HGATE: Heterogeneous Graph Attention Auto-Encoders

Wei Wang, Xiaoyang Suo, Xiangyu Wei, Bin Wang, Hao Wang, *Senior Member, IEEE*, Hong-Ning Dai, *Senior Member, IEEE*, and Xiangliang Zhang, *Senior Member, IEEE*

Abstract—Graph auto-encoder is considered a framework for unsupervised learning on graph-structured data by representing graphs in a low dimensional space. It has been proved very powerful for graph analytics. In the real world, complex relationships in various entities can be represented by heterogeneous graphs that contain more abundant semantic information than homogeneous graphs. In general, graph auto-encoders based on homogeneous graphs are not applicable to heterogeneous graphs. In addition, little work has been done to evaluate the effect of different semantics on node embedding in heterogeneous graphs for unsupervised graph representation learning. In this work, we propose a novel Heterogeneous Graph Attention Auto-Encoders (HGATE) for unsupervised representation learning on heterogeneous graph-structured data. Based on the consideration of semantic information, our architecture of HGATE reconstructs not only the edges of the heterogeneous graph but also node attributes, through stacked encoder/decoder layers. Hierarchical attention is used to learn the relevance between a node and its meta-path based neighbors, and the relevance among different meta-paths. HGATE is applicable to transductive learning as well as inductive learning. Node classification and link prediction experiments on real-world heterogeneous graph datasets demonstrate the effectiveness of HGATE for both transductive and inductive tasks.

Index Terms—Graph embedding representation, heterogeneous graphs, hierarchical attention, transductive learning, inductive learning

1 Introduction

There are many graph structures in the real world, such as social networks, telecommunication networks, citation networks, and biological networks. Graph representation learning is one of the most widely-used graph analysis methods. It has been widely applied in various tasks, such as node classification [1], node clustering [2], link prediction [3],[4], community detection [5], entity alignment [6],graph classication [7] and recommendation system [8].

Some powerful graph representation learning methods, such as Graph Convolution Networks [9] and Graph Attention Networks [10], are supervised methods that depend on data label information.

In real-world applications, however, it is not easy to appropriately and precisely label a large number of nodes and obtain a high-quality labeled data set. On the one hand, the type of label is difficult to determine. On the other hand, the labeling process costs a lot of manpower and material resources. As a robust unsupervised graph representation learning method, graph auto-encoders [11] avoid the problem of node labeling and thus has been a widely studied topic. Some graph auto-encoders only use graph structure information for node embedding [12], while others use both graph structure information and node attributes that are applicable to attributed networks [13], [14], [15], [16]. Graph attention auto-encoders [17] employ attention mechanisms to aggregate neighbor nodes' features to get node representation that assigns large weights to more important nodes, so as to improve the learning performance.

Most existing graph auto-encoders are based on *homogeneous* graphs, which are not applicable to *heterogeneous* graphs that contain rich semantic information. Heterogeneous graphs contains different types of nodes and edges. Homogeneous graphs which consist of only one type of nodes and edges can be described by first order, second order [18],[19] or community structures[20], but the structure in heterogeneous graphs is usually semantic dependent, such as meta-path structure, meta-graph structure[21], implying that the local structure of one node in heterogeneous graphs can be very different described when considering different types of relations, we thus need different methods to preserve the complex structures; Different types of nodes or edges have different meaning, and the importance of different edges or

- *Wei Wang, Xiaoyang Suo, and Xiangyu Wei are with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China. E-mail: {wangwei1, 17120481, 16120338}@bjtu.edu.cn.*
- *Bin Wang is with the Zhejiang Key Laboratory of Multi-dimensional Perception Technology, Application and Cybersecurity, Hangzhou 310053, China. E-mail: bin_wang@zju.edu.cn.*
- *Hao Wang is with the Research Center for Optical Fiber Sensing, Zhejiang Laboratory, 310000 Hangzhou, China, and also with the School of Cyber Engineering, Xidian University, Xi'an 710000, China. E-mail: hawa@ntnu.no.*
- *Hong-Ning Dai is with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong, China. E-mail: hndai@ieee.org.*
- *Xiangliang Zhang is with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA. E-mail: xzhang33@nd.edu.*

Manuscript received 12 Oct. 2020; revised 22 Sept. 2021; accepted 12 Dec. 2021. Date of publication 28 Dec. 2021; date of current version 7 Mar. 2023.
The work was supported in part by the National Key R&D Program of China under Grant 2020YFB2103802, in part by the National Natural Science Foundation of China under Grant U21A20463, in part by the Fundamental Research Funds for the Central Universities under Grant KKJB320001536, and in part by Macao Science and Technology Development Fund under Macao Funding Scheme for Key R & D Projects under Grant 0025/2019/AKP, and in part by Research Initiation Project of Zhejiang Lab under Grant 113012-PI2013.
(Corresponding authors: Xiangyu Wei and Bin Wang.)
Recommended for acceptance by P. Tsaparas.
Digital Object Identifier no. 10.1109/TKDE.2021.3138788

nodes is different. Therefore, it is necessary to learn the importance of these kinds of relationships between users. And different types of nodes and edges have different attributes, which are usually located in different feature spaces, and thus when designing heterogeneous graph embedding methods, especially heterogeneous graph neural networks, we need to overcome the heterogeneity of attributes to fuse information [22].

Heterogeneous graphs represent the complex graph structure of multi-type objects and links in the real world, and thus contain comprehensive information and rich semantics among the objects. Heterogeneous graphs have been widely studied in graph analysis and data mining tasks.

A *meta-path* is a composite relation path between two objects, which is usually used to represent the multi-types of a semantic relation [23]. For example, in citation networks, two authors' relationships can be represented by a meta-path *Author-Paper-Conference-Paper-Author* which implies that the two authors published the paper in the same conference and a meta-path *Author-Paper-Author* which describes co-author relationship. The different meta-paths can have different semantics. We can use a meta-path to describe a composite relation or the high-order similarity of two nodes.

It is a big challenge to embed the heterogeneous graph by considering the comprehensive and different heterogeneous graph information, including node attributes, graph structure and semantic information.

Liu *et al.* [24] used graph structure features to embed the heterogeneous graph for semantic proximity search. Chang *et al.* [25] and Peng *et al.* [26] used both network structure features and node attributes for heterogeneous graph embedding. Wang *et al.* [27] used attention mechanisms to assign different importance to different types of nodes in a neighborhood. However, this work ignored the importance of different meta-paths. Wang *et al.* [28] employed hierarchical attention mechanisms to learn not only the relevance between a node and its meta-path based neighbors, but also the relevance among different meta-paths. However, all the above methods are supervised learning methods. They are not applicable to unsupervised heterogeneous graph embedding in scenarios where labelling information is unavailable.

Most previous related work focused on generating node representation of a single fixed graph. However, some real-world applications require to quickly generate the representation of nodes that have not appeared before. Compared to transductive learning, inductive learning is particularly difficult to conduct because the training models should be suitable for unseen nodes. There are some graph embedding methods involved in inductive learning based on homogeneous graphs [29], [30]. However, they are not suitable for heterogeneous graphs.

In this work, we propose a novel heterogeneous graph attention auto-encoders (HGATE) to learn node representation for heterogeneous graphs in an unsupervised manner. HGATE adopts hierarchical attention mechanism, including node-level attention and semantic-level attention. The node-level attention considers node attributes and graph structure information. The semantic-level attention fully learns semantic information represented by Meta-path in heterogeneous graph. HGATE reconstructs not only the edges of the heterogeneous graph but also node attributes, through stacked encoder/decoder layers. We include both node-level encoders and a semantic-level encoder. In the node-level encoder, the node-level attention mechanism is used to learn the attention values between the nodes and their meta-based neighbors. In the semantic-level encoder, semantic-level attention mechanism is used to learn the attention values among different meta-paths in heterogeneous graph. And so does the decoder. In the node-level encoder, node attributes are fed into stacked layers to generate meta-path based node representations. In the semantic-level encoder, the model generates new meta-path based node representations by utilizing semantic-level attention. We use the sum of all the meta-path node representations as the final node representation. In the node-level decoder, we reverse the node-level encoder to reconstruct meta-path node attributes. Each node-level decoder layer reverses the process of its corresponding node-level encoder layer. In the semantic-level decoder, we reverse the semantic-level encoder to reconstruct final node attributes. HGATE reconstructs both node attributes and the edges of the heterogeneous graph. It can efficiently generate node embedding for previously unseen data, and thus can be applied to inductive learning.

In summary, our contributions are highlighted as follows:

- We propose a novel heterogeneous graph auto-encoders (HGATE) for unsupervised representation learning on heterogeneous graph-structured data by reconstructing both node attributes and the edges of the heterogeneous graph. To the best of our knowledge, this is the first time that Heterogeneous Graph Attention Auto-Encoders is proposed.
- We use hierarchical attention for unsupervised attributed graph representation learning, in which node-level attention learns the relevance between a node and its meta-based neighbors, and semantic-level attention learns the relevance among meta-paths. Therefore, HGATE can capture semantic information in heterogeneous graphs.
- Our proposed HGATE can efficiently generate node embedding for previously unseen data. It can thus be applied to both transductive and inductive learning.
- We conduct extensive experiments on real-world heterogeneous graph data sets and the results demonstrate that our algorithms outperform the state-of-the-art methods for node classification.

