 15th European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[53] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, Conference Track Proceedings*, 2019.

Yao Wu received the B.S. degree in communication from Huazhong University of Science and Technology, Wuhan, China, in 2016. Currently, she is pursuing her Ph.D. degree in Huazhong University of Science and Technology. Her research interests include machine learning, data mining, social network analysis.

Jia Shi received his B.S. degree from Lanzhou University, China, in 2018. Currently, he is pursuing his master degree in Huazhong University of Science and Technology. His research fields include data mining and natural language processing.

Xia Xie is an associate professor at Huazhong University of Science and Technology (HUST) in China. She received her Ph.D. in computer architecture from HUST in 2006. Her research interests include data mining and big data.

Hong Huang is an associate professor in Huazhong University of Science and Technology, China. She received her Ph.D degree in Computer Science from University of Göttingen, Germany in 2016, and her M.E. degree in Electronic Engineering from Tsinghua University, Beijing, China in 2012. Her research interests lie in social network analysis, data mining and knowledge graph.

Yu Song received his B.S. degree in Electronic Information Engineering from Huazhong University of Science and Technology, Wu Han, China, in 2018. Currently, he is pursuing his master degree in Huazhong University of Science and Technology. His research interests include data mining and social network analysis.

Hai Jin is a Cheung Kung Scholars Chair Professor of computer science and engineering at Huazhong University of Science and Technology (HUST) in China. Jin received his PhD in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. Jin worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. Jin is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientists of National 973 Basic Research Program Project of Virtualization Technology of Computing System, and Cloud Security. Jin is a fellow of the IEEE, a fellow of the China Computer Federation (CCF), and a member of the Association for Computing Machinery (ACM). He has co-authored 22 books and published over 700 research papers. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, big data and network security.