Heterogeneous Graph Neural Network With Multi-View Representation Learning

Zezhi Shao, Yongjun Xu, Wei Wei, Fei Wang, Zhao Zhang, and Feida Zhu

Abstract—In recent years, graph neural networks (GNNs)-based methods have been widely adopted for heterogeneous graph (HG) embedding, due to their power in effectively encoding rich information from a HG into the low-dimensional node embeddings. However, previous works usually easily fail to fully leverage the inherent heterogeneity and rich semantics contained in the complex local structures of HGs. On the one hand, most of the existing methods either inadequately model the local structure under specific semantics, or neglect the heterogeneity when aggregating information from the local structure. On the other hand, representations from multiple semantics are not comprehensively integrated to obtain node embeddings with versatility. To address the problem, we propose a *Heterogeneous Graph Neural Network* for HG embedding *within a Multi-View representation learning framework* (named MV-HetGNN), which consists of a view-specific ego graph encoder and auto multi-view fusion layer. MV-HetGNN thoroughly learns complex heterogeneity and semantics in the local structure to generate comprehensive and versatile node representations for HGs. Extensive experiments on three real-world HG datasets demonstrate the significant superiority of our proposed MV-HetGNN compared to the state-of-the-art baselines in various downstream tasks, e.g., node classification, node clustering, and link prediction.

Index Terms—Heterogeneous graphs, graph neural networks, graph embedding

1 Introduction

Graph structured data is ubiquitous in the real world, such as social networks [1], [2], [3], [4] and citation networks [5], [6]. In recent years, *graph neural networks* (GNNs) have become one of the standard paradigms for analyzing graph structured data. The core idea of GNNs is to explore the multi-hop local structure of the target node, i.e., the *ego network* or *ego graph* [7], by stacking multiple layers. As a basic graph structure, homogeneous graphs consist of only one type of nodes and edges, and the ego graph in it has a clear definition and intuition [8], e.g., *first*-order or *second*-order structure. Therefore, GNNs have achieved great success on homogeneous graphs [5], [6].

However, traditional GNN-based approaches cannot be directly applied to *heterogeneous graphs* (HGs) because of two essential properties of HGs: *heterogeneity* and *semantics*. An example of an ego graph depicting the complex local structure in HGs is shown in Fig. 1a. *First*, the ego graph of HGs is equipped with multiple types of nodes and relations, i.e., *heterogeneity*. The features of different types of nodes fall in different feature spaces, hindering the aggregation operation of GNNs. For example, the *Author* nodes need to be projected into the feature space of the *Paper* nodes through the *Write* relation so as to aggregate with other *Paper* nodes. *Second*, many meaningful and complex semantic information implicitly exists in the ego graph of HGs, i.e., *semantics*. These implicit but important relationships are captured by high-order relations, i.e., *metapaths*. Different metapaths reveal different semantics, which can be regarded as a view to observe the target node's local structure. For example, as shown in Fig. 1b, *Author-Paper-Author (APA)* indicates the co-author relationship while the *Author-Paper-Conference-Paper-Author (APCPA)* indicates the co-conference relationship.

In order to apply GNNs on HGs, a number of works have emerged recently. Considering the two properties mentioned above, we divide related works into two categories. The first category mainly focuses on heterogeneity. These works model type-specific mapping functions to project heterogeneous nodes to the same feature space to eliminate heterogeneity. Then, they apply GNNs and stack multiple layers to catch high-order information [11], [12], [13], [14]. However, these methods do not explicitly make use of semantics. Moreover, simply stacking multiple layers to catch high-order information causes lower-order semantic information to suffer from over-smooth due to the different lengths of semantics in HGs. Instead, the second category explicitly utilizes metapaths as semantics[1] to guide information aggregation. First, they

- *Zezhi Shao is with the Institute of Computing Technology, CAS, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: shaozezhi19b@ict.ac.cn.*
- *Yongjun Xu, Fei Wang, and Zhao Zhang are with the Institute of Computing Technology, CAS, Beijing 100190, China. E-mail: {xjy, wangfei, zhangzhao2021}@ict.ac.cn.*
- *Wei Wei is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: weiw@hust.edu.cn.*
- *Feida Zhu is with the School of Information Systems, Singapore Management University, Singapore 178902. E-mail: fdzhu@smu.edu.sg.*