The rest of the paper is organized as follows. In Section 2, we review the related works, including graph auto-encoder and heterogeneous graph neural network. In Section 3, we briefly introduce the notations used in this paper. Then we present the architecture of Heterogeneous graph attention auto-encoders (HGATE) in Section 4. In Section 5, we quantitatively and qualitatively evaluate HGATE, and present datasets, baseline methods, experiments and results. Section 6 concludes this paper.

2 Related Work

Our study is closely related to graph auto-encoder and heterogeneous graph neural networks. In this section, we briefly review the state-of-the-art literature.

2.1 Graph Auto-Encoder

As unsupervised learning frameworks, graph auto-encoders convert graph structural data into vectors in a low dimensional space, and then learn the low dimensional node vectors through stacked encoder/decoder layers. Based on the information used in graph embedding, graph auto-encoders are typically divided into two categories: topological auto-encoders and content enhanced auto-encoders. Topographic graph auto-encoder only uses topological structure information for graph embedding. Baldi [11] presented a general mathematical framework for the study of both linear and non-linear autoencoders. Kipf *et al.* [12] proposed the Variational Graph Auto-Encoders (VGAE) that used latent variables and were capable of learning interpretable latent representations for undirected graphs.

Content enhanced auto-encoder uses both topological structure features and node attributes for attributed graph representation. Compared with Topographic graph auto-encoder, Content enhanced auto-encoder can use more information to learn more comprehensive graph embedding representation. Variation autoencoder (VAE) [31] is a deep network representation model that seamlessly integrates the text information and structure of a network. Pan *et al.* [14] proposed two variants of adversarial graph embedding approach, adversarially regularized graph autoencoder (ARGA) and adversarially regularized variational graph autoencoder (ARVGA). MGAE [13] corrupts network node content, allowing node content to interact with network features, and marginalizes the corrupted features in a graph autoencoder context to learn graph representations. GraphSAGE [29] leverages node feature information to generate node embeddings, which was applicable to inductive learning. Gao *et al.* [15] captured the high nonlinearity and preserved various proximities in both topological structure and node attributes. Zhang *et al.* [16] proposed ANRL, a neighbor enhancement autoencoder, to model the node attribute information. It seamlessly integrated network structural proximity and node attribute affinity into low-dimensional representation spaces. GATE [17] stacks encoder/decoder layers and employs self-attention to reconstruct both node attributes and the graph structure. Most existing graph encoders are based on homogeneous graphs and are thus not applicable to heterogeneous graphs. In this work, we propose a novel heterogeneous graph attention auto-encoders (HGATE) for unsupervised representation learning on heterogeneous graph-structured data.

2.2 Heterogeneous Graph Neural Network

Heterogeneous graph neural network is a supervised graph embedding learning method for the heterogeneous graph. Compared to homogeneous graph embedding, heterogeneous graph embedding is much more challenging, as it needs to consider the heterogeneity and rich semantic information contained in various types of nodes and edges in the graphs. Heterogeneous graph embedding project graph data into a low dimensional space while preserving the heterogeneous network structure and node attributes so that the learned embedding can be applied to the downstream network tasks.

Some heterogeneous graph neural networks only used heterogeneous structural features [32], [33]. Jacob *et al.* [32] learned mapping heterogeneous graphs nodes representations onto a common latent space, which exploits the dependencies on the node classes, and relationships between nodes.HEER [33] embeds heterogeneous graph via edge representations to discover emerging relations from news.ProxEmbed [24] adopts Long short-term memory (LSTM) to embed the network structure between two possibly distant nodes in the heterogeneous graph to achieve semantic proximity search. Some heterogeneous graph neural networks use both heterogeneous structural features and node attributes.Zhang *et al.* [34] aggregated different types of neighbors' content features and topological features through using attention to learn the different importance between the node and its different types of neighbor groups. Zhang *et al.* [35] proposed a heterogeneous graph attention networks to model different types of entities. However, they did not consider the different weights of different entities. HANE [27] leverages heterogeneity and node attributes to generate high-quality embedding through attention mechanism. HNE [25] explores global consistency between different heterogeneous objects to learn unified feature representations guided by network structures.

In the heterogeneous graph, some methods use the meta-path to represent the multi-type semantic relations between entities [36], [37], [38], [39].PP-GCN [26] builds an event-based heterogeneous information network (HIN) and designs an event meta-schema to characterize the semantic relatedness of social events. HERec [40] uses a random walk strategy guided by meta-paths to generate meaningful node sequences for network embedding based on recommendation.MEIRec [41] leverages meta-paths to guide the selection of different-step neighbors and designed a heterogeneous GNN to obtain the rich embeddings of users and queries in intention recommendation.HAN [28] generates node embedding by aggregating features from meta-path based neighbors in a hierarchical manner, which fully considered the importance of node and meta-path. In summary, there is no unsupervised learning heterogeneous neural network model to distinguish the importance of meta-paths.

3 Preliminary

In this section, we present the notations used in this paper. They are summarized in Table 1.

Heterogeneous Graph [37]. A heterogeneous graph is represented as $G = \{V, E\}$ consisting of an object set V and a link set E. A heterogeneous graph is also associated with a node type mapping function $\phi : V \rightarrow A$ and a link type mapping function $\psi : E \rightarrow R$, where A and R denote the sets of predefined objects and link types. The number of object types $|A| > 1$ or the number of link type $|R| > 1$.

Meta-Path [23]. A meta-path is represented as m, defined on the graph of network schema $T_G = (A, R)$ of the form $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} A_3 \cdots A_L \xrightarrow{R_L} A_{L+1}$ which describes a composite relation $R = R_1 \cdot R_2 \cdots R_L$ between objects $A_1, A_2, A_3 \cdots A_{L+1}$, where $\cdot$ denotes the relation composition operator, and $L + 1$ is the length of m.

Meta-Path Based Neighbors [28]. In a heterogeneous graph, the meta-path-based neighbors N_i^m of node i are defined by the set of nodes which connect with node i via meta-path m. Note that the node's neighbors include itself.

TABLE 1
The Main Notations Used in the Paper

Notations	Definitions
N	The number of node in the graph
E	The number of edges in the graph
M	The number of meta-path in the graph
F	The number of node attribute features
k	The number of node-level neural networks layer
m	The Meta-path
$A \in R^{N \times N}$	The adjacency matrix
$X \in R^{F \times N}$	The node feature matrix
$x_i \in R^F$	The features of node i
$\hat{X} \in R^{F \times N}$	The reconstructed node feature matrix
$\hat{x}_i \in R^F$	The reconstructed features of node i
$H^k \in R^{d \times N}$	The node representation matrix generated by the k^{th} node-level encoder layer
$\hat{H}^k \in R^{d \times N}$	The node representation matrix generated by the k^{th} node-level decoder layer
$\alpha_{i,j}^k$	The attention coefficient in the k^{th} node-level encoder layer
$\hat{\alpha}_{i,j}^k$	The attention coefficient in the k^{th} node-level decoder layer
$\theta_i^{m_x,m_y}$	The attention coefficient in the semantic-level encoder layer
$\hat{\theta}_i^{m_x,m_y}$	The attention coefficient in the semantic-level decoder layer
N_i^m	The meta-path based neighborhood of node i, including itself

4 HGATE

In this section, we describe our novel graph attention auto-encoders (HGATE) for heterogeneous graphs. The architecture of HGATE reconstructs not only the edges of the heterogeneous graph but also node attributes, through stacked encoder/decoder layers based on hierarchical attention, including node-level and semantic-level attentions.

Fig. 1 illustrates the workflow of HGATE that follows a hierarchical attention structure: node-level attention encoder, semantic-level attention encoder, node-level attention decoder and semantic-level attention decoder. First, we present the encoder and decoder to show how our auto-encoder reconstructs node features using the heterogeneous graph structure and semantic information. We then describe the proposed loss function that learns node representations by minimizing the reconstruction loss of the node features and the edges of the heterogeneous graph.

4.1 Node-Level Encoder

The node-level attention encoder can generate new meta-path based representations of nodes by learning the relevance between the node and its meta-path based neighbors in the heterogeneous graph. We use a transformation matrix W to transform the input features and then project them into the next neural network layer feature space. Each encoder layer aggregates node attributes and structure information from a node's first-order neighbors by using self-attention to learn the relevance between nodes and their meta-path based neighbors. Under the same meta-path, the self-attention is shared among nodes.

