IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 16, 2020, accepted February 13, 2020, date of publication March 2, 2020, date of current version March 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977332

WMGCN: Weighted Meta-Graph Based Graph Convolutional Networks for Representation Learning in Heterogeneous Networks

JINLI ZHANG[1], ZONGLI JIANG[1], ZHENG CHEN[2], AND XIAOHUA HU[2]
[1]Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[2]College of Computing and Informatics, Drexel University, Philadelphia, PA 19104, USA

Corresponding author: Jinli Zhang (lz73798@gmail.com)

This work was supported in part by the China Scholarship Council under Grant NSF IIS 1815256 and Grant NSF IIS 1744661.

ABSTRACT Network embedding has been an effective tool to analyze heterogeneous networks (HNs) by representing nodes in a low-dimensional space. Although many recent methods have been proposed for representation learning of HNs, there is still much room for improvement. Random walks based methods are currently popular methods to learn network embedding; however, they are random and limited by the length of sampled walks, and have difficulty capturing network structural information. Some recent researches proposed using meta paths to express the sample relationship in HNs. Another popular graph learning model, the graph convolutional network (GCN) is known to be capable of better exploitation of network topology, but the current design of GCN is intended for homogenous networks. This paper proposes a novel combination of meta-graph and graph convolution, the meta-graph based graph convolutional networks (MGCN). To fully capture the complex long semantic information, MGCN utilizes different meta-graphs in HNs. As different meta-graphs express different semantic relationships, MGCN learns the weights of different meta-graphs to make up for the loss of semantics when applying GCN. In addition, we improve the current convolution design by adding node self-significance. To validate our model in learning feature representation, we present comprehensive experiments on four real-world datasets and two representation tasks: classification and link prediction. WMGCN's representations can improve accuracy scores by up to around 10% in comparison to other popular representation learning models. What's more, WMGCN'feature learning outperforms other popular baselines. The experimental results clearly show our model is superior over other state-of-the-art representation learning algorithms.

INDEX TERMS Heterogeneous network, weighted meta-graph, graph convolutional network, representation learning.

I. INTRODUCTION

Networks are widely used for constructing and organizing a variety of objects by concisely representing the relationships in the interrelated world. For instance, social networks, biological networks, transportation networks, and bibliographic information networks are recently drawing increasing attention. All of these networks could be categorized as two groups - homogeneous networks and heterogeneous networks by types of entities and links. As shown in Fig. 1, a co-author homogeneous network contains only a single type of entity: the authors. For another example, the bibliographic information network in Fig. 2 consists of many types of entities: universities (U), authors (A), papers (P) and conferences (C); they form a heterogeneous network with different relationships between different types of entities. The main goal of this paper is to improve feature learning from heterogenous networks, using combined techniques of meta-graph and graph convolution, so that the learned features can in turn improve tasks such as node classification and link prediction.

[Heterogeneous Networks (HNs)] Current popular researches on heterogeneous networks involve various data mining tasks. These researches believe the heterogeneous network is a more appropriate graph modeling paradigm to express the gigantic complex relationships and multi-typed entities in the real world. The researchers from both academia and industry have applied HNs in a wide range of

The associate editor coordinating the review of this manuscript and approving it for publication was Fatos Xhafa.

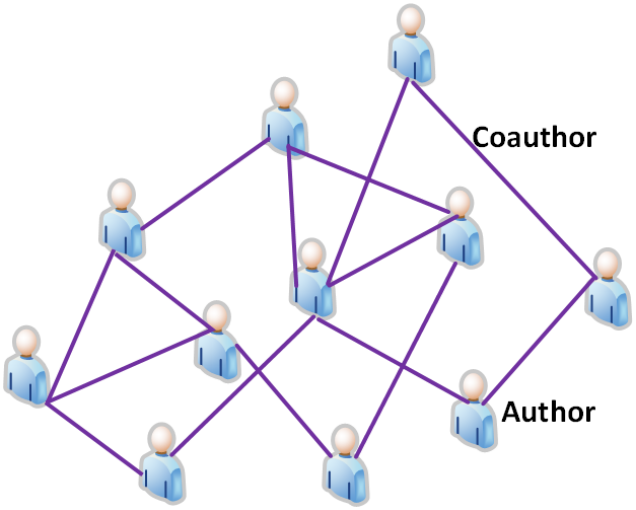


FIGURE 1. The illustration of a Heterogeneous Network. There are four types nodes: authors, papers, university.

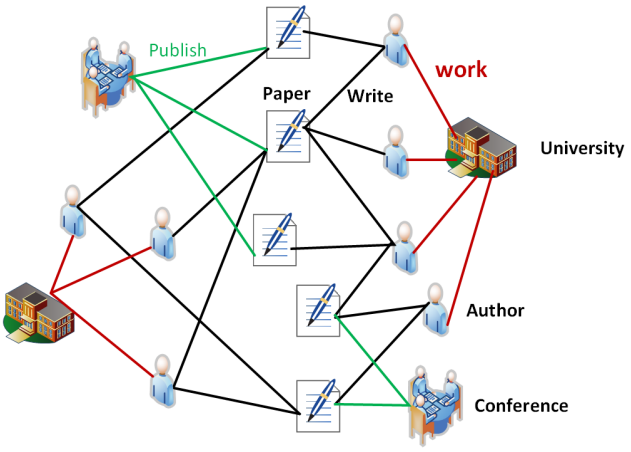


FIGURE 2. The illustration of a Heterogeneous Network. There are four types of nodes: author, paper, university, and conference; also three types of edges: working, publishing and writing.

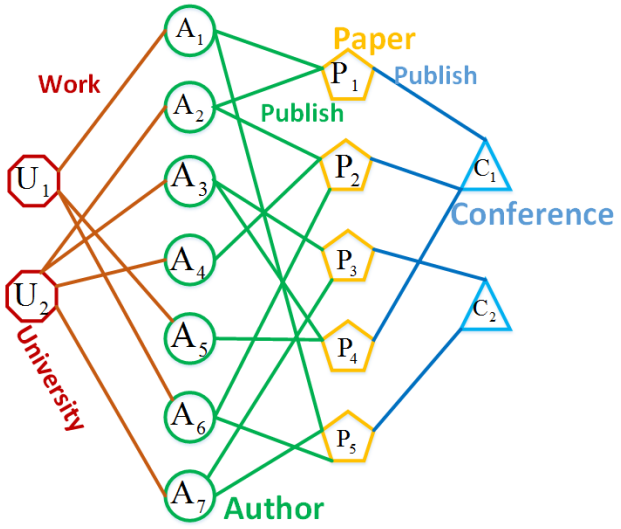


FIGURE 3. An Example of Heterogeneous Network. It is a simplified version of Fig. 2.

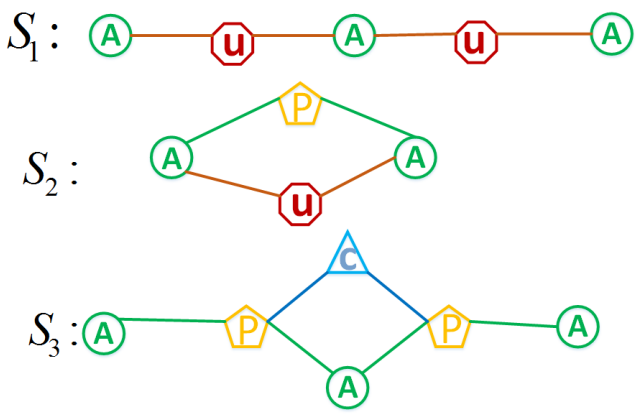


FIGURE 4. An Example of meta-graph. It is the meta-graph of Fig. 3. U:university; A:author; C:conference; P:paper.

data mining tasks, such as classification [1], [2], clustering [3], ranking, recommendation [4], similarity search [5] and prediction and so on.