Manuscript received 15 August 2021; revised 16 July 2022; accepted 12 November 2022. Date of publication 23 November 2022; date of current version 6 October 2023.
This work was supported in part by the National Natural Science Foundation of China under Grants 61902376, 61902382, and 62276110, in part by CCF-AFSG Research Fund under Grant RF20210005, and in part by the fund of Joint Laboratory of HUST and Pingan Property & Casualty Research (HPL). In addition, The work of Zhao Zhang was supported in part by the China Postdoctoral Science Foundation under Grant 2021M703273.
(Corresponding authors: Wei Wei and Fei Wang.)
Recommended for acceptance by P. Cui.
Digital Object Identifier no. 10.1109/TKDE.2022.3224193

1. Following [9], [10], we use semantic and metapath interchangeably in this paper.

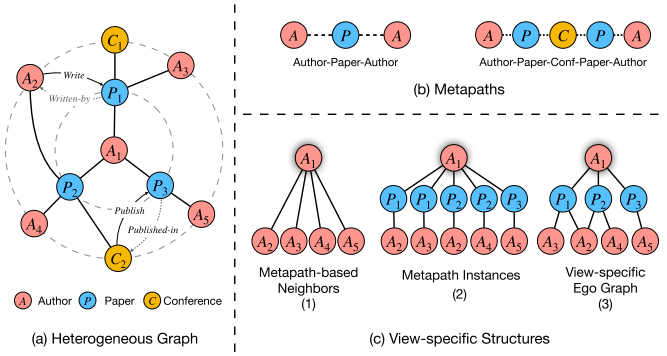


Fig. 1. An illustrative example of heterogeneous graph (DBLP) and some key concepts. (a) The local structure of node A_1 in the heterogeneous graph DBLP. (b) Two metapaths in DBLP. (c) Three types of metapath-based local structures of author node A_1 (based on metapath APA) proposed by HAN[9], MAGNN[10], and our MV-HetGNN.

decompose the multi-hop local structure of HGs, which contains complex semantic information, into multiple local structures under different semantics. Then, the node representations under different semantics are extracted by aggregating the local structure. Finally, representations under different semantics are fused together. Benefiting from modeling the heterogeneity and semantics simultaneously, these methods are effective and popular [9], [10], [15], [16], [17], [18], [19].

Despite the encouraging results, previous works still fail to fully leverage the inherent heterogeneity and rich semantics contained in the complex local structures of HGs. Specifically, they still suffer from at least one of the following limitations. (1) The modeling of local structures under each semantics is inadequately [9], [10], [19]. For example, as shown in Fig. 1c, HAN [9] aggregates information from metapath-based neighborhoods, discarding the intermediate nodes. MAGNN [10] aggregates information at the metapath instance level (i.e., sequence level), and neglects the overall graph structure of the local structure, which causes information loss and inflexibility in the aggregation process. (2) The modeling of the mapping function between heterogeneous nodes is neglected when aggregating the local structures under each semantics [17], [18]. That is, the heterogeneity of HGs is not fully explored. Most of these works only project the feature of heterogeneous nodes to the same dimension[9], [19], [20] and aggregate features directly, neglecting the mapping relation between them. (3) The representations from multiple semantics are not comprehensively utilized to obtain versatile node representations. Most methods use simple concatenation or the attention mechanism to fuse representations from different semantics [9], [10], [15], [20]. However, the simple concatenation preserves redundant information among multiple semantics, leading to poor performance. Meanwhile, the attention mechanism can not theoretically guarantee versatile node embeddings, i.e., superior to any single view representation. Moreover, our experiments in Section 5 demonstrate that the attention mechanism can not consistently outperform the simple mean average or single view.