Single encoder layer can aggregate node attributes and structure information from a node's first-order neighbors. We use multiple encoder layers to fully learn the node representation. Through the stack encoder layer, information can propagate through network structure. Multiple encoder layers can aggregate node attributes and structure information from node's multi-order neighbors. In the k^{th} node-level encoder layer, the relevance of meta-path based node pairs (i, j) can be formulated as follows:

$$e_{i,j}^{m,k} = \sigma(V_s^{m,k^T} W^{m,k} h_i^{m,k-1} + V_r^{m,k^T} W^{m,k} h_j^{m,k-1}), \tag{1}$$

where $W^{m,k} \in \mathbb{R}^{d^k \times d^{k-1}}$, $V_s^{m,k} \in \mathbb{R}^{d^k}$ and $V_r^{m,k} \in \mathbb{R}^{d^k}$ are the trainable parameters of the k^{th} node-level encoder layer, and σ denotes the activation function. The term $e_{i,j}^{m,k}$ represents the relevance of node j to node i, which is different from the relevance of node i to node j. Node-level attention preserves the asymmetry of heterogeneous graphs.

To make the coefficients comparable among the neighbors of nodes, we use the softmax function to normalize $e_{i,j}^{m,k}$:

$$\alpha_{i,j}^{m,k} = \frac{\exp(e_{i,j}^{m,k})}{\sum_{l \in N_i^m} \exp(e_{i,l}^{m,k})}, \tag{2}$$

where N_i^m denotes the neighborhood of node i based on meta-path, including node i itself.

We use the node feature as the initial node representation, i.e., $h_i^{m,0} = x_i$. For meta-path m, the k^{th} encoder layer generates the representation of node i in layer k as follows,

$$h_i^{m,k} = \sum_{j \in N_i^m} \alpha_{i,j}^{m,k} (W^{m,k} h_j^{m,k-1}). \tag{3}$$

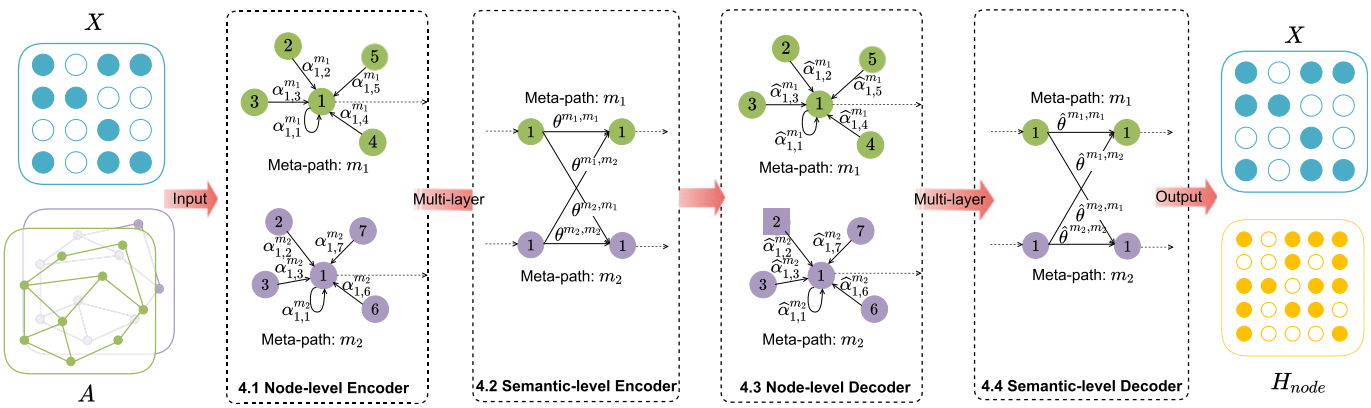

Fig. 1. The architecture of HGATE.

The node-level encoder is based on meta-path, which can capture the corresponding single semantic information.

4.2 Semantic-Level Encoder

In the heterogeneous graph, different meta-paths represent different semantic information. The semantic-level encoder can aggregate different semantic information to generate a more comprehensive node embedding through using semantic-level attention to automatically learn the relevance among different meta-paths. We use a single layer semantic-level encoder to capture the semantic information in this work.

For node i, the relevance between meta-path m_x and m_y based node embedding can be formulated as follows:

$$\omega_i^{m_x,m_y} = \sigma(V_q^T h_i^{m_x,k} + V_t^T h_i^{m_y,k}), \tag{4}$$

where $V_q \in \mathbb{R}^d$ and $V_t \in \mathbb{R}^d$ are the trainable parameters of the semantic-level encoder layer, and σ denotes the activation function.

In order to make the coefficients among meta-paths comparable, we use the softmax function to normalize $\theta_i^{m_x,m_y}$:

$$\theta_i^{m_x,m_y} = \frac{exp(\omega_i^{m_x,m_y})}{\sum_{z=1}^{M} exp(\omega_i^{m_x,m_z})}. \tag{5}$$

Then, the meta-path based embedding of node i can be aggregated by other meta-path based embedding with the corresponding coefficients as follows:

$$h_i^{m_x} = \sum_{y=1}^{M} \theta_i^{m_x,m_y} h_i^{m_y,k}. \tag{6}$$

After applying the semantic-level encoder, we consider the sum of all meta-path based node embeddings of the semantic-level encoder layer as the final node representation. The final node representation is as follows:

$$h_i = \sum_{x=1}^{M} h_i^{m_x}. \tag{7}$$

4.3 Node-Level Decoder

The node-level decoder layer reconstructs the attributes of the node based on meta-path by utilizing the representations of their neighbors according to their attention values. We use a decoder with the same number of layers as the encoder. Each decoder layer reverses the process of its corresponding encoder layer following the work of [17]. The reciprocal k^{th} node-level decoder layer corresponds to the k^{th} node-level encoder layer encoder. We use k' to represent the reciprocal k^{th} node-level decoder layer. Similar to the encoding layer, the attention value of a meta-path based neighboring node j to node i in the reciprocal k^{th} decoder layer is computed as follows:

$$\widehat{e}_{i,j}^{m,k'} = \sigma(\widehat{V}_s^{m,k'^T} \widehat{W}^{m,k'} \widehat{h}_i^{m,k'+1} + \widehat{V}_r^{m,k'^T} \widehat{W}^{m,k'} \widehat{h}_j^{m,k'+1}), \tag{8}$$

where $\widehat{W}^{m,k'} \in \mathbb{R}^{d^{k'} \times d^{k'-1}}$, $\widehat{V}_s^{m,k'} \in \mathbb{R}^{d^{k'}}$ and $\widehat{V}_r^{m,k'} \in \mathbb{R}^{d^{k'}}$ are the trainable parameters of the reciprocal k^{th} node-level decoder layer.

Similarly, we use softmax function to normalize $\widehat{e}_{i,j}^{m,k'}$ to make the coefficients comparable among the neighbors of nodes.

$$\widehat{\alpha}_{i,j}^{m,k'} = \frac{\exp(\widehat{e}_{i,j}^{m,k'})}{\sum_{l \in N_i^m} \exp(\widehat{e}_{i,l}^{m,k'})}. \tag{9}$$

Using the output of the semantic-level encoder as the input of the node-level decoder, the reciprocal k^{th} node-level decoder layer reconstructs the representation of node i in the previous layer as follows:

$$\widehat{h}_i^{m,k'} = \sum_{j \in N_i^m} \widehat{\alpha}_{i,j}^{m,k'} (\widehat{W}^{m,k'} \widehat{h}_j^{m,k'+1}). \tag{10}$$

In the node-level decoder, we are motivated to reverse the node-level encoder to reconstruct meta-path based node attributes. Each node-level decoder layer reverses the process of its corresponding node-level encoder layer.

4.4 Semantic-Level Decoder

The semantic-level decoder layer reverses the semantic-level encoder to aggregate the different meta-path information to reconstruct node attributes. The normalized relevance between meta-paths of node i in the semantic-level decoder is computed as follows:

$$\widehat{\omega}_i^{m_x,m_y} = \sigma(\widehat{V}_q^T \widehat{h}_i^{m_x} + \widehat{V}_t^T \widehat{h}_i^{m_y}), \tag{11}$$

$$\widehat{\theta}_i^{m_x,m_y} = \frac{\exp(\widehat{\omega}_i^{m_x,m_y})}{\sum_{z=1}^{M} \exp(\widehat{\omega}_i^{m_x,m_z})}. \tag{12}$$

The meta-path based embedding of node i can be aggregated by other meta-path based embedding with the corresponding coefficients as follows:

$$\widehat{h}_i^{m_x} = \sum_{y=1}^{M} \widehat{\theta}_i^{m_x,m_y} \widehat{h}_i^{m_y}. \tag{13}$$

After applying the semantic-level decoder, we consider the sum of all meta-path based node embedding of semantic-level decoder as the reconstructed node attributes.

The final reconstructed node attributes as follows:

$$\widehat{h}_i = \sum_{x=1}^{M} \widehat{h}_i^{m_x}. \tag{14}$$

4.5 Loss Function

Our model reconstructs both node attributes and the edges of the heterogeneous graph. Therefore, the loss function consists of two components: attribute loss and edge of heterogeneous graphs loss.

We minimize the reconstruction loss of node attribute feature as follows:

$$\mathcal{L}_{\text{fea}} = \sum_{i=1}^{N} ||X_i - \widehat{X}_i||_2. \tag{15}$$

In general, connected nodes in the graph are more likely to be similar to each other. We minimize the reconstruction

loss of the edges of the heterogeneous graph by making the representations of meta-path based neighboring nodes similarly:

$$\mathcal{L}_{str} = -\sum_{m=1}^{M}\sum_{i=1}^{N}\sum_{j\in N_i^m}\log\left(\frac{1}{1+\exp(-h_i^{mT}h_j^m)}\right). \tag{16}$$

M is the type of meta-path, N is the number of nodes under a certain meta-path, and N_i^m is the set of neighbor nodes of node i under meta-path m.