[Meta-paths and Meta-graphs] One popular tool to mine HNs is the meta-paths, which could better preserve relationship information between different types of entities in HNs in comparison to random walks A meta-path is defined as a sequence of object types. Consider Fig. 2, the meta-path "author-paper-author-paper-author"(abbreviated as "APAPA") conveys the information that both authors coauthor the same paper, while the meta-path "APCPA" indicates both papers are published in the same conference. However, a meta-path itself has limited semantics, since the length of meta-paths is no more than four in most studies [5]. Another problem is that typical meta-path generation can produce many short paths or even isolated nodes which are hard for further process and learning [6]. For example, in Fig. 3, the relationship between a_1 and a_5 is missed when choosing meta-path "APAPA"; similarly, after selecting meta-path "APCPA", we are unable to capture the relation between a_2 and a_6; also a single meta-path cannot express relationships between distant objects. To compensate such weakness of meta-path, the concept of meta-graph is proposed by [7], combining many different meta-paths, and the meta-graph is shown to have a clear advantage in tasks like entity-resolution, ranking and clustering. An example of the meta-graph is illustrated in Fig. 4.

[Weights of meta-graph] Different meta-paths attempt to capture different semantics in HNs. When predicting the label of author a_2 in Fig. 3, $a_2 \rightarrow p_2 \rightarrow a_6$ where two authors co-author the same paper is intuitively more important than $a_2 \rightarrow u_2 \rightarrow a_7$ that two authors work in the same university. Inspired by this, we introduce the weights for the meta-graph in the learning process. Details are presented in Section III-B.

[Network embedding] One ultimate goal of this paper is to learn better node features. Automatic graph representation learning is a research area of long history and is essential

for handling large networks. Previously, matrix factorization techniques like eigen-decomposition of Laplacian matrix [8], and the graph factorization [9], have been well studied. Recently, the applications of neural networks on graph learning are showing strong performance against classic methods [10]–[13]. In particular, this paper employs the graph convolutional network [14]. GCN is an end-to-end semi-supervised neural network model which integrates the links among nodes and prior knowledge to compute its parameters. GCN itself is intended for homogenous network due to its design; therefore we feed GCN with embeddings learned by our weighted meta-graph as prior knowledge. GCN is able to make full use of graph structure information and better characterize a node's neighborhood. It defines a convolution operator and iteratively aggregates the embeddings of neighbors for a node and then further uses its embedding from the previous iteration to obtain the new embedding. In addition, we modify the convolution operator to consider what we named as node ''self-significance'' - how much a node should consider its own feature when convolving with its neighbors through the iterations. Performance improvement is observed in our experiments because of considering this ''self-significance''. Based on our experiments, when combined with the weighted meta-graph, the features learned by our modified GCN are compentent in the tasks of multi-label classification and link prediction, outperforming state-of-the-arts models LINE, node2vec, metapath2vec and HIN2Vec on four data sets DBLP, Aminer, Blogcatalog, and IMDB. Details are presented in Section IV.

Our main contributions to the representation learning of HNs are now summarized as follows:

- To take full advantage of the meta-paths and remedy the gap of different objects, we propose the weighted meta-graph to better capture graph relationship semantics.
- We propose a modified version of GCN to further process the results from the meta-graph to learn HNs embeddings. The weighted meta-graph makes up for the lack of semantics in GCNs, while the GCN mixes the network topological structure information with the semantics.
- Extensive experiments are done to compare with a variety of recent models on four real-world data sets, with different train-test splits, to confirm the effectiveness of our approach in classification and link prediction.

We organize the rest of the paper as follows. We review the related work in Section II, and introduce some definitions and conceptions in Section III. Then we propose our model in Section III and report our experimental results in Section IV. Finally, we make conclusions and discuss future work in Section V.

II. RELATED WORK

Previous works on network embedding and neural networks inspire this research. The following briefly reviews the related work.

A. NETWORK EMBEDDING

Network embedding assigns nodes in a network to low-dimensional representations and effectively preserves the network structure [15]. [16], [17] perform network embedding by matrix factorization. A matrix factorization algorithm represents the links between nodes using adjacency matrix, then obtains the network embedding to factorize adjacency matrix. Although matrix factorization algorithms capture the global structure of the networks, the bottleneck is the decline of performance in facing a large network with millions of nodes. Recently, high-quality features learned by new network embedding methods have been applied in various machine learning algorithms, and have achieved remarkable effects. Compared to the traditional methods, these new network embedding methods avoid the problem of high computational complexity of matrix factorization. These representation learning or embedding models include Deepwalk, Node2vec, subgraph2vec [10], [11], [18]. One common technique of these methods firstly generate node sequences by random walks, and then learn to embed using Skip-Gram. However, these models are all for feature learning from homogeneous networks, and they are not applied in HNs.

HIN2Vec [19] is designed to capture the different semantics between nodes based on meta-paths in heterogeneous information networks. Unfortunately, HIN2Vec does not take into account different weights in meta-paths.

B. NEURAL MODEL ON NETWORKS

Recent graph neural models are showing strong performance in their ability to extract node features and demonstrated ground-breaking performance on many tasks [20]. Reference [21] uses Graph Nueral network and metagraphs to solve the semi-supervised learning in attributed heterogeneous information networks. The definition of metagraphs in [21] is different from us, and they don't give the effect of different metagraphs. Graph Convolutional Networks (GCNs) [14] et al. utilize a first-order approximation of spectral convolutions on the data of the graphs to represent the structure and relationships between nodes. [22] utilizes GCN to embed temporal heterogeneous graph to perform the community detection task.

References [23]–[25] integrate meta paths and attention mechanism to capture the semantic information in HNs. Reference [26] separates the same type of nodes from Heterogeneous Information Networks (HINs), then aggregates the heterogeneous neighbors. R-GCNs [27] and FastGCN [28] are based on GCNs to more efficiently learn hidden layer representations. Graph neural network is another model family. For this paper, we focus on applying GCN to learn from HNs.

III. THE PROPOSED APPROACHES

In this section, firstly, we give our main structure of our model, then we review some basic concepts about HNs, and meta-graph. At last, we introduce the background of the GCNs.

The main structure of our proposed model is as follows:

- Extracting the neighbors' set of given meta graphs, and construct different homogeneous networks according to the types of objects
- According to the weights of different meta graphs to aggregate the homogeneous networks
- Utilizing GCN with node self-significance to embed the heterogeneous networks

A. PRELIMINARIES AND PROBLEM DEFINITION

Definition 1: Heterogeneous Information Network. An heterogeneous network is a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \phi, \psi\}$ with an object type mapping function ϕ: $\mathcal{V} \rightarrow \mathcal{T}$ and a link type mapping function ψ: $\mathcal{E} \rightarrow \mathcal{R}$, where each object v $\in$ V belongs to one particular object type $\phi(v)$ $\in$A, and each link e $\in$E belongs to a particular relation $\psi(e)$ $\in$ $\mathcal{R}$, with $|\mathcal{T}| + |\mathcal{R}| > 2$. $T_G = (\mathcal{T}, \mathcal{R})$ is denoted as the network schema.

Definition 2 ([7] Meta-Graph): A meta-graph $\mathcal{S} = (\mathcal{N}, \mathcal{A}, n_t, n_s)$ where $\mathcal{N} \subseteq \mathcal{V}$ is a set of nodes and $\mathcal{A} \subseteq \mathcal{E}$ is a set of edges, is defined on a HN schema $T_G = (\mathcal{T}, \mathcal{R})$. $\mathcal{S}$ is a directed acyclic graph (DAG), and n_t, n_s are the single source and target node, respectively.