To tackle these issues, we model the HG embedding from the perspective of *Multi-view Representation Learning* (MvRL). The intuitive idea of introducing MvRL is that the semantic property of HGs conforms to the idea of MvRL, i.e., each semantics can be regarded as a view to observe the target node's local structure. Different from existing works, the MvRL idea specifically requires adequate modeling of the structure and heterogeneity under each view, and comprehensive integration of representations from multiple views. On the one hand, within each view, it requires adequately aggregating local structures and modeling heterogeneity to obtain view-specific representations. On the other hand, among multiple views, it requires comprehensively integrating representations from multiple views into a latent representation with theoretical guarantees of versatility (i.e., superior to any single view) and better performance in downstream tasks. Based on the above motivation, we propose a *Heterogeneous Graph Neural Network* for HG embedding *within a Multi-View representation learning framework* (named MV-HetGNN). MV-HetGNN solves the above problems with two key components, the view-specific ego graph encoder and auto multi-view fusion layer. On the one hand, MV-HetGNN first decomposes the original complete ego graph in HGs into a set of view-specific ego graphs, which preserve complete local structures under each semantics. Then, the view-specific ego graph encoder leverages TransE [21] to learn the representations of relations in HGs, so as to model the mapping function between heterogeneous nodes and aggregate information at the graph level in a bottom-up manner. On the other hand, the auto multi-view fusion layer aims to integrate the embeddings from different views comprehensively. There are two main challenges. First, there is much redundant information among multi-view representations [22], i.e., the overlap of representations. Second, the fused representations require a theoretical guarantee of versatility [23], i.e., performing at least as good as any single-view representation. Auto multi-view fusion layer overcomes both challenges by using hierarchical autoencoders with orthogonal regularization. In general, our method solves many issues and extends the second category by introducing the idea of MvRL. In summary, this work makes several major contributions:

- We model the local structure under each semantics as view-specific ego graphs and propose a novel view-specific ego graph encoder module. It learns the representations of relations to model the mapping function between heterogeneous nodes to address heterogeneity and aggregates information comprehensively.
- We propose a novel auto multi-view fusion layer to comprehensively integrate the embeddings from different semantics. It utilizes hierarchical autoencoders with orthogonal regularization to remove the redundant information cross multiple views and obtain versatile node embeddings with theoretical guarantees.
- We conduct extensive experiments on three public real-world HG datasets with three different tasks. The experimental results demonstrate the superiority of MV-HetGNN over seven state-of-the-art models.

The paper is organized as follows. Section 2 covers related work, and Section 3 presents preliminaries and the problem definition. In Section 4, we present the MV-HetGNN model in detail. In Section 5, we conduct extensive experiments and visualizations to evaluate the effectiveness

of MV-HetGNN. In particular, Section 5.2 compares the performance of MV-HetGNN with other baselines. Section 5.3 compares our view-specific ego graph encoder with existing approaches [9], [10], and Section 5.4 compares our auto multi-view fusion layer with other popular approaches in detail. Finally, Section 6 concludes the paper.

2 Related Work

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) are proposed to apply deep neural networks to deal with graph-structured data. GNNs can be divided into two categories: spectral-based and spatial-based methods. Spectral-based GNNs define convolution operations in the Fourier domain by computing the eigendecomposition of graph Laplacian [6], [24], [25]. Therefore, these models usually require an entire graph as input, and the eigendecomposition is computationally expensive, making them inefficient and lacking generalization ability.

Spatial-based GNNs [5], [26] define convolution operations directly in the graph domain by aggregating the information from the target nodes' local structures [27], i.e., ego graph [7]. For example, GAT [26] aggregates information according to the importance score of neighborhoods assigned by a masked self-attention layer [28]. Recently, many new powerful GNN models and theories [29], [30], [31] have been proposed, which improve the representation power of GNNs. In order to exceed the expressive power of the 1-WL test, ID-GNN [7] assigns different message-passing parameters to the central node and other nodes when aggregating information from the ego graph. Due to their high efficiency, strong performance, and generalization ability, spatial GNNs have become mainstream. However, these powerful models and theories are built for homogeneous graphs, in which the local structure is well defined and has good intuition, e.g., first-order, second-order [8]. The unique properties of HGs, *heterogeneity* and *semantics*, are not considered. Hence, they cannot be directly adapted to HGs.

2.2 Heterogeneous Graph Neural Networks

Heterogeneous GNN models [9], [10], [11], [14], [20], [32], [33], [34], [35], [36], [37], [38] extend GNN techniques on HGs to fully use the rich node features and semantic information. According to the two basic properties of HGs, GNNs can be divided into two categories.