By merging feature loss and heterogeneous graph local structure loss, we minimize the reconstruction loss of node features and the edges of the heterogeneous graph as follows:

$$\mathcal{L} = \mathcal{L}_{fea} + \lambda\mathcal{L}_{str} \tag{17}$$

where λ controls the contribution of the edges of heterogeneous graph reconstruction loss. The overall process of HGATE is described in Algorithm 1.

Algorithm 1. The Overall Process of HGATE

Input: The node feature matrix X and The adjacency matrix set A based on meta-path.
Output: The node representation matrix H and the reconstructed node feature matrix.

Initialize node-level encoder/decoder parameters: $W^{m,k}$, $\widehat{W}^{m,k}$, $V_s^{m,k}$, $\widehat{V}_s^{m,k}$, $V_r^{m,k}$, $\widehat{V}_r^{m,k}$, semantic-level encoder/decoder parameters: $V_q, \widehat{V}_q, V_t, \widehat{V}_t$.
for $m = 1, 2, \ldots, M$ **do**
  for $k = 1, 2, \ldots, K$ **do**
    Calculate $\alpha_{i,j}$ according to (2)
    Calculate $h_i^{m,k}$ according to (3)
  end for
  Calculate $\theta_i^{m_x,m_y}$ according to (5)
  Calculate $h_i^{m_x}$ according to (6)
end for
Calculate h_i according to (7)
for $m = 1, 2, \ldots, M$ **do**
  for $k = 1, 2, \ldots, K$ **do**
    Calculate $\widehat{\alpha}_{i,j}$ according to (9)
    Calculate $\widehat{h}_i^{m,k}$ according to (10)
  end for
  Calculate $\widehat{\theta}_i^{m_x,m_y}$ according to (12)
  Calculate $\widehat{h}_i^{m_x}$ according to (13)
end for
Calculate $\widehat{h}_i$ according to (14)

4.6 Analysis of HGATE

Compared with existing methods, the proposed HGATE model has advantages such as unsupervised graph representation learning, suitability for heterogeneous graphs, applicability to transductive and inductive learning and high efficiency. We next present the analysis of the HGATE model in detail as follows.

HGATE is an unsupervised graph representation learning method that avoids the limitations of supervised learning methods, such as extensive efforts in constructing the labelled dataset. Compared with the supervised learning method, HGATE does not need to use node labels, which saves manpower and material resources to label data and avoids the impact of data label quality on the performance of the model.

HGATE is suitable for heterogeneous graphs, which reconstructs both node attributes and edges of the heterogeneous graph to generate node representations. HGATE uses the hierarchical attention mechanism to capture semantic information.

HGATE can be applied to both transductive and inductive learning. The sharing of attention parameters among nodes in HGATE can effectively generate nodes embedded in previously undiscovered data. HGATE can be applied to some real-world scenarios that need to generate the embedded representation of new nodes.

HGATE is efficient and can be parallelized. Because the computation of attention can be carried out individually across all nodes and meta-paths. The computational complexity of our architecture for one iteration is $O\Big(M \times (N \times F \times D + E \times D) + M \times N \times D\Big)$, where N is the number of nodes, E is the max number of edges based on meta-path, M is the number of meta-paths, and D is the maximum $d^{(k)}$ in all layers.

5 Evaluation

In this section, we evaluate HGATE via conducting extensive experiments on three datasets. In Section 5.1, we first introduce three heterogeneous graphs datasets. We describe the baseline approaches in Section 5.2. In Sections 5.3 and 5.4, we present the node classification and link prediction results. In Section 5.5, we introduce three kinds of variation experiments. We also present parameters sensitivity experiments in Section 5.6. We visualize the node embedding results in Section 5.7.

5.1 Datasets

We use three datasets, ACM [28], DBLP [28] and Sina Weibo for transductive and inductive tasks. We summarize the statistics of data sets in Table 2. The ACM and DBLP datasets are benchmark datasets. The ACM dataset contains 1) one type of nodes, i.e., *Papers* and 2) two types of meta-paths including *Paper-Author-Paper* and *Paper-Subject-Paper*. Paper features are the elements of a bag-of-words represented of keywords. The DBLP dataset contains 1) one type of nodes, i.e., *Author* and 2) three types of meta-paths including *Author-Paper-Author*, *Author-Paper-Conferences-Paper-Author* and *Author-Paper-Term-Paper-Author*. Author features are the elements of a bag-of-words represented of keywords. The Sina Weibo dataset is a subset extracted from Sina Weibo dataset collected by Fudan University[1], which contains one type of nodes, i.e., *User* and three kinds of meta-paths including the *following relationship, forwarding relationship* and *@ relationship*. *User* features are the elements of a bag-of-words represented of keywords. We manually divide users into three classes including the *Internet*, *Movie*, and *Politics*.

For transductive tasks, we have the access to the whole heterogeneous graph structure and all nodes' attributes on all datasets during model training. For inductive tasks, we can only access the heterogeneous graph structure and node attributes of a subgraph on all datasets during model

1. http://sma.fudan.edu.cn/datainfo/weibo.html

TABLE 2
The Statistics of the Benchmark Datasets

Datasets	#Node	Meta-paths	#Edges	#Features	#Classes
ACM	3,025	PAP	29,281	1,830	3
		PSP	2,210,761		
DBLP	4,057	APA	11,113	334	4
		APCPA	5,000,495		
		APTPA	6,772,278		
Weibo	3,135	Follow	98,573	1,303	3
		Forward	19,695		
		@	11,781		

training. And the heterogeneous graph structure and node attributes of other nodes outside the subgraph are unseen. We then use the model to generalize the representation of unseen nodes. More specifically, we randomly select 2000 nodes in the training data, and then predict the rest of the node representation by the trained models.

5.2 Baselines

We compare our HGATE with the state-of-the-art methods, including three supervised methods and five unsupervised methods. Supervised methods are GCN [9], GAT [10] and HAN [28]. Unsupervised methods are DeepWalk [42], GAE [12], VGAE [12], ASNE [43] and GATE [17]. We next briefly describe them as follows.

Supervised Methods

- GCN [9]: Graph Convolutional Networks is a semi-supervised graph convolutional network designed for homogeneous graphs.
- GAT [10]: Graph Attention Network is a semi-supervised neural network for homogeneous graphs. GAT leverages masked self-attention to learning the influence of neighboring nodes.
- HAN [28]: Heterogeneous Graph Attention Network is a semi-supervised heterogeneous graph neural network using the hierarchical attention mechanism.

Unsupervised methods

- DeepWalk [42]: DeepWalk learns social representations of a graph's vertices within short random walks for homogeneous graphs.
- GAE [12]: Graph Auto-Encoder is an unsupervised graph neural network for homogeneous graphs. GAE uses graph convolutional networks as the encoder and reconstructs the graph structure in the encoder.
- VGAE [12]: Variational Graph Auto-Encoder is a variant of GAE for homogeneous graphs. VGAE uses a graph convolutional network encoder and a simple inner product decoder.
- ASNE [43]: ASNE is a generic Attributed Social Network Embedding framework, which learns the representations for social actors while preserving both the structural proximity and attribute proximity.
- GATE [17]: Graph Attention Auto-Encoder is a neural network for the unsupervised representation of homogeneous graphs. GATE reconstructs both node attribute and graph structure by using the attention mechanism.

For the homogeneous graph embedding methods, we test all the meta-paths and report the best performance in our results.

5.3 Node Classification

In experiments, we randomly initialize parameters and use Adam optimizer to learn model parameters. Tensorflow is used to implement HGATE.

In the node classification task, we set the learning rate to 10^{-4} on ACM and Sina Weibo datasets in the transductive learning. And we set the learning rate to 5×10^{-3} on DBLP dataset. For the inductive tasks, we set the learning rate to 10^{-2}, 5×10^{-5} and 10^{-4} on ACM, DBLP, and Sina Weibo datasets, respectively. For all datasets, we use two node-layers with 512 node representation dimensions and a single semantic layer with 512 node representing dimensions. The model is trained for 200 epochs with the λ equalt to 1. We only use a half of the trainable parameters by setting equal parameters for both decoder layers and encoder layers. Both the datasets and the program codes of our HGATE are publicly available at website[2].

For baseline methods, we optimize their parameters using the validation set. For inductive learning, we use the same subgraph with HGATE to train the baseline models. For supervised methods, we use 20% of model training data as training set, 10% as verification set and 70% as test set. Unsupervised methods do not require class labels. To ensure the fairness of the experiments, we set the node embedding dimension to 512 for all baseline methods.

In this subsection, we compare our HGATE with the aforementioned state-of-the-art baselines based on transductive and inductive node classification by logistic regression. We generate the representation of all nodes. Then we use different training set proportions for training, including 20% , 50%, 80%. And we use the rest of the dataset as the test set. We report the average classification Micro-F1 and Macro-F1 in Tables 3 and 4 on the test nodes after 10 runs of training.