The meta-graphs of Fig.3 are shown in Fig.4. As in [29], only the sub-graphs that have one source node and one target node are meta-graphs. The matrix of meta-graph is the same as definition of meta path.

Definition 3: Matrix of meta-graph. Given a meta-graph $\mathcal{S} = (\mathcal{T}_1\mathcal{T}_2 \ldots \mathcal{T}_l)$, the matrix $\mathcal{M}$ of $\mathcal{S}$ is

$$\mathcal{M} = M_{\mathcal{T}_1\mathcal{T}_2} M_{\mathcal{T}_2\mathcal{T}_3} \ldots M_{\mathcal{T}_{l-1}\mathcal{T}_l} \tag{1}$$

where $M_{\mathcal{T}_i\mathcal{T}_j}$ is the adjacency matrix between type $\mathcal{T}_i$ and $\mathcal{T}_j$.

Taking S_3 in Fig. 3 as an example of computing matrix of meta-graph to better comprehension. Without any loss of generalizations, meta-graph S_3 can be written as **{AP {A;C} PA}**. For meta-graph, computing matrix is as follows:

$$M_{S_3} = M_{AP} \otimes \{M_{PC} \otimes M_{CP} \otimes \{M_{PA} \otimes M_{AP}\}\} \otimes M_{PA} \tag{2}$$

where $\otimes$ represents Hadamard product (element-wise product).

Definition 4: Representation Learning of HNs. The representation learning of a graph $\mathcal{G}$ is a map $\varphi : \mathcal{V} \rightarrow \mathbb{R}^d$ which projects every node $v \in \mathcal{V}$ to a low d-dimensional space $\mathbb{R}^d$, where $d \ll | \mathcal{V} |$.

In this work, we aim to get the utmost out of the complex structure of HNs and multiple semantic relationships between nodes in HNs. In order to get better representation learning of HNs, we face several challenges, as follows:

- Utilize meta-graphs to represent the different effect of each meta path in HINs. The proposed model MGCN is a better designed model to efficient and effective representation the weight of different meta-graph.
- Absence of semantic of GCN in graph. Due to the lack of semantic in original GCN, how to express the complex semantic is critical in HNs.
- Regulation. In order to express the latent vector of each node in HNs, reasonable and effective regularization is required in the process of convolution.

B. WEIGHTED META-GRAPH

As shown above, different meta-graphs have different semantics; thus we believe the meta graph weight learning is important for modeling HNs. Without loss of generality, let θ be the weight of the meta-graph $\mathcal{S}$. Since a meta-graph is the similar structure with the meta path with a single source and target node, we can define the relevance measure based on meta-graph as the PathSim [5]:

$$R(n_s, n_t \mid \mathcal{S}) = \frac{2N_{n_s n_t}}{N_{n_s} + N_{n_t}} \tag{3}$$

where $N_{n_s n_t}$ denotes the number of $\mathcal{S}$ instance between n_s and n_t, N_{n_s} and N_{n_t} are the number of $\mathcal{S}$ between n_s and n_s, n_t and n_t, respectively.

In order to learn the weight θ_s of each meta-graph $s \in \mathcal{S}$, we optimize an objective function $\mathcal{O}$ by training data. The principle of training process is maximizing the correlations of objects that share the same labels and minimizing the correlations of objects that are given different labels:

$$\mathcal{O}(\theta) = \max_{\theta_s} \sum_{x_i, x_j \in \mathcal{V}_\mathcal{T}} \left(sign(x_i, x_j) \sum_s \theta_s R_s(x_i, x_j)\right)^2 - \frac{\lambda}{2}\|\theta\|_2^2 \tag{4}$$

where $\|\cdot\|$ is the $\ell^2 - norm$ and λ is a regularization parameter which applies to balance the objective function. $\mathcal{V}_\mathcal{T} \subset \mathcal{V}$ are labelled nodes set. $sign(x_i, x_j)$ is an indicator function: only x_i and x_j have the same label, the value is 1; otherwise is 0.

Let the partial derivatives of $\mathcal{O}(\theta)$ be 0, then we solve the optimization problem to get θ_s:

$$\theta_s = \frac{1}{\lambda} \sum_{x_i, x_j \in \mathcal{V}_\mathcal{T}} (Sign(x_i, x_j) \cdot R_s(x_i, x_j)) \tag{5}$$

C. GCN WITH NODE SELF-SIGNIFICANCE

Graph convolutional network (GCN) [14], [30] is a popular model in graph learning algorithms that utilize the latest deep-learning technologies. When we talk about convolutions in deep learning, we often consider a 1D sequence, 2D images, or occasionally 3D tensors. However, we could understand all these as special cases of the graph learning problem for regular graphs, where the graph is a line, a 2D grid or a 3D cube lattice; a regular graph is a graph where each vertex has the same number of neighbors. Therefore, the GCN is a generalization of convolutional networks. Every GCN layer is an activation function applied on a linear transformation of the convoluted node features. Given a graph $G = (V, E)$, where $V = v_1, \ldots, v_N$ are the nodes and $E \subseteq V \times V$ are the edges, let A be the $N \times N$ graph adjacency matrix where the *ith* row represents the weighted/unweighted adjacency of v_i, and let $\tilde{A} = A + I$ be the adjacency matrix with "self-loop" for convenience; also let $\tilde{D}$ be the degree matrix with self-loop,

i.e. $\tilde{D}$ is a diagonal matrix where $\tilde{D}(i,i)$ is the degree of v_i plus one. Now a usual normalization of $\tilde{A}$ can be written as $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, where the weight of each edge in $\tilde{A}$ is normalized by the degrees or the two edge nodes. Let X be the node feature matrix where each row is a feature vector, then one graph convolution operator g for GCN can be defined as

$$g \star X := \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X \tag{6}$$

where "$\star$" is conventionally used to denote a convolution operator. Intuitively, it simply computes each node's convoluted feature as the weighted average of neighbors' features and the node's own feature according to the weights in the adjacency matrix $\tilde{A}$. It is not hard to check that two composed convolution operator $g^2 \star X := g \star (g \star X)$ convolute the features from the neighborhood's neighborhood. Finally, let $H^{(i)}, i = 1, 2, \ldots$ be the output of each layer and $H^{(0)} = X$, then one convolutional layer in [14] is defined as

$$H^{(i)} = activation(g \star H^{(i-1)}W^{(i)}) \tag{7}$$

and one typical example of network structure is

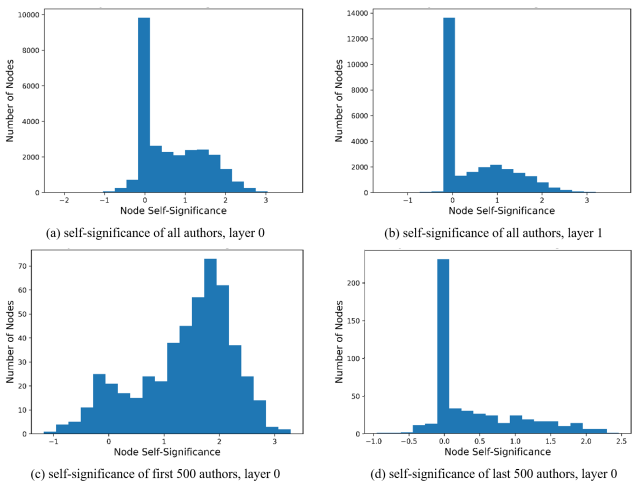
$$Z = softmax(g \star \text{relu}(g \star XW^{(0)})W^{(1)}) \tag{8}$$