The first category focuses on modelling the heterogeneity of the ego graph. These models eliminate heterogeneity by modelling mapping functions to project heterogeneous nodes to the same feature space to apply GNNs. For example, HGT [14] introduces the node- and edge-type dependent attention mechanism to handle graph heterogeneity. HetSANN [38] uses edge-type specific transformation operations to project the hidden state in the space of source node type to the hidden space of the target node type. CompGCN [39] uses composition operations, such as TransE [21] or DistMult [40], to model the mapping function efficiently by learning edge representation vectors. Although they try to thoroughly model the heterogeneity to apply GNNs, the semantics property of HGs are not fully utilized. Moreover, since the order of semantic information is inconsistent, e.g., *APA* and *APCPA* in the DBLP dataset, simply stacking multiple layers to catch high-order semantic information will cause lower-order semantic information to face the over-smooth problem.

The second category focuses on semantics. These models explicitly utilize semantic information (captured by metapaths) to analyze the ego graph. Specifically, these methods are generally divided into two steps. *First*, the ego graph is decomposed by multiple metapaths to get local structures, which are further encoded to get node representations under each semantics. For example, as shown in Fig. 1c HAN [9] and HGSRec [19] model the local structure under each semantics as metapath-based neighbours. However, they discard all intermediate nodes along metapath. MAGNN [10] fixes that by aggregating information at the metapath instance level, i.e., sequence level. However, the graph structure of local structures is neglected, which causes information loss and inflexibility in the aggregation process. Similar to our model, several works leverage graph-like structures to model the local structure under each semantics, such as MEIRec [17], RecoGCN [18], T-GCN[20]. However, MEIRec neglects the feature information of intermediate and target nodes. In addition, they all ignore the modelling of mapping functions between heterogeneous nodes. *Second*, diverse semantics need to be exploited to get superior representations. Attention-based methods are widely used in these models [9], [10], [18], [19], [20] to softly select the most meaningful metapath and fuse the representations under different semantics. However, the final representations cannot be guaranteed theoretically superior to any single semantic representation.

In general, although the heterogeneous graph neural networks have made notable progress, these problems still hinders the performance of heterogeneous graph neural networks, and there is still much room for improvement by solving these problems.

3 Problem Definition

In this section, we define some important concepts, including the heterogeneous graph, metapath, and metapath-based ego graph. Then, we define the heterogeneous graph embedding problem and the versatility of multi-view representations. Frequently used notations are summarized in Table 1.

Definition 1. Heterogeneous Graph. *A heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}; \phi, \omega)$ is composed of a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$, along with object type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and edge type mapping function $\omega : \mathcal{E} \rightarrow \mathcal{R}$. $\mathcal{A}$ and $\mathcal{R}$ denote the predefined sets of object types and edge types, respectively, where $|\mathcal{A}| + |\mathcal{R}| > 2$. $\mathcal{R}$ could be further split into two subsets: $\mathcal{R}^+$ and $\mathcal{R}^-$. Note that $self$ loop relation can be randomly included in one of the two categories. An example is given in Fig. 1a.*

Definition 2. Metapath. *Consider $A_i \in \mathcal{A}$ and $R_i \in \mathcal{R}$ denote a node type and an edge type, respectively, a metapath P is defined as a path in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$, which describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between object types A_1 and A_{l+1}, where $\circ$ denotes the composition operator over relations. Examples are given in Fig. 1b.*

Definition 3. Metapath Based Ego Graph. *Given a metapath $P : A_1 \rightarrow A_2 \rightarrow ... \rightarrow A_N$ and a target node v with type A_N, the ego graph ($\mathcal{EG}_v^P$) is a directed graph induced by the*

TABLE 1
Important Notations Used in This Paper

Notations	Definitions
$\mathcal{V}$	The set of nodes in a graph
$\mathcal{A}$	The set of node types
$\mathcal{E}$	The set of edges in a graph
$\mathcal{R}$	The set of edge types, i.e., relations
$\mathcal{G}$	A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$
P	A metapath
$\mathcal{P}$	The set of metapaths
v	A node $v \in \mathcal{V}$
r	A kind of relation $r \in \mathcal{R}$
$\mathcal{N}_v$	The set of neighbors of node v
$\mathcal{EG}_v^P$	The ego graph of node v under metapath P
$\mathbf{h}_v$	Hidden state of node v
$\mathbf{h}_r$	Hidden state of relation r
$\mathbf{H}$	Hidden states of nodes with the same node type
$\odot$	The hadamard product
$\lvert\cdot\rvert$	The cardinality of a set
$\lVert\cdot\rVert_F$	Frobenius norm of a matrix
$\lVert\cdot\rVert_1$	L1 norm of a matrix

metapath-based neighborhoods and intermediate nodes along metapaths as well as v itself. An example is shown in Fig. 1c.