Table 3 shows the transductive node classification results for the ACM, DBLP, and Sina Weibo datasets. HGATE achieves powerful performance across all three datasets.

From the top of Table 3, we observe that HGATE outperforms all supervised and unsupervised baselines on the ACM dataset for transductive learning. It proves the effectiveness of our method in transductive learning. The performance of using node features for classification is better than DeepWalk, GAE, VGAE and ASNE, implying that node attributes have a significant impact on classification results in ACM dataset. GATE outperforms GAE, ASNE and VGAE, which indicates the attention mechanism can distinguish the subtle influence of different neighbors to the node. Further, compared with unsupervised method GATE, HGATE is improved by 1.15% on Micro-F1 and by 1.19% on Macro-F1 for ACM dataset with using 80% data as training set, since the semantic-level attention mechanism can capture the semantic information of heterogeneous graphs through learning the relevance among different meta-paths. HGATE is competitive with the performance of the best supervised graph embedding baseline. HGATE outperforms GCN and GAT, since HGATE is an unsupervised learning algorithm that uses feature loss and

2. https://github.com/fantasy-sxy/HGATE

TABLE 3
Experiment Results (%) on the Transductive Node Classification Task

Datasets	Training	Metrics	Feature	DeepWalk	GAE	VGAE	ASNE	GATE	**HGATE**	GCN	GAT	HAN
ACM	20%	Macro-F1	89.02	76.45	83.74	88.80	88.00	90.99	91.86	90.70	91.67	**92.30**
		Micro-F1	89.09	76.12	83.80	88.80	87.89	90.95	91.78	90.74	91.65	**92.23**
	50%	Macro-F1	89.11	77.90	84.67	89.42	87.81	91.92	**92.90**	91.82	92.16	92.87
		Micro-F1	89.16	77.46	84.73	89.43	87.64	91.94	**92.80**	91.87	92.20	**92.80**
	80%	Macro-F1	90.38	79.78	84.92	90.02	89.94	92.47	**93.66**	90.99	92.05	93.03
		Micro-F1	90.25	79.01	84.96	90.08	89.75	92.40	**93.55**	91.07	92.06	92.89
DBLP	20%	Macro-F1	78.04	91.89	91.34	91.83	90.26	89.02	90.95	90.25	91.03	**93.54**
		Micro-F1	78.80	92.39	92.08	92.42	90.82	89.93	91.65	91.22	91.90	**94.02**
	50%	Macro-F1	79.99	91.66	90.90	90.95	91.22	89.67	91.52	90.55	92.02	**94.08**
		Micro-F1	80.63	92.21	91.52	91.52	91.77	90.49	92.11	91.38	92.80	**94.48**
	80%	Macro-F1	79.92	92.13	91.62	91.64	92.09	90.74	92.57	90.83	92.42	**94.67**
		Micro-F1	80.67	92.61	92.24	92.24	92.61	91.63	93.10	91.63	93.06	**95.07**
Sina Weibo	20%	Macro-F1	90.49	96.72	97.39	97.08	96.83	95.56	**98.00**	96.22	97.38	97.50
		Micro-F1	90.55	96.69	97.37	97.09	96.77	95.53	**97.97**	96.13	97.33	97.49
	50%	Macro-F1	93.31	97.41	97.49	97.33	97.65	96.35	98.00	96.89	97.82	**98.15**
		Micro-F1	93.30	97.45	97.51	97.39	97.64	96.43	98.02	96.88	97.83	**98.21**
	80%	Macro-F1	94.68	98.44	97.73	97.38	97.32	97.49	98.60	97.52	98.43	**99.53**
		Micro-F1	94.74	98.41	97.77	97.45	97.29	97.45	98.56	97.45	98.41	**99.52**

Available data types are the node features matrix X, adjacency matrix A, and labels Y.

TABLE 4
Experiment Results (%) on the Inductive Node Classification Task

Datasets	Training	Metrics	Feature	GAE	VGAE	GATE	**HGATE**	GCN	GAT	HAN
ACM	20%	Macro-F1	89.02	78.95	78.15	88.96	**90.82**	82.68	87.42	87.17
		Micro-F1	89.09	78.93	78.18	88.88	**90.70**	83.22	87.35	87.07
	50%	Macro-F1	89.11	80.69	80.41	90.96	**92.54**	82.66	88.31	87.88
		Micro-F1	89.16	80.70	80.37	90.95	**92.47**	83.34	88.30	87.84
	80%	Macro-F1	90.38	81.46	80.29	91.44	**93.00**	82.04	90.01	88.41
		Micro-F1	90.25	81.16	80.33	91.41	**92.89**	82.81	89.91	88.26
DBLP	20%	Macro-F1	78.04	89.58	90.02	89.14	91.26	91.33	91.57	**93.51**
		Micro-F1	78.80	90.76	91.04	90.08	91.90	91.83	92.40	**94.02**
	50%	Macro-F1	79.99	90.86	90.25	90.22	92.02	91.02	91.74	**93.71**
		Micro-F1	80.63	91.57	91.03	91.13	92.56	91.52	92.47	**94.14**
	80%	Macro-F1	79.92	91.28	91.28	90.99	93.08	91.27	91.03	**94.56**
		Micro-F1	80.67	91.99	92.00	91.87	93.60	91.87	91.74	**94.95**
Sina Weibo	20%	Macro-F1	90.49	92.95	95.56	95.71	**96.52**	86.89	86.49	93.26
		Micro-F1	90.55	93.42	95.65	95.69	**96.49**	87.92	88.24	93.54
	50%	Macro-F1	93.31	94.97	96.69	96.11	**97.04**	91.43	91.24	95.36
		Micro-F1	93.30	95.28	96.81	96.17	**97.07**	91.84	92.22	95.60
	80%	Macro-F1	94.68	94.68	97.19	97.13	**98.44**	92.70	93.65	97.08
		Micro-F1	94.74	94.90	97.29	97.13	**98.41**	92.82	94.26	97.13

Available data types are the node features matrix X, adjacency matrix A, and labels Y.

edge of the graph loss for optimization without using node category labels. Compared with supervised method HAN, HGATE is improved by 0.66% on Micro-F1 and 0.63% on Macro-F1 for ACM dataset with using 80% data as the training set, which proves the effectiveness of unsupervised methods.

From the middle of Table 3, HGATE outperforms all supervised and unsupervised baselines except HAN on DBLP dataset for transductive learning. DeepWalk performs well on DBLP dataset. This is mainly because the graph structure features of DBLP has more influence on node classification. HGATE performs not as effective as HAN. Because HAN is a supervised learning method with the hierarchical attention mechanism, which uses the node classification label to learn the embedded representation of nodes. While HGATE is an unsupervised learning method, which does not use the node classification label. On the DBLP dataset, our method is still better than the latest unsupervised method.

TABLE 5
Experiment Results on the Transductive Link Prediction Task

Datasets	Metrics	Feature	DeepWalk	GAE	VGAE	ASNE	GATE	**HGATE**	GCN	GAT	HAN
ACM	Macro-F1	70.86	52.04	98.43	**99.73**	98.70	95.79	96.82	89.08	93.31	97.40
	Auc	79.22	56.84	99.89	**99.98**	99.83	98.88	98.63	96.28	96.82	99.86
DBLP	Macro-F1	92.46	52.69	97.16	97.02	97.01	98.72	**99.43**	85.59	94.88	92.43
	Auc	97.31	53.37	99.12	99.12	99.53	99.88	**99.98**	93.31	98.05	96.09
Sina Weibo	Macro-F1	66.51	57.03	88.02	85.37	**90.79**	79.76	85.12	70.66	81.45	82.89
	Auc	70.08	60.98	93.83	92.64	**97.13**	85.54	91.51	86.05	89.61	89.99

Available data types are the node features matrix X, adjacency matrix A.

Compared with unsupervised graph embedding method ASNE, HGATE is improved by 0.49% on Micro-F1 and by 0.48% on Macro-F1 with using 80% data as training set.

From the bottom of Table 3, HGATE outperforms all supervised and unsupervised baselines except HAN on Sina Weibo dataset for transductive learning. The performance of classification using user attribute features directly and DeepWalk is better on Sina Weibo dataset, which indicates that both user attributes features and graph structure features have great influence on node classification. HGATE is better than the state-of-the-art unsupervised methods. Compared with unsupervised graph embedding method GAE, HGATE is improved by 0.87% on Macro-F1 and 0.79% on Micro-F1 with using 80% data as training set. Similar to the results on DBLP dataset, HGATE performs slightly worse than the HAN on Sina Weibo dataset.

Table 4 shows the inductive node classification results for ACM, DBLP, and Sina Weibo datasets. Accordingly, we observe that HGATE also performs well in the inductive task, which can effectively generate the embedded representation of unseen nodes in model training. The performance of inductive learning is generally not as good as that of transductive learning, because inductive learning only uses part of data set for training, and the available information is limited.