where Z gives a probabilistic estimation of the node labels and we can use negative log-likelihood as the loss function. In [14], the convolution operator of Eq. 6 can be formally derived by spectral graph convolutions and its first-order Chebyshev polynomial approximation. An approximation of a higher order gives different convolution operators, but this is out of our scope due to space limitations. We now refine the convolution operator in Eq. 6 by adding what we defined as node significance. Technically, we observe the $\tilde{A} = A + I$ in Eq. 6 does not give any weight to the "self-loop" for the convolution. Intuitively, each node in a network might be more or less "self-significant", i.e. each node's own features play different levels of role in the convolution. The straightforward way to implement this idea is by relaxing the identity matrix I as a trainable diagonal matrix D_s, i.e. redefine "$\tilde{A} := A + D_s$", where the subscript "s" stands for "self-significant". However, this brings computation inconvenience; the normalization cannot be precomputed with the variable D_s in $\tilde{A}$. Therefore, we instead move $\tilde{A}$ outside of the convolution and redefine graph convolution with node significance. In addition, we may allow the ith layer ($i = 0, 1, \ldots$) to have its own $D_s^{(i)}$; consequently, the convolution operator is now different for each layer. As a result, graph convolution with node significance is defined as the following,

$$g_s^{(i)} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}D_s^{(i)} \tag{9}$$

$$H^{(i)} = activation(g_s^{(i)} \star H^{(i-1)}W^{(i)}) \tag{10}$$

For computation convenience, we allow D_s to contain negative numbers; roughly only one in twenty nodes would have negative learned significance. We note that a negative significance has the same meaning as the positive one, e.g. node significance of -3 and 3 both mean a node has relatively higher influence on the convolution, while node significance of -0.1 and 0.1 both mean the node has little influence. To better show the above-defined node significance is making sense, we plot histograms for the trained node significance from a 2-layer convolutional network on the DBLP Author-Author network for node label prediction. The histograms for all nodes are shown in (a), (b). We can see 1) the majority of nodes have zero importance, i.e. their labels can be predicted by just looking at their neighborhoods, but these also exist many nodes with self-significance higher than 1; 2) the second layer has more nodes with near-zero significance; this makes sense because the algorithm looks into a larger local community, and thus reduces more nodes' role in the convolution. We also plot the histogram for the first and last 500 nodes. In the DBLP dataset, authors are ordered by their IDs, and the top authors tend to have more links in the author-author network (i.e. they tend to be influential researchers). We can immediately see top authors tend to have higher self-significance, while the prediction for the last 500 authors is heavily dependent on the neighborhood. We later show in the experiment that adding node self-significance can increase performance by 0.2% to 2%, especially when hide more labels.



(a) self-significance of all authors, layer 0 (b) self-significance of all authors, layer 1

(c) self-significance of first 500 authors, layer 0 (d) self-significance of last 500 authors, layer 0

FIGURE 5. Histograms of learned node self-importance from 2-layer convolutions on the DBLP author-author network for node label prediction. (a), (b) are for all nodes, (c) is for the top 500 authors of smallest author IDs in the DBLP dataset, and (d) is for the last 500 authors of largest author IDs in the DBLP dataset. In DBLP, an author of smaller ID tends to have more co-authorship with other authors.

At last, we apply the following modified 2-layer GCN for our experiments, similar to Eq.8 but now with the modified convolution operators in Eq.9, where $\mathbf{X}$ are features learned from metagraphs, and $g^{(0)} \star XW^{(0)}$ are treated as the learned embeddings.

$$Z = \text{softmax}(g^{(1)} \star \text{relu}(g^{(0)} \star XW^{(0)})W^{(1)}) \tag{11}$$

IV. EXPERIMENT

In this section, we evaluate WMGCN with real-world datasets to verify its effectiveness for representation learning in HNs; the results show our approach outperforms the state-of-the-art methods for classification and link prediction.

TABLE 1. Description of datasets.

Datasets	♯ Nodes	♯ Edges	♯ Node Types	♯ Labels
Blogcatalog	10,351	348460	2	1
DBLP	41,936	534,009	3	2
Aminer	12,522,027	12,518,144	3	2
IMDB	12,890	19,120	3	1

A. EXPERIMENT SETUP

1) DATASETS

The benchmark datasets used in our experiment include: scientific publication datasets (DBLP and Aminder), social network datasets (Blogcatalog and Yelp) and a movie network (IMDB). Basic statistics of these datasets are summarized in Table 1.

DBLP [31] is a bibliographic network in computer science. DBLP contains 28702 authors(A), 20 conferences(C), 13214 terms(T). We employ meta-graph sets $\{A\{C, T\}A\}$, $\{AT\{A; C\}TA\}$ based on the links between nodes.

Blogcatalog is a social network where the relationship is provided by the blogger authors. All bloggers (U) and their groups (G) are nodes, and the edges are friendships (U-U) and users' groups (U-G). In our experiment, we use the meta-graph $\{UGU\}$.

Aminder [32] consists of 9,323,739 authors, 3,194,405 papers and 3,883 venues (held until 2016). It is computer science citation network. It contains papers (P), authors (A) and conferences (C) as nodes and the relationships: writing (A-P), publish (P-C) are used as edges. We employ the meta-graph $\{APA\}$, $\{AT\{A; C\}TA\}$, extracting from Aminer.

IMDB is a movie network among actors(A), directors(D) and movies(M). According to the movies' genre, we divide the movies into actions, comedy and drama. There are two types of edges (A-M,M-D) in the IMDB dataset. Here, we employ the meta-graph sets $\{M\{A, D\}M\}$, $\{MA\{D; M\}AM\}$ to perform our experiments.

2) BASELINES

We compare with recent representation/embedding learning methods for both homogeneous networks and heterogeneous networks to verify the effectiveness of WMGCN.

LINE [12] has two variants: LINE-1st and LINE-2nd. These models sample the edges with the probabilities proportional to their weights, and represents the nodes with vectors in a low-dimension space.

Node2vec [11] is a generalization of deepwalk by introducing two control parameters to tune the balance between two extreme sampling strategies: Breadth-first Sampling (BFS) and Depth-first Sampling (DFS). The sampled random walks are learned by skip-gram.

metapath2vec [13] samples neighborhood sets with meta-paths, and then leverages the negative sampling technique to learn the node features. We use the best performance of the meta paths APTPA and APCPA in DBLP and AMiner respectively, which has the highest weights.

HIN2Vec [19] learns embeddings for HNs. HIN2Vec applies random walks and negative sampling to prepare training data, then applies the logistic binary classifier to predict the relationship between nodes. As HIN2Vec learns representation of meta paths, we can use HIN2Vec to analyze the meta-graph which contains multiple meta-paths.

TABLE 2. Weights of meta-graph in Aminder1.

meta-graph	Weights
APA	$0.005 \sim 0.15$
$AP\{A; C\}PA$	$0.90 \sim 0.98$

TABLE 3. Accuracy results for Aminer1. Comparing the weights of meta-graphs in Aminer1.

meta-graph	Accuracy	Macro-F1
$HIN2Vec(APA)$	0.4984	0.2885
$HIN2Vec(APAPA)$	0.4633	0.2200
$HIN2Vec(APCPA)$	0.6014	0.4457
$WMGCN(0.03\{APA\} + 0.97\{AP\{A; C\}PA\})$	0.6600	0.5147

B. WEIGHT LEARNING FOR META-GRAPH

We first complete a simple experiment to test the necessity of weighted meta-graph in the classification task. Indeed, a higher weight should be assigned to a more effective meta-graph in HNs.