Definition 4. Heterogeneous Graph Embedding.*Heterogeneous graph embedding aims to learn a function that embeds the nodes in $\mathcal{G} = (\mathcal{V}, \mathcal{E}; \phi, \omega)$ into a d-dimensional euclidean space where $d \ll |\mathcal{V}|$.*

Definition 5. Versatility of Multi-View Representations. [23] *Given representations $\mathbf{h}^1, \ldots, \mathbf{h}^V$ from V views, the multi-view representation $\mathbf{h}$ is of versatility if $\forall\, v$ and $\forall$ mapping $\varphi(\cdot)$ with $y^v = \varphi(\mathbf{h}^v)$, there exists a mapping $\psi(\cdot)$ satisfying $y^v = \psi(\mathbf{h})$.*

4 The Proposed Framework

In this section, we present the MV-HetGNN, which sufficiently models the heterogeneity and semantics in the complex local structure in HGs from the perspective of multi-view representation learning. As shown in Fig. 2, MV-HetGNN contains three primary steps. First, since different node types are associated with features of different dimensions, the node feature transformation converts them into features of the same dimension. Second, we treat each semantic (captured by the metapaths) P as the view to observe target nodes' local structure and conduct view-specific ego graph encoder to obtain the node embeddings under each view. Finally, the auto multi-view fusion layer comprehensively integrates the embeddings from different views to obtain versatile node embeddings.

4.1 Node Feature Transformation

Feature vectors with different dimensions are troublesome when aggregating information in subsequent modules. To address this issue, we apply node-type specific transformations to convert heterogeneous feature vectors into features of the same dimension. Given any node type $A_i \in \mathcal{A}$, the node feature transformation layer can be shown as follows:

$$\mathbf{H}'_{A_i} = \sigma(\mathbf{X}_{A_i}\mathbf{W}_{A_i}), \tag{1}$$

where $\mathbf{H}'_{A_i} \in \mathbb{R}^{|\mathcal{V}_{A_i}| \times d'}$ is the transformed latent vectors of nodes of A_i type. $\mathbf{X}_{A_i} \in \mathbb{R}^{|\mathcal{V}_{A_i}| \times d_{A_i}}$ is the original feature vector of all A_i nodes. $\mathbf{W}_{A_i} \in \mathbb{R}^{d_{A_i} \times d'}$ is the type-specific parameters, and d' is the unified dimension. σ means the activation function, such as ReLU [41].

The node feature transformation unifies the feature dimensions and facilitates the usage of subsequent modules. However, it should be noted that the heterogeneity has not been resolved. Although the dimensions of the feature vector of all nodes are the same, they still lie in different embedding spaces, so they should not be aggregated directly.

4.2 View-Specific Ego Graph Encoder

Considering the *heterogeneity* and *semantics* of HGs, the local structure of the target node in HGs is rather complex. This