From the top of Table 4, similar to the transductive tasks, HGATE achieves the best performance among all the supervised and unsupervised baseline methods on the ACM datasets for inductive learning. For ACM dataset, HGATE is 1.56% higher than the best unsupervised learning algorithm on Macro-F1 with using 80% data as training set. HGATE is 2.99% higher than the best supervised learning algorithm on Macro-F1 and 2.98% higher on Micro-F1 with using 80% data as training set. HGATE outperforms HAN and GATE is better than GAT, showing that the unsupervised learning algorithm is better than the supervised learning methods in inductive learning on ACM dataset.

From the middle of Table 4, HGATE outperforms all supervised and unsupervised baselines except HAN on DBLP dataset for inductive learning. For DBLP datasets, HGATE achieves better performance than unsupervised baseline methods. Compare unsupervised baseline methods, HGATE achieves an improvement gain of 1.60% on Micro-F1 and 1.80% on Macro-F1 with using 80% data as the training set. And HGATE performs better than supervised baseline methods except HAN.

From the bottom of Table 4, HGATE performs better than all supervised and unsupervised baselines on Sina Weibo dataset. Compared with unsupervised baseline methods, HGATE achieves an improvement gain of 1.25% on Macro-F1 and 1.12% on Micro-F1 with using 80% data as training set. Compared with supervised baseline methods, HGATE achieves an improvement gain of 1.36% on Macro-F1 and 1.28% on Micro-F1 with using 80% data as training set. For inductive learning, the performance of unsupervised method is better than that of the supervised method on Sina Weibo dataset, which proves the superiority of unsupervised method for inductive learning.

Based on the above analysis, HGATE performs better than the most stats-of-art unsupervised methods for node classification. And HGATE outperforms or matches the most state-of-art supervised methods in the node classification experiment. HGATE can efficiently generate node embedding for unseen nodes.

5.4 Link Prediction

In this subsection, we compare our HGATE with the aforementioned baselines on transductive and inductive link prediction by logistic regression. We compare different models based on their ability to correctly classify each example (node pair) into links and non-links. We create evaluation examples from the links and an equal number of randomly sampled pairs of unconnected nodes. We compute the feature representation for a pair of nodes, using the Hadamard Operator[3] for all methods. The Hadamard operator computes the element-wise product of two vectors and closely mirrors inner product operation in learning node embeddings. We report the area under the ROC curve(AUC) and F1 scores for each model on the test set. For the homogeneous graph embedding methods, we test all the meta-paths and report the best performance in our results. We randomly select 100000 edges in each graph, and then choose 80% training-set proportions for training. Meanwhile, we use the rest of the dataset as the validation set and the test set. We report Macro-F1 and AUC in Tables 5 and 6 on the test nodes.

For the transductive tasks, we set the learning rate to 5×10^{-2}, 10^{-3} , 10^{-2} on ACM, DBLP and Sina Weibo datasets, respectively. For the inductive tasks, we set the learning rate to 10^{-3} on ACM, DBLP, and 5×10^{-2} Sina Weibo datasets, respectively. For all datasets, we use two node-layers with 512 node representation dimensions and a single semantic layer with 512 node representing dimensions. The model is trained for 200 epochs with the λ equalt to 1.

For baseline methods, we optimize their parameters using the validation set. For inductive learning, we use the

TABLE 6
Experiment Results on the Inductive Link Prediction Task

Datasets	Metrics	Feature	GAE	VGAE	GATE	**HGATE**	GCN	GAT	HAN
ACM	Macro-F1	70.86	**99.75**	99.62	97.32	95.42	86.52	84.41	96.79
	Auc	79.22	99.95	**99.98**	99.44	98.56	85.55	92.15	99.37
DBLP	Macro-F1	92.46	96.87	98.01	98.15	**98.75**	85.59	94.16	92.39
	Auc	99.23	99.12	**99.78**	99.71	99.70	93.38	98.49	92.89
Sina Weibo	Macro-F1	66.51	78.33	78.61	80.67	**84.67**	68.07	81.25	82.95
	Auc	70.08	88.17	87.69	87.98	**91.12**	81.75	89.42	88.84

Available data types are the node features matrix X, adjacency matrix A.

same subgraph with HGATE to train the baseline models. To ensure the fairness of the experiments, we set the node embedding dimension to 512 for all baseline methods.

Results for the link prediction task are summarized in Tables 5 and 6, the observations are as follows:

1) The performance is weak for all datasets for deepwalk, that is because deepwalk is based on random walk without considering the feature. This demonstrates the usefulness of attributes in predicting missing links. Note that the method based on features has a good performance in DBLP datasets.
2) The performance of our model outperforms the GCN, GAT cross all the datasets. And all the methods based on auto-encoder outperform GCN, GAT, HAN. GCN, GAT and HAN can fully leverage node label and the rich information in attributes, this maybe damage the performance of link prediction because two nodes maybe similar but there is no chance to have a link between them in actual scene.
3) In transductive link prediction, our model outperforms other methods in DBLP datasets. And in inductive link prediction, our model outperforms other methods in DBLP and Weibo datasets. Compared to the GAE, VGAE and GATE, we observe performance drop of our model and HAN. The reason is GAE, VGAE, ASNE and GATE learn different representation vectors for the same node in different graphs in experiments. However, our model learns the same vector for the same node in different meta-paths. In addition, ASNE and GAE, VGAE model integrate negative sample into the loss function, their models can weaken the similarity of unconnected nodes node better than other methods. The performance of our model in DBLP dataset is better than the results in ACM and Sina Weibo datasets. The reason is ACM and Weibo datasets contain less link information. The link sparsity problem makes our model cannot fully learn the graph structure.

5.5 Ablation Experiments

We compare our architecture with its three variant based on transductive and inductive learning. We use the node classification task as representatives. Our model HGATE contains three important components including semantic-level attention, reconstruction of node attributes and reconstruction of edges of the heterogeneous graph features. And we design the variant methods of HGATE by using these three components. We compare the influences of the three components on both ACM and DBLP datasets in node classification task. The three variant method including HGATE$_{nodeatt}$, HGATE$_{str}$ and HGATE$_{fea}$.

- HGATE$_{fea}$: a variant of HGATE which uses semantic-level attention and reconstruction node attributes.
- HGATE$_{str}$: a variant of HGATE which uses semantic-level attention and reconstruction the edges of the heterogeneous graph features.
- HGATE$_{nodeatt}$: a variant of HGATE which reconstruction node attributes and reconstruction the edges of the heterogeneous graph features removing the semantic-level encoder/decoder layers.

Figs. 2 and 3 show the transductive and inductive results of HGATE's variants respectively. HGATE outperforms or matches its variants in all the datasets. This approves that each component contributes to the overall performance of our architecture.

For the transductive learning results in Fig. 2, HGATE$_{nodeatt}$ performs worse than or equal to other variants on ACM dataset, which approve that semantic-level attention has a significant effect on the node embedding results on ACM dataset. HGATE$_{str}$ performs worse than HGATE$_{fea}$ on ACM, which approve that graph structure is more important than node attributes on node classification for ACM dataset. The results of HGATE$_{fea}$ and HGATE are extraordinary close on ACM dataset, indicating that the contribution of node attributes to the node embeddings result is strong on ACM dataset. On the contrary, HGATE$_{fea}$ performs worse than other variants on DBLP dataset, which approve that among the three components, node attributes are the least important for node embedding results on DBLP dataset. The results of HGATE$_{str}$ and HGATE are extraordinary close on DBLP dataset, indicating that the contribution of graph structure to the node embeddings result is strong on DBLP dataset. Similar to the results of DBLP dataset, the performance of HGATE$_{str}$ is better than that of HGATE$_{fea}$, which indicates that graph structure is more important than node attributes on Weibo dataset. The performance of HGATE$_{nodeatt}$ is close to that of HGATE, which proves that three different meta-paths have similar effects on the node embedding of Sina Weibo dataset.

For the inductive learning results in Fig. 3, HGATE$_{nodeatt}$ performs worse than other variants on ACM dataset, which approve that semantic-level attention can well learn the importance among different meta-paths on ACM dataset.

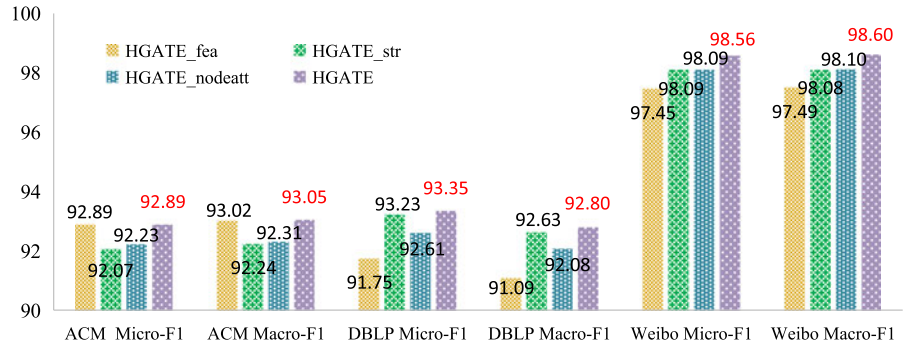

Fig. 2. The node classification results on transductive learning for variants of HGATE.