In the experiment, we randomly choose 1500 authors, 500 papers and 10 conferences from the Aminer dataset to calculate the weights for each meta-graph. We denote the sampled dataset as "Aminer1". **{APA, AP {A; C} PA}** are used to construct the meta-graph in our experiment. We learn weights from the labeled data and distribute weights to meta-graph on the basis of the classification performance by the optimization described in Section III-B. In Table 3, we can see that the meta-graph **{AP {A; C} PA}** has the highest classification accuracy. WMGCN learns the subtle change and sets up the highest weight for meta-graph **{AP {A; C} PA}** in Table 2; WMGCN also assigns the smallest weight to meta-graph **{APA}** due to the lowest accuracy with **{APA}**.

From above results, we can see the effectiveness of the learned meta-graph weights. Again, the meta-graph has richer semantics, and the semantics of meta-graph are indirectly mapped to learn the weights of meta-graphs. Therefore, our approach learns the weights of meta-graphs that should play a more significant role in classification. The learned weights are highly consistent with intuition, and the meta-graphs combined through weights in turn improve the prediction accuracy.

C. MULTI-LABEL CLASSIFICATION OF NODES

Node classification in the graph setting arises in many real world-networks, such as identifying people of the same "interest" in a social network. Being able to label a particular entity in a graph based on its nearby graph structure and predicting relationships between entities play an important role in graph analytics [33].

To verify our proposed model WMGCN for graph representation learning, we perform experiments on various datasets to compare with other recent models. Similar experiment setup and datasets can be found in [10]. In all

TABLE 4. Multi-label classification in DBLP. The highest performance are in bold.

	$\%Labeled-Nodes$	9%	8%	7%	6%	5%	4%	3%	2%	1%
accuracy	LINE-1st	0.6158	0.5916	0.5677	0.5379	0.5121	0.5042	0.3783	0.2944	0.2984
	LINE-2nd	0.6579	0.6548	0.6494	0.6454	0.6398	0.6316	0.6255	0.6109	0.5868
	node2vec	0.7368	0.7339	0.7232	0.7100	0.7052	0.6989	0.6891	0.6841	0.6694
	metapath2vec	0.8658	0.8709	0.8796	0.8780	0.8782	0.8787	0.8749	0.8434	0.5079
	WMGCN-1st	**0.9632**	**0.9644**	0.9631	**0.9618**	0.9589	0.9569	0.9540	0.9489	0.9508
	WMGCN-2nd	0.9630	0.9640	**0.9639**	**0.9647**	**0.9649**	**0.9612**	**0.9582**	**0.9565**	**0.9585**
Macro-F1	LINE-1st	0.5675	0.5427	0.5128	0.4686	0.4365	0.4168	0.2473	0.1148	0.1149
	LINE-2nd	0.6336	0.6364	0.6314	0.6245	0.6181	0.6123	0.6038	0.5879	0.5702
	node2vec	0.7221	0.7216	0.7100	0.6951	0.6889	0.6832	0.6751	0.6695	0.6572
	metapath2vec	0.8575	0.8629	0.8746	0.8729	0.8718	0.8717	0.8673	0.8356	0.4263
	WMGCN-1st	**0.9602**	**0.9626**	0.9616	0.9601	0.9571	0.9552	0.9521	0.9472	0.9495
	WMGCN-2nd	**0.9602**	0.9624	**0.9620**	**0.9625**	**0.9631**	**0.9593**	**0.9557**	**0.9540**	**0.9541**

TABLE 5. Multi-label classification in Blogcatalog. The highest performance are in bold.

	$\%Labeled-Nodes$	90%	80%	70%	60%	50%	40%	30%	20%	10%
accuracy	LINE-1st	0.2162	0.2065	0.1929	0.1912	0.1787	0.1621	0.1407	0.1153	0.1045
	LINE-2nd	0.2298	0.2221	0.2173	0.2233	0.2214	0.2098	0.2036	0.1878	0.1438
	node2vec	0.2532	0.2455	0.2364	0.2345	0.2301	0.2106	0.1940	0.1620	0.1129
	metapath2vec	0.6241	0.6245	0.6149	0.6106	0.6098	0.5993	0.5900	0.5659	0.4918
	WMGCN-1st	0.6641	0.6488	0.6444	0.6422	0.6358	0.6176	0.6036	0.5729	0.5088
	WMGCN-2nd	**0.6736**	**0.6572**	**0.6531**	**0.6547**	**0.6502**	**0.6350**	**0.6196**	**0.5949**	**0.5293**
Macro-F1	LINE-1st	0.0722	0.0688	0.0639	0.0589	0.0487	0.0373	0.0235	0.0103	0.0049
	LINE-2nd	0.0747	0.0713	0.0726	0.0722	0.0725	0.0662	0.0622	0.0531	0.0215
	node2vec	0.0894	0.0878	0.0870	0.0834	0.0806	0.0699	0.0593	0.0400	0.0097
	metapath2vec	0.4623	0.4655	0.4587	0.4310	0.4302	0.3948	0.3452	0.2631	0.2043
	WMGCN-1st	0.5668	0.5264	0.5305	0.5209	0.5075	0.4915	0.4735	0.4296	0.3414
	WMGCN-2nd	**0.5735**	**0.5350**	**0.5399**	**0.5321**	**0.5224**	**0.5085**	**0.4927**	**0.4508**	**0.3620**

comparisons, we vary the train-test split and randomly sample the training data to evaluate how well the embeddings learned by our proposed WMGCN. We repeat each confirmation results ten times, and report the average performance in terms of accuracy and Macro-F1. We choose the same parameters for random-walk based methods 1) all random walks have length 80; 2) the skip-gram context window size is 10, and negative sample size is 5. For all models, the embedding dimension is 128. WMGCN-1st uses the original design of 2-layer GCN by equation 8, and WMGCN-2nd applies the modified GCN with node self-significance as by equation 11.

1) DBLP

For this dataset, we follow [10] and vary the training data from 1% to 9% to predict the authors' label. Our weighted meta-graphs learn author features from the heterogeneous relationship among authors, terms and conference, and then GCN predicts author labels based on co-authorship, The results are presented in Table 4. LINE-1st, LINE-2nd and node2vec directly run on the co-author network; as algorithms don't intend for heterogenous networks, their accuracy and Macro-F1 are all no higher than 70%. The accuracy of Metapath2vec can achieve 87%, as it makes use of its sampled meta paths that embed information of different types of relationships. By Table 4, WMGCN-1st outperforms the baselines by 10%, and WMGCN-2nd gains up to 1% additional accuracy. Despite of very limited number of labeled nodes for training, our WMGCN models stably achieve stronger performance,

2) BLOGCATALOG

We vary the training set ratio from 90% to 10% for this dataset. By Table 5, this is a dataset where LINE and node2vec poorly perform, nearly 30% $\sim$ 40% less prediction accuracy than models particularly designed for HNs. The performance of WMGCN models still stably exceed the metapth2vec model by 1% to 2%, and WMGCN-2nd is the best in all cases, and the gap is more noticeable when the training set is small.

3) AMINER

Aminer is a relatively larger heterogeneous network and it is closer to the real-world situation with a large number of edges of different relation types. This is the largest data set we experiment for this paper. We also vary the Aminer training set ratio from 90% to 10%. Table 6 presents the results, which are consistent with the previous two experiments. Even when only 10% of the labeled data is used for training, WMGCN-1st can achieve as high as 92% in accuracy, 20%, 10% more than node2vec and LINE respectively. WMGCN-2nd still performs better than WMGCN-1st, especially when the training set is small, consistent with DBLP and Blogcatalog. We suspect it is because the meta-graph already encodes rich information in the features, and when fewer labeled nodes are given, the prediction starts to rely more on the node's own features rather than the neighbors' features.