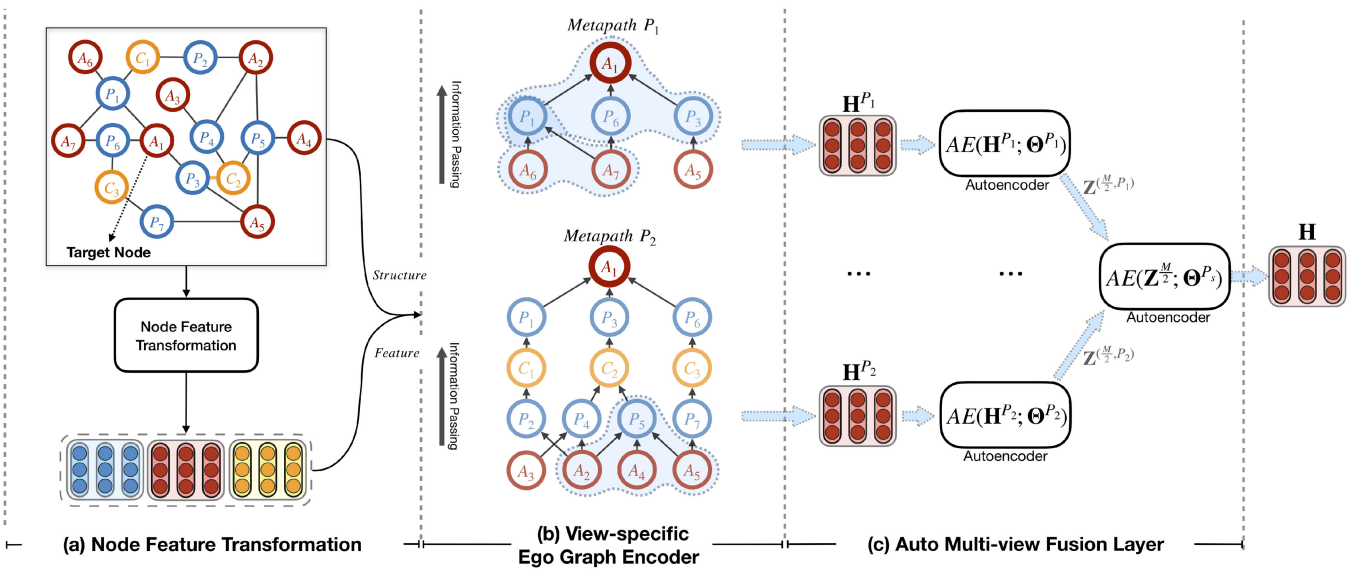


Fig. 2. The overall architecture of MV-HetGNN. (a) Node feature transformation projects the feature vectors of all types of nodes to the same dimension. (b) View-specific ego graph encoder generates embedding for a target node under each view, such as A_1 in the picture above. (c) Auto multi-view fusion layer integrates multi-view embeddings by hierarchical autoencoders to obtain more versatile node embeddings.

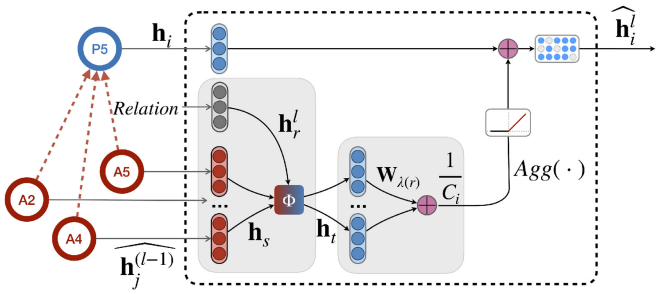


Fig. 3. Information aggregation for node $P5$ at level l. The Author node's feature vector ($\mathbf{h}_s$), relation vector ($\mathbf{h}_r^{(l)}$), and $P5$'s feature vector ($\mathbf{h}_i$) are aggregated to obtain new encoded feature vector for $P5$.

module aims to model the local structure under each semantics adequately and aggregate information with addressing the heterogeneity.

Specifically, we treat each semantic P as the view to observe the target node's local structure. Specifically, the original complete ego graph is decomposed into multiple view-specific ego graphs. The advantage of this operation is that multiple semantics can be decoupled. Furthermore, compared to metapath-based neighborhoods [9] and metapath instances [10], the view-specific ego graph preserves more complete local structure under each semantics. Another benefit is that the view-specific ego graph has better characteristics: there is only one type of node in the same order, and there is only one type of relation between different orders. Given a metapath $P = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{N-1}} A_N$ and a target node v with $\phi(v) = A_N$, the view-specific ego graph $\mathcal{EG}_v^P$ preserves the original and complete multi-hop structure under semantic P, where the nodes of the ith hop have the same node type A_{N-i} and the relation between the ith hop and the $(i-1)$th hop ($i \geq 1$) is R_{N-i}. An example is shown in Fig. 1c.