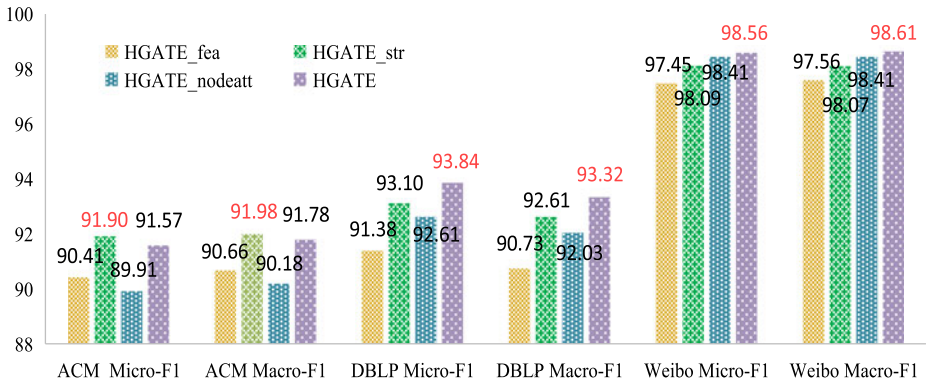

Fig. 3. The node classification results on inductive learning for variants of HGATE.

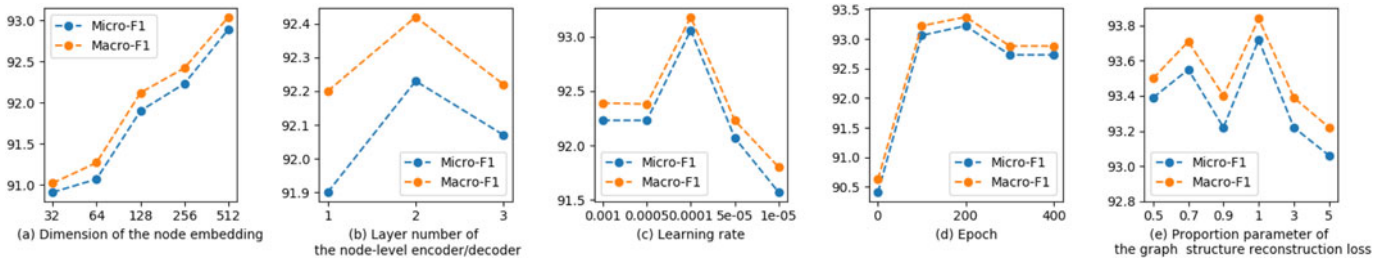

Fig. 4. Parameters sensitivity of HGATE.

HGATE_{fea} performs worse than HGATE_{str} on ACM and DBLP datasets, which approve that graph structure is more important than node attributes on node classification for ACM dataset and DBLP datasets. The results of HGATE_{str} and HGATE are extraordinary close on ACM dataset, indicating that the contribution of node attributes to the node embeddings result is weak on ACM dataset. HGATE_{fea} performs worse than other variants on DBLP dataset, which approve that among the three components, node attributes are the least important for node embedding results on DBLP dataset. For Sina Weibo dataset, HGATE_{fea} performs worse than other variant models, which indicates that node attributes have less effect on node embedding for inductive learning.

5.6 Parameters Sensitivity

We analyze the influence of model parameters on the experimental results of node classification on ACM dataset for transductive learning. As shown in Fig. 4, we choose the appropriate parameters to achieve the satisfied node classification performance.

Node Embedding Dimension. With the increase of node embedding dimension, the performance of node classification is improved. Higher node embedding dimension can represent more comprehensive node attribute and structure information. However, as the node embedding dimension increases, the complexity of the algorithm is also increased. In order to achieve the balance between the classification results and the training time of the model, we set the node dimension to 512.

Node-Level Encoder/Decoder's Layer Numbers. With the increase the number of node-level encoder/decoder's layer, the performance of node classification first rises and then falls. The suitable node-level encoder/decoder's layer numbers can improve the learning ability of HGATE. However, the layer numbers continue to increase, the effective transfer of information in the model is hindered, which leads to poor embedding effect. As shown in Fig. 4, in order to achieve better performance of node classification for transductive learning, we set the layer number of the node level encoder/decoder to 2.

Learning Rate. Similarly, the performance of node classification first rises and then falls with the increase of learning rate. HGATE needs an appropriate learning rate to minimize model loss. We set the learning rate to 0.0001 to achieve better performance of node classification for transductive learning.

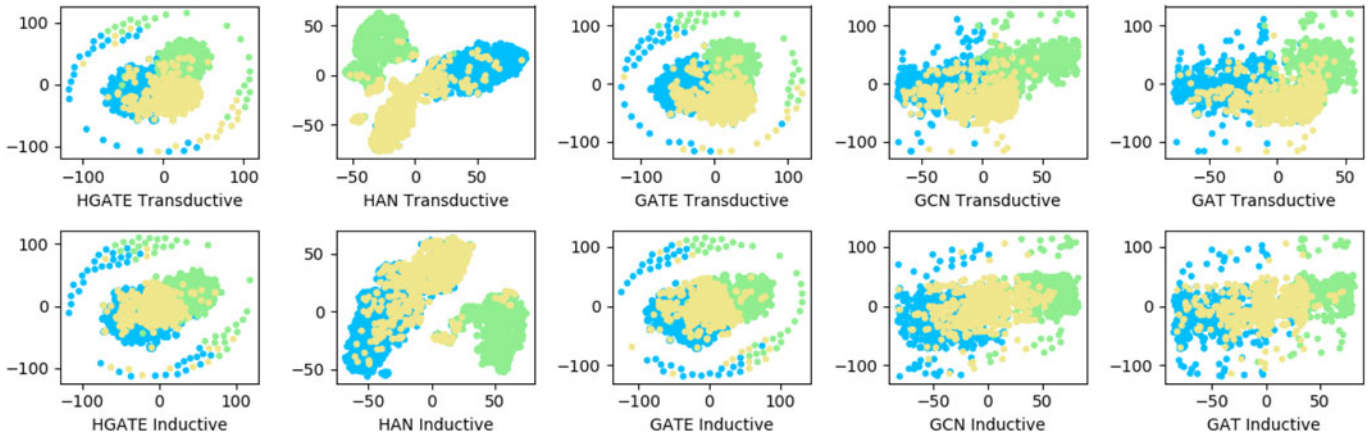

Fig. 5. Visualization embedding on ACM data.

Epoch. With the increase of the training epoch, the performance of node classification first increases significantly and then changes slowly. With the increase of epoch, the model continues to learn from the original data to obtain a better node embedding representation. According to the elbow method, when the training epoch is 200, HGATE achieves better node classification performance.

Proportion Parameter of the Edges of the Heterogeneous Graph Reconstruction Loss. With the increase of the proportion parameter of the edges of the heterogeneous graph reconstruction loss, the performance of node classification fluctuates slightly. HGATE model loss includes node attribute feature loss and graph structure loss. By adjusting the proportion parameter of the edges of the heterogeneous graph reconstruction loss, we can adjust the influence of graph structure on the result of node embedding, and then affect the result of node classification. When the proportion parameter of the edges of the heterogeneous graph reconstruction loss is 1, HGATE achieves better node classification performance.

5.7 Visualization

In order to compare the node embedding results of different models, we visualize the node embedding results on the ACM dataset for inductive and transductive learning. In order to realize visualization, we utilize t-SNE to project the node embedding results into two-dimensional space. We compare the node embedding results of HGATE with the other five models, including HAN, GATE, GCN and GAT. The visualization results are shown in Fig. 5, and the color of the node represents the category label of the node.

From Fig. 5, the result of node embedding in transductive learning of all models is better than that of inductive learning. Because transductive learning uses all data to learn the embedded representation of nodes, which can fully learn the relationship between nodes, while inductive learning uses part of data for training, and the available information is limited. The node embedding of HAN is better than other models, because HAN is a supervised learning method based on hierarchical attention mechanism, which uses the node category label to get the node embedding representation. The nodes embedding representation of our model (HGATE) shows better visualization performance, on account of HGATE is an unsupervised learning method without using node labels, which optimizes the results by minimizing the loss of node attributes and heterogeneous graph structure. Similarly, GATE is an unsupervised learning method, its visualization results are similar to our model HGATE. GCN and GAT are supervised homogeneous graph methods, and their visualization results are similar.

6 Conclusion

In this work, we propose the heterogeneous graph auto-encoders (HGATE) that is a novel unsupervised heterogeneous graph embedding method. HGATE uses the hierarchical attention mechanism to learn the importance of nodes and meta-paths, which can capture complex structures and rich semantic information on the heterogeneous graph. HGATE not only reconstructs the edges of the heterogeneous graph but also reconstructs the node attributes. It efficiently generates node embedding for previously unseen data, and thus can be applied to both transductive and inductive learning. We conduct comprehensive experiments on real-world heterogeneous graphs datasets to validate our models in node classification and link prediction task.

In future work, we will consider multi-types of nodes in the heterogeneous graph for graph embedding representation.

Acknowledgements

Xiaoyang Suo and Xiangyu Wei contributed to the work equally and should be regarded as co-second authors.