4) IMDB

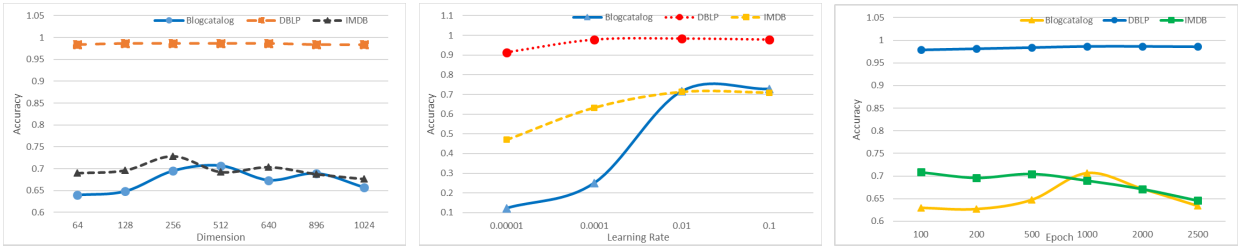
IMDB is a movie network. In this experiment, we also vary the training ratio from 90% to 10%. Table 7 lists the results. The performances of all models improve when given a larger training ratio. Especially, when the ratio of the labels is 10%, WMGCN-1st improves by 14% against metapath2vec which is the best baselines in our experiment.

TABLE 6. Multi-label classification in Aminer. The highest performance are in bold.

	$\%Labeled-Nodes$	90%	80%	70%	60%	50%	40%	30%	20%	10%
accuracy	LINE-1st	0.9002	0.9010	0.9005	0.8998	0.8967	0.8912	0.8883	0.8867	0.8844
	LINE-2nd	0.9114	0.9110	0.9102	0.9067	0.9035	0.9017	0.8994	0.8982	0.8975
	node2vec	0.6915	0.6923	0.6921	0.6917	0.6909	0.6905	0.6886	0.6822	0.6818
	metapath2vec	0.9329	0.9325	0.932	0.9307	0.9295	0.9285	0.9254	0.9219	0.9184
	WMGCN-1st	**0.9474**	**0.9463**	0.9458	0.9404	0.9377	0.9325	0.9279	0.9227	0.9196
	WMGCN-2nd	0.9470	0.9460	**0.9463**	**0.9415**	**0.9392**	**0.9341**	**0.9297**	**0.9256**	**0.9243**
Macro-F1	LINE-1st	0.8912	0.8910	0.8904	0.8896	0.8871	0.8854	0.8834	0.8815	0.8795
	LINE-2nd	0.9052	0.9047	0.9035	0.9022	0.9014	0.9002	0.8983	0.8974	0.8946
	node2vec	0.7108	0.7111	0.7103	0.7085	0.7077	0.7079	0.7028	0.6964	0.6958
	metapath2vec	0.9344	0.9340	0.9337	**0.9312**	**0.9295**	**0.9267**	**0.9252**	**0.9223**	**0.9208**
	WMGCN-1st	0.9338	0.9324	0.9302	0.9247	0.921	0.9184	0.9151	0.9115	0.9098
	WMGCN-2nd	**0.9322**	**0.9305**	**0.9274**	0.9213	0.9188	0.9146	0.9139	0.9117	0.9112

TABLE 7. Multi-label classification in IMDB. The highest performance are in bold.

	$\%Labeled-Nodes$	90%	80%	70%	60%	50%	40%	30%	20%	10%
accuracy	LINE-1st	0.5833	0.5801	0.5812	0.5901	0.5830	0.5800	0.5730	0.5602	0.4438
	LINE-2nd	0.5872	0.6000	0.5897	0.5929	0.5978	0.5967	0.5838	0.5677	0.5489
	node2vec	0.5628	0.5705	0.5791	0.5830	0.5832	0.5762	0.5675	0.5504	0.5220
	metapath2vec	0.6064	0.6244	0.6205	0.6147	0.6078	0.5978	0.5884	0.5701	0.5464
	WMGCN-1st	**0.6987**	0.7064	0.7047	0.7035	0.7033	0.7027	**0.7031**	0.7001	0.6933
	WMGCN-2nd	0.6701	**0.7102**	**0.7116**	**0.7118**	**0.7141**	**0.7135**	**0.7031**	**0.7125**	**0.7053**
Macro-F1	LINE-1st	0.4492	0.4297	0.4232	0.4317	0.4242	0.4204	0.4161	0.4037	0.2314
	LINE-2nd	0.5317	0.5540	0.5321	0.5275	0.5296	0.5270	0.5013	0.4574	0.4012
	node2vec	0.4435	0.4373	0.4417	0.4392	0.4356	0.4244	0.4142	0.4017	0.3816
	metapath2vec	0.5370	0.5657	0.5572	0.5477	0.5490	0.5275	0.4965	0.4521	0.4066
	WMGCN-1st	0.6756	0.6825	0.6797	0.6771	0.6795	0.6799	**0.6792**	**0.6763**	0.6689
	WMGCN-2nd	**0.6761**	**0.6844**	**0.6860**	**0.6842**	**0.6914**	**0.6903**	**0.6792**	0.6170	**0.6799**



(a) Dimension (b) Learning Rate (c) Epoch

FIGURE 6. Impact of parameters on WMGCN.

Besides, Macro-F1 score of WMGCN-1st is up to 26% than metapath2vec only with 1% training data. However, WMGCN-2nd improves 1% than WMGCN-1st on average, which demonstrates the effectiveness of the node self-significance. In other training ratios, WMGCN-1st performs better than other baselines from 8% to 20% in the accuracy, and Macro-F1. In summary, when only few labels are given, our proposed model WMGCN can also have a strong performance in social networks, citation networks and movie networks. Not only because WMGCN can extract complicate relationships and weight the meta-graph, but also the improved deep convolutional model can better capture the deep feature.

D. PARAMETER SENSITIVITY

Finally, we evaluate the impact of parameters in the following categories on classification tasks. In addition, similar impact of parameters can be observed in other tasks.

1) EMBEDDING DIMENSIONALITY

Fig. 6(a) examines the performance of WMGCN via changing the number of dimensions d in the different reality HNs.

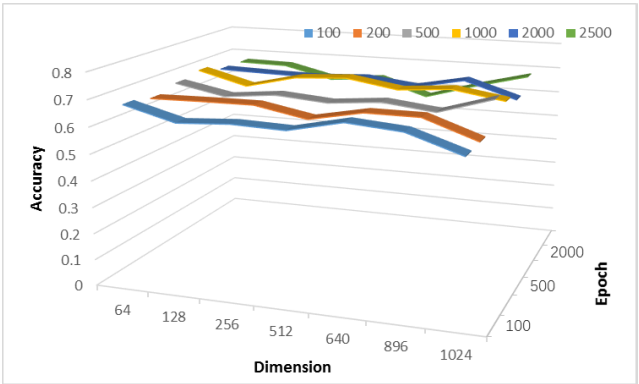
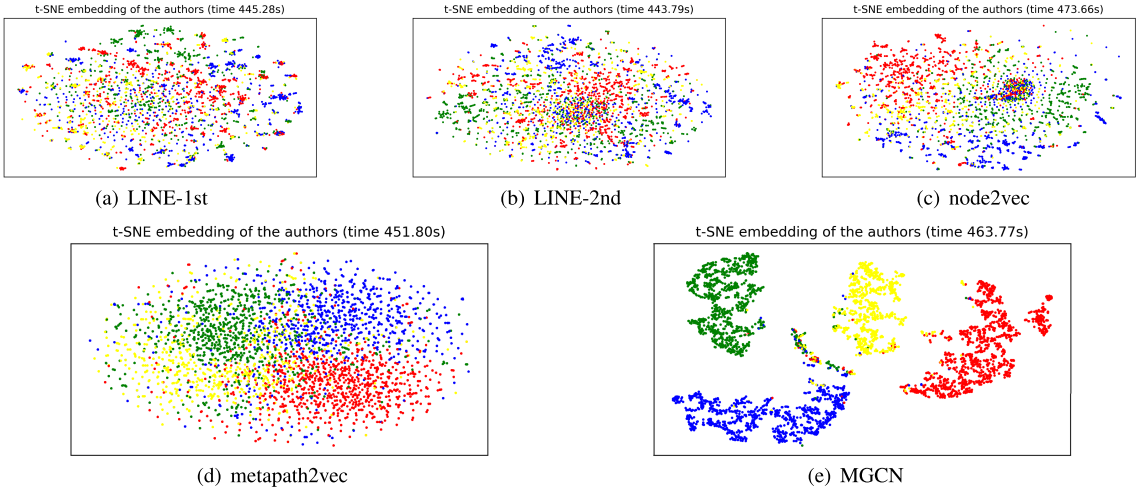


FIGURE 7. Effect of Accuracy and Dimension on Blogcatalog.