Further, in order to handle the heterogeneity and encode the structural and feature information adequately from the ego graph, we develop an ego graph encoder, which has two advantages. First, it can aggregate node features guided by the structure of the view-specific ego graph. Second, it handles the heterogeneity in the ego graph by modeling the representations of relations and the mapping function between heterogeneous nodes. Two example of view-specific ego graph encoder is illustrated in Fig. 2b. Assuming that level 1 is the bottom level and level K is the top level, ego graph encoder aggregates information from level 1 to K, updating the representation $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d'}$ of each node i at level l. The aggregation process (blue region in Fig. 2b) between any intermediate two levels is illustrated in Fig. 3. Specifically, we set the hidden state of node i at level 1 as $\mathbf{h}_i^{(1)} = \mathbf{h}_i'$, which is generated by the node feature transformation module. For any specific node i at level l ($l \neq 1$), we first aggregate its neighborhood information and then apply an activation function. Combined with its own features, the encoded feature is calculated as

$$\mathbf{h}_i^{(l)} = o(\mathbf{h}_i' + \sigma(\mathbf{Agg}(\{\mathbf{h}_j^{(l-1)}, v_j \in \mathcal{N}_i^{(l-1)}\}, \mathbf{h}_r^l))), \tag{2}$$

where $\mathbf{h}_i^{(l)} \in \mathbb{R}^{d'}$ is the encoded hidden vector of node i at level l, and $\mathbf{h}_i' \in \mathbb{R}^{d'}$ is its initial state generated by the component of node content transformation. $\mathcal{N}_i^{(l-1)}$ is the neighborhood set of node i at level $(l-1)$, $\mathbf{h}_j^{(l-1)} \in \mathbb{R}^{d'}$ is the encoded hidden vector of node j at level $(l-1)$, $\mathbf{h}_r^l$ is the representation of the relation r between level l and $(l-1)$. $\sigma(\cdot)$ is the ReLU activation function. Note that the representation is randomly initialized and optimized jointly with the network parameters. o is the dropout layer. $\mathbf{Agg}(\cdot)$ is an aggregate function to capture 1-hop neighborhoods information and relation representation:

$$\begin{aligned}&\mathbf{Agg}(\{\mathbf{h}_j^{(l-1)}, v_j \in \mathcal{N}_i^{(l-1)}\}, \mathbf{h}_r^l)\\ &= \tfrac{1}{C_i} \textstyle\sum_{v_j \in \mathcal{N}_i^{(l-1)}} \mathbf{W}_{\lambda(r)} \Phi(\mathbf{h}_j^{(l-1)}, \mathbf{h}_r^l),\end{aligned} \tag{3}$$

where $\Phi(\cdot,\cdot)$ is the mapping function for handling the heterogeneity, $\mathbf{W}_{\lambda(r)} \in \mathbb{R}^{d' \times d'}$ is the relation-specific message passing parameter, C_i is the normalization term. For simplicity, we set $C_i = |\mathcal{N}_i^{(l-1)}|$, i.e., mean average, in our method. Kindly note that with the benefit of the graph structure, we can further improve the performance by setting C_i to be learnable, e.g., through attention mechanisms. Related experiments can be found in Section 5.3.

For one thing , benefiting from the unified dimensions of all types of nodes, we can model the relations $\mathbf{h}_r^{(l)}$ as $\mathbb{R}^{d'}$ vectors and use the knowledge graph embedding approach to model the mapping function between the feature vectors of different types of nodes. Many functions can be adopted here and we apply TransE [21], which gains an impressive performance and efficiency in our experiments. In TransE [21], for a relation triplet (s, r, t), there will be $\mathbf{s} + \mathbf{r} \approx \mathbf{t}$. Therefore:

$$\Phi(\mathbf{h}_s, \mathbf{h}_r) = \mathbf{h}_s + \mathbf{h}_r. \tag{4}$$

For another thing, the message passing parameter $\mathbf{W}_{\lambda(r)}$ would suffer from the over-parameterization problem with the growth of the number of relations, since each relation r is associated with a matrix $\mathbf{W}_{\lambda(r)}$. Inspired by [39], [42], we simplify it to a direction-specific matrix, i.e., $\lambda(r) = dir(r)$, which is defined as follows:

$$\mathbf{W}_{\lambda(r)} = \mathbf{W}_{dir(r)} = \begin{cases} \mathbf{W}_O, & r \in \mathcal{R}^+ \\ \mathbf{W}_I, & r \in \mathcal{R}^- \end{cases} s, \tag{5}$$

where $\mathcal{R}^+$ and $\mathcal{R}^-$ are opposite sets of two relations. For example, $\mathcal{R}^+ = \{write, publish\}$ and $\mathcal{R}^- = \{written-by, published-in\}$. Note that the *self loop* relation can be randomly included in one of the two categories.