References

[1] P. Kazienko and T. Kajdanowicz, "Label-dependent node classification in the network," *Neurocomputing*, vol. 75, no. 1, pp. 199–209, 2012.

[2] H. Narayanan, M. Belkin, and P. Niyogi, "On the relation between low density separation, spectral clustering and graph cuts," in *Proc. Adv. Neural Inf. Process. Syst. 19 Proc. 20th Annu. Conf. Neural Inf. Process. Syst.*, Vancouver British Columbia Canada, 2006, pp. 1025–1032.

[3] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining San Francisco*, 2016, pp. 855–864

[4] P. Wang, K. Agarwal, C. Ham, S. Choudhury, and C. K. Reddy, "Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks," in *Proc. Web Conf.*, 2021, pp. 2946–2957.

[5] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. 31tst AAAI Conf. Artif. Intell.*, 2017, pp. 203–209. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14589
[6] S. Pei, L. Yu, G. Yu, and X. Zhang, "REA: Robust cross-lingual entity alignment between knowledge graphs," in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Mining Virtual Event*, 2020, pp. 2175–2184.
[7] X. Sun *et al.*, "Heterogeneous hypergraph embedding for graph classification," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 725–733.
[8] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 974–983.
[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.
[10] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Representations*, 2018, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ
[11] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proc. Int. Conf. Mach. Learn. Workshop Unsupervised Transfer Learn.*, 2012, pp. 37–49.
[12] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *CoRR*, vol. abs/1611.07308, 2016. [Online]. Available: http://arxiv.org/abs/1611.07308
[13] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized graph autoencoder for graph clustering," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 889–898.
[14] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2609–2615.
[15] H. Gao and H. Huang, "Deep attributed network embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, vol. 18, 2018, pp. 3364–3370.
[16] Z. Zhang *et al.*, "ANRL: Attributed network representation learning via deep neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3155–3161.
[17] A. Salehi and H. Davulcu, "Graph attention auto-encoders," *CoRR*, vol. abs/1905.10715, 2019. [Online]. Available: http://arxiv.org/abs/1905.10715
[18] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 891–900.
[19] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1225–1234.
[20] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. 31st Conf. Assoc. Adv. Artif. Intell.*, 2017, pp. 203–209.
[21] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "MetaGraph2Vec: Complex semantic path augmented heterogeneous network embedding," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Berlin, Germany: Springer, 2018, pp. 196–208.
[22] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and P. S. Yu, "A survey on heterogeneous graph embedding: Methods, techniques, applications and sources," pp. 1–23, 2020, *arXiv:2011.14867*. [Online]. Available: https://arxiv.org/abs/2011.14867
[23] 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. Very Large Data Bases Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.
[24] Z. Liu *et al.*, "Semantic proximity search on heterogeneous graph by proximity embedding," in *Proc. Conf. Assoc. Advancement Artif. Intell.*, 2017, pp. 154–160.
[25] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 119–128.
[26] H. Peng *et al.*, "Fine-grained event categorization with heterogeneous graph convolutional networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3238–3245.
[27] Y. Wang, Z. Duan, B. Liao, F. Wu, and Y. Zhuang, "Heterogeneous attributed network embedding with graph convolutional networks," in *Proc. Conf. Assoc. Advancement Artif. Intell.*, 2019, pp. 10 061–10 062.
[28] X. Wang *et al.*, "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, 2019, pp. 2022–2032.
[29] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[30] B. Huang and K. M. Carley, "Residual or gate? Towards deeper graph neural networks for inductive graph representation learning," *Clin. Orthopaedics Related Res.*, vol. abs/1904.08035, pp. 1–8, 2019.
[31] H. Li, H. Wang, Z. Yang, and M. Odagaki, "Variation autoencoder based network representation learning for classification," in *Proc. Anterior Cruciate Ligament Student Res. Workshop*, 2017, pp. 56–61.
[32] Y. Jacob, L. Denoyer, and P. Gallinari, "Learning latent representations of nodes for classifying in heterogeneous social networks," in *Proc. 7th ACM Int. Conf. Web Search Data Mining*, 2014, pp. 373–382.
[33] J. Zhang, C.-T. Lu, M. Zhou, S. Xie, Y. Chang, and S. Y. Philip, "HEER: Heterogeneous graph embedding for emerging relation detection from news," in *Proc. IEEE Int. Conf. Big Data*, 2016, pp. 803–812.
[34] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 793–803.
[35] F. Zhang *et al.*, "OAG: Toward linking large-scale heterogeneous entity graphs," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2585–2595.
[36] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu, "Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2012, pp. 1348–1356.
[37] Y. Sun and J. Han, "Mining heterogeneous information networks: A structural analysis approach," *ACM Sigkdd Explorations Newslett.*, vol. 14, no. 2, pp. 20–28, 2013.
[38] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu, "Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," *ACM Trans. Knowl. Discov. Data*, vol. 7, no. 3, pp. 11:1–11:23, Sep. 2013.
[39] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 135–144.
[40] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, Feb. 2018.
[41] S. Fan *et al.*, "Metapath-guided heterogeneous graph neural network for intent recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2478–2486.
[42] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
[43] L. Liao, X. He, H. Zhang, and T. Chua, "Attributed social network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2257–2270, Dec. 2018.

Wei Wang received the PhD degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2006. He is currently full professor in the School of Computer and Information Technology, Beijing Jiaotong University, China. He was a postdoctoral researcher in University of Trento, Italy, from 2005 to 2006. He was a postdoctoral researcher in TELECOM Bretagne and in INRIA, France, from 2007 to 2008. He was a european ERCIM fellow in Norwegian University of Science and Technology (NTNU), Norway, and in Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, from 2009 to 2011. He visited INRIA, ETH, NTNU, CNR, New York University Tandon School of Engineering, and KAUST. He was one of the "Highly Cited Chinese Researchers" selected by Elsevier in 2020. He is an editorial board member for Computer & Security. He has authored or coauthored more than 100 peer-reviewed papers in various journals and international conferences. His main research interests include machine intelligence and its applications such as social networks and data security.

Xiaoyang Suo received the BS degree from Lanzhou Jiaotong University, Lanzhou, China, in 2017 and the MS degree from Beijing Jiaotong University, Beijing, China, in 2020. Her main research interests lie in data mining.

Xiangyu Wei received the BS degree from Shanxi University, Taiyuan, China, in 2016. She is currently working toward the PhD degree with the School of Computer and Information Technology in Beijing Jiaotong University, Beijing, China. Her main research interests lie in social network analysis and data security.

Bin Wang is currently a research professor with Zhejiang Key Laboratory of Multi-dimensional Perception Technology, Application and Cybersecurity. His research interests mainly include computer networks, artificial intelligence security, Internet of Things security. In these areas he has published more than 50 papers in international journals or conferences.

Hao Wang (Senior Member, IEEE) received the BEng and PhD degrees in computer science and engineering from the South China University of Technology, Guangzhou, China, in 2000 and 2006, respectively. He is currently an associate professor with the Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway. He is also a visiting PI with the Research Center for Optical Fibre Sensing, Zhejiang Lab, China, and a visiting professor with the School of Cyber Engineering, Xidian University, Xi'an, China. He has authored or coauthored more than 170 papers in reputable international journals and conferences. His research interests include big data analytics, industrial Internet of Things, high-performance computing, and safety-critical systems. He is the editorial board member and guest editor for several international journals. He was a TPC co-chair for IEEE GPC 2021, CPSCom 2020, IEEE CIT 2017, ES 2017, IEEE DataCom 2015, a senior TPC member for ACM CIKM 2019, and reviewer for many prestigious journals and conferences. He is a member of ACM. He was the recipient of prestigious awards such as the IEEE Outstanding Paper Award and the IEEE Outstanding Leadership Award. He is the chair for Sub TC on Healthcare in IEEE IES Technical Committee on Industrial Informatics.

Hong-Ning Dai (Senior Member, IEEE) received the PhD degree in computer science and engineering from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. He is currently an associate professor with Department of Computing and Decision Sciences, Lingnan University, Hong Kong. His current research interests include the Internet of Things, Big Data, and blockchain technology. He has served as associate editors/editors for *IEEE Transactions on Industrial Informatics*, *IEEE Systems Journal*, *Ad Hoc Networks*, and *Connection Science*. He is also a senior member of Association for Computing Machinery (ACM).

Xiangliang Zhang (Senior Member, IEEE) received the PhD degree in computer science from INRIA-Universite Paris Sud, France, in 2010. She is currently an associate professor with the department of Computer Science and Engineering, University of Notre Dame. She was an associate professor in Computer Science with the King Abdullah University of Science and Technology (KAUST), Saudi Arabia. She has authored or coauthored more than 170 refereed papers in leading international conferences and journals in data sciences. She serves as associate editor of *IEEE Transactions on Dependable and Secure Computing*, *Information Sciences and International Journal of Intelligent Systems*, and regularly serves as area chair or on the (senior) program committee of IJCAI, SIGKDD, NeurIPS, AAAI, ICML, and WSDM. Her main research interests and experiences are in machine learning and data mining.