All datasets are not sensitive to the embedding dimensionality. What is more, the optimal results can be achieved when d is 256 and 512 in IMDB and Blogcatalog, respectively.

We also compare the effects of varying the epoch and the embedding dimensionality at the same time. Fig. 7 shows that the performance of different various sizes stay stable no matter how changes the epoch. The experimental results



(a) LINE-1st (b) LINE-2nd (c) node2vec

(d) metapath2vec (e) MGCN

FIGURE 8. Visualization of DBLP Coauthor Network. Color indicates the category of author.

show that WMGCN can not only perform better than various models but also be stable facing varieties of parameters.

2) LEARNING RATE

Furthermore, we demonstrate the effects of learning rate in various HNs. The experimental results can be shown in Fig. 6(b). In the beginning, the effects of the WMGCN vary significantly as increasing the learning rate θ. When $\theta > 0.001$, the effect is stable.

3) EFFECT OF EPOCH

We increase the number of epoch over 100, 200, 500, 1000, 2000, 2500 on datasets: DBLP, Blogcatalog, IMDB. From Fig. 6(c), the accuracy of Blogcatalog dataset changes better until up to the number of 1000. The best results can be achieved at the number of 1000 epoch.

To sum up, as the number increases, the results change better. However, after epoch number is up to some threshold, the performance is worse because of the overfitting. Furthermore, the performance of WMGCN is stable when the number of epoch is changing.

E. VISUALIZATION OF DBLP COAUTHOR NETWORK EMBEDDING

Embedding visualization like t-SNE is a useful tool to demo embedding quality. Due to space limitations, we only visualize the DBLP co-author network embeddings. For this experiment, as shown in Table 4, the performance improvement by our WMGCN is outstanding, and we expect the visualization of the embeddings learned by WMGCN are distinct from others as well. The results for the case of 9% training ratio is shown in Fig. 8. For LINE and node2vec, a large number of embeddings from different classes are pretty mixed. For metapath2vec, the embeddings are clearly more distinguishable, and such improvement is facilitated by better use of network semantics. Finally, the embeddings learned by our proposed WMGCN are well separated by class, except for only a few mixes in the middle and several exceptions lying on the periphery of each class. The visualization results well support our superior performance in Table 4.

TABLE 8. Area Under Curve (AUC) scores for link prediction. Comparison with popular embedding models using binary operators: Average, Hadamard, Weighted-L1, and Weighted-L2 to merge node embeddings as the edge embedding.

Binary	Model	DBLP	IMDB	Blogcatalog
Average	LINE-1st	0.7210	0.7405	0.9151
	LINE-2nd	0.8214	0.7538	0.9138
	node2vec	0.6958	0.6906	0.8869
	metapath2vec	0.7150	0.6435	0.8416
	WMGCN-1st	0.8222	0.7034	0.9289
	WMGCN-2nd	**0.8545**	**0.7755**	**0.9425**
Hadamard	LINE-1st	0.9541	0.9065	0.8212
	LINE-2nd	0.9694	0.9003	0.8369
	node2vec	0.8975	0.9056	0.8933
	metapath2vec	0.8983	0.8741	0.8541
	WMGCN-1st	0.9898	0.9162	0.9120
	WMGCN-2nd	**0.9934**	**0.9165**	**0.9157**
Weight-L1	LINE-1st	0.9842	0.7774	0.5939
	LINE-2nd	0.8431	0.6676	0.6016
	node2vec	0.8942	0.7006	0.7368
	metapath2vec	0.8779	0.7722	0.7914
	WMGCN-1st	**0.9962**	0.8021	0.8443
	WMGCN-2nd	**0.9962**	**0.8070**	**0.8482**
Weight-L2	LINE-1st	0.7210	0.7882	0.5933
	LINE-2nd	0.8444	0.6675	0.7028
	node2vec	0.8659	0.7043	0.8473
	metapath2vec	0.8823	0.7425	0.8468
	WMGCN-1st	0.9933	0.8041	0.9092
	WMGCN-2nd	**0.9951**	**0.8150**	**0.9137**

F. LINK PREDICTION

Link prediction is another important task for graph learning. As the term suggests, it predicts if there is an edge between a pair of nodes, given information (like features) of both nodes. Link prediction is applicable to a wide variety of application areas, such as building recommendation systems [34], co-authorship prediction [35], and protein-protein interaction (PPI) prediction [36].

To further evaluate the quality of learned node embeddings by our proposed model WMGCN, we experiment the feature-based link prediction on three datasets DBLP, IMDB and Blogcatalog. Let $\mathbf{u}$ and $\mathbf{v}$ represent the node feature vectors for two nodes a, b then the embedding for a node pair (a, b) can be calculated in several ways according to [11], as shown below. Then a logistic regression model is

applied on the edge embeddings to predict the existence of edge between a, b

- The average of two features: $\frac{\mathbf{u}+\mathbf{v}}{2}$
- The Hadamard product (element-wise product) of two node features: $\mathbf{u} * \mathbf{v}$
- The Weighted-L1: $|\mathbf{u} - \mathbf{v}|$
- Weighted-L2: $|\mathbf{u} - \mathbf{v}|^2$

The Area Under Curve(AUC) scores for link prediction are summarized in Table 8. From the results, we can see that WMGCN-1st outperforms most of the baselines scores. When we add node self-significance, WMGCN-2nd slightly gains some additional scores. For DBLP dataset, WMGCN models outperform the best baseline by around 10% for Weight-L1. For the other two datasets, the performance of WMGCN models increase about 4%. The results of link prediction again confirms the high-quality embeddings learned by our proposed WMGCN.

V. CONCLUSION

In this paper, we proposed a novel framework WMGCN for the representation learning of heterogeneous networks. Previous models do not consider the different levels of effects in the meta-graph. To compensate, we make use of the labeled data to quantify the weights of each meta-graph that express the significance each meta-graph should play in a graph learning task. After that, we combine the meta-graph with a graph convolutions, where meta-graph makes up the GCN's inability to capture graph semantics in a heterogeneous network, and GCN in turn help mix and propagate metgraph features among close nodes by convolution operations. In addition, we propose to consider node significance for the graph convolution operator and show such modification can further improve performance.

Experiments on various real-world datasets illustrate the effectiveness of our proposed representation learning approach for the tasks of classification and link prediction. As future work, we could experiment the learned embedding on other tasks like network alignment. In this work, the complex semantics of HNs are mostly captured by the meta-graphs. Considering the outstanding performance of convolutional networks, we would like to consider more improvement of graph convolutions to learning from HNs.

REFERENCES

[1] C. Wan, X. Li, B. Kao, X. Yu, Q. Gu, D. Cheung, and J. Han, "Classification with active learning and meta-paths in heterogeneous information networks," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2015, pp. 443–452.

[2] J. Zhang, Z. Jiang, and T. Li, "CHIN: Classification with META-PATH in heterogeneous information networks," in *Proc. Int. Conf. Appl. Informat.* Cham, Switzerland: Springer, 2018, pp. 63–74.

[3] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 17–37, Jan. 2017.

[4] B. Hu, C. Shi, W. X. Zhao, and T. Yang, "Local and global information fusion for top-N recommendation in heterogeneous information network," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2018, pp. 1683–1686.

[5] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "PathSim: Meta path-based top-K similarity search in heterogeneous information networks," *Proc. VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[6] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "MetaGraph2Vec: Complex semantic path augmented heterogeneous network embedding," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2018, pp. 196–208.

[7] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li, "Meta structure: Computing relevance in large heterogeneous information networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2016, pp. 1595–1604.

[8] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, vol. 14, no. 6, pp. 585–591.

[9] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, 2013, pp. 37–48.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2014, pp. 701–710.

[11] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2016, pp. 855–864.

[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web (WWW)*, 2015, pp. 1067–1077.

[13] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2017, pp. 135–144.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907

[15] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2019.

[16] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proc. 10th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2017, pp. 731–739.

[17] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation," in *Proc. IJCAI*, 2016, pp. 3889–3895.

[18] A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Saminathan, "Subgraph2vec: Learning distributed representations of rooted subgraphs from large graphs," 2016, *arXiv:1606.08928*. [Online]. Available: http://arxiv.org/abs/1606.08928

[19] T.-Y. Fu, W.-C. Lee, and Z. Lei, "HIN2Vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2017, pp. 1797–1806.

[20] J. Zhou, G. Cui, Z. Zhang, C. Yang, and M. Sun, "Graph neural networks: A review of methods and applications," 2018, *arXiv:1812.08434*. [Online]. Available: https://arxiv.org/abs/1812.08434

[21] A. Sankar, X. Zhang, and K. C.-C. Chang, "Meta-GNN: Metagraph neural network for semi-supervised learning in attributed heterogeneous information networks," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2019, pp. 137–144.

[22] Y. Zheng, S. Chen, X. Zhang, and D. Wang, "Heterogeneous graph convolutional networks for temporal community detection," 2019, *arXiv:1909.10248*. [Online]. Available: http://arxiv.org/abs/1909.10248

[23] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 2022–2032.

[24] S. Wang, Z. Chen, D. Li, Z. Li, L.-A. Tang, J. Ni, J. Rhee, H. Chen, and P. S. Yu, "Attentional heterogeneous graph neural network: Application to program reidentification," in *Proc. SIAM Int. Conf. Data Mining*, 2019, pp. 693–701.

[25] Y. Wang, Z. Duan, B. Liao, F. Wu, and Y. Zhuang, "Heterogeneous attributed network embedding with graph convolutional networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 10061–10062.

[26] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2019, pp. 793–803.

[27] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. V. D. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.*, 2017, pp. 593–607.

[28] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," 2018, *arXiv:1801.10247*. [Online]. Available: https://arxiv.org/abs/1801.10247

[29] L. Sun, L. He, Z. Huang, B. Cao, C. Xia, X. Wei, and P. S. Yu, "Joint embedding of meta-path and meta-graph for heterogeneous information networks," in *Proc. IEEE Int. Conf. Big Knowl. (ICBK)*, Nov. 2018, pp. 131–138.
[30] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[31] Y. Sun, J. Han, J. Gao, and Y. Yu, "ITopicModel: Information network-integrated topic modeling," in *Proc. 9th IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 493–502.
[32] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 990–998.
[33] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social Network Data Analytics*. Boston, MA, USA: Springer, 2011, pp. 115–148.
[34] Z. Huang, X. Li, and H. Chen, "Link prediction approach to collaborative filtering," in *Proc. 5th ACM/IEEE-CS Joint Conf. Digit. Libraries (JCDL)*, 2005, pp. 141–142.
[35] M. Pavlov and R. Ichise, "Finding experts by link prediction in co-authorship networks," in *Proc. Int. Conf. Finding Experts Web Semantics*, 2007, pp. 42–55.
[36] J. Shen, J. Zhang, X. Luo, W. Zhu, K. Yu, K. Chen, Y. Li, and H. Jiang, "Predicting protein-protein interactions based only on sequences information," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 11, pp. 4337–4341, Mar. 2007.

JINLI ZHANG is currently pursuing the Ph.D. degree in computer science and technology with the Beijing University of Technology, China. She is also a joint Ph.D. Student with Drexel University. Her research interests include artificial intelligence in biomedical engineering, data mining and knowledge discovery, and machine learning.

ZONGLI JIANG is currently a Professor with the College of Computer Science, Beijing University of Technology. His major research interests include trusted web security, data mining and knowledge discovery, and machine learning.

ZHENG CHEN is currently pursuing the Ph.D. degree with the College of Computer and Informatics, Drexel University. He is also an enter of Amazon. His research interests include pattern recognition, machine learning, and data mining and knowledge discovery.

XIAOHUA HU joined Drexel University, in 2002. He is currently a Full Professor and the Founding Director of the Data Mining and Bioinformatics Lab, College of Computing and Informatics (the former College of Information Science and Technology, one of the best information science schools in USA, ranked as #1 in 1999 and #6 in 2010 in information systems by U.S. News & World Report). He is also serving as the founding Co-Director of the NSF Center (I/U CRC) on Visual and Decision Informatics (NSF CVDI), the IEEE Computer Society Bioinformatics and Biomedicine Steering Committee Chair, and the IEEE Computer Society Big Data Steering Committee Chair. He is also a Scientist, a Teacher, and an Entrepreneur. He founded the *International Journal of Data Mining and Bioinformatics* (SCI indexed) in 2006. He has published more than 280 peer-reviewed research articles (Google citation more than 20000) in various journals, conferences and books, such as various the IEEE/ACM TRANSACTIONS (IEEE/ACM TCBB, IEEE TFS, IEEE TDKE, IEEE TITB, IEEE SMC, IEEE Computer, IEEE NanoBioscience, and IEEE INTELLIGENT SYSTEMS), JIS, KAIS, CI, DKE, IJBRA, SIG KDD, IEEE ICDM, IEEE ICDE, SIGIR, ACM CIKM, IEEE BIBE, IEEE CICBC, and so on, and the Co-Edited 20 books/proceedings. His current research interests are in big data, data/text/web mining, bioinformatics, information retrieval and information extraction, social network analysis, and healthcare informatics. He has received a few prestigious awards, including the 2005 National Science Foundation (NSF) Career Award, the Best Paper Award at the 2007 International Conference on Artificial Intelligence, the Best Paper Award at the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, the 2010 IEEE Granular Computing Outstanding Contribution Awards, the 2007 IEEE Bioinformatics and Bioengineering Outstanding Contribution Award, the 2006 IEEE Granular Computing Outstanding Service Award, and the 2001 IEEE Data Mining Outstanding Service Award. He has also served as a Program Co-Chair/Conference Co-Chair of 14 international conferences/workshops and a Program Committee Member in more than 80 international conferences in the above areas. He is also the founding Editor-In-Chief of the *International Journal of Data Mining and Bioinformatics* (SCI indexed) and *International Journal of Granular Computing, Rough Sets and Intelligent Systems*, and an Associate Editor/Editorial Board Member of four international journals (KAIS, IJDWM, IJSOI, and JCIB). His research projects are funded by the National Science Foundation (NSF), the U.S. Department of Education, the PA Department of Health and Industries Labs. He has obtained more than US$9.0 million research grants in the past 12 years as PI or Co-PI (PIs of nine NSF grants, PI of one IMLS grant in the last ten years).