After $K-1$ times of calculation on ego graph $\mathcal{EG}_v^P$, the representation of the target node v is $\mathbf{h}_v^{(K)}$. Note that the lengths of the metapaths are inconsistent, which results in different times of aggregation of Eq. (2).

$$\mathbf{h}_v^P = \frac{\mathbf{h}_v^{(K)}}{depth(\mathcal{EG}_v^P)} \tag{6}$$

where $depth(\mathcal{EG}_v^P)$ is the depth of the view-specific ego graph, which equals to the length of P.

In summary, given the features generated by the node feature transformation and the view set $\mathcal{P}_{A_i} = \{P_1, \ldots, P_{|\mathcal{P}_{A_i}|}\}$ where the metapath start or end with the node type $A_i \in \mathcal{A}$, the view-specific ego graph encoder will

generate a set of representations under each view for node $v \in \mathcal{A}_i$, denoted as $\{\mathbf{h}_v^{P_1}, \mathbf{h}_v^{P_2}, \ldots, \mathbf{h}_v^{P_{|\mathcal{P}_{A_i}|}}\}$.

4.3 Auto Multi-View Fusion Layer

In this section, we aim to fuse diverse representations from multiple views. There are two key challenges. First, the fused representation requires a theoretical guarantee of a crucial characteristic, versatility [23] (Definition 5), which means the fused representation can perform at least as good as any single-view representation. Popular approaches in existing HG embedding works, such as attention-based approaches [9], [10], [15], [16], can not hold the versatility. Second, there is much redundant information among multi-view representations, which must be removed in the fusion process [22]. The simple concatenation operation preserves the redundant information, leading to poor performance, even if it holds the versatility intuitively. The auto multi-view fusion layer addresses these two challenges through hierarchical autoencoders and orthogonal regularization.

4.3.1 Hierarhical Autoencoders

Given a metapath P_j in view set $\mathcal{P}_{A_i}$, we denote the embeddings of all nodes generated by view-specific ego graph encoder as $\mathbf{H}^{P_j} = [\mathbf{h}_1^{P_j}, \mathbf{h}_2^{P_j}, \cdots, \mathbf{h}_{|\mathcal{V}_{A_i}|}^{P_j}]^T \in \mathbb{R}^{|\mathcal{V}_{A_i}| \times d'}$. The representations from all views $\{\mathbf{H}^{P_1}, \mathbf{H}^{P_2}, \cdots, \mathbf{H}^{P_{|\mathcal{P}_{A_i}|}}\}$ need be encoded into versatile representations $\mathbf{H}$. In order to do that, hierarchical autoencoders are used to preserve intra-view information and encode inter-view information simultaneously. We first demonstrate a standard autoencoder as $AE(\mathbf{X}; \mathbf{\Theta})$, where $\mathbf{X}$ is the input of autoencoder and $\mathbf{\Theta} = \{\mathbf{W}_{ae}^{(m)}, \mathbf{b}_{ae}^{(m)}\}|_{m=1}^{M}$ is the network parameters with M being the number of layers. The first $\frac{M}{2}$ layers are the encoder network, and the last $\frac{M}{2}$ layers are the decoder network. Let $\mathbf{Z}^{(0)} = \mathbf{X}$, then the output of the mth layer is:

$$\mathbf{Z}^{(m)} = \sigma(\mathbf{Z}^{(m-1)}\mathbf{W}_{ae}^{(m)} + \mathbf{b}_{ae}^{(m)}), \tag{7}$$

where $\mathbf{Z}^{(m)} \in \mathbb{R}^{\cdot \times d^m}$. d_m is the output dimension of m-th layer. $\mathbf{W}_{ae}^{(m)} \in \mathbb{R}^{d^{m-1} \times d^m}$ and $\mathbf{b}_{ae}^{(m)} \in \mathbb{R}^{d^m}$ denote the weights and bias of the mth layer, respectively. $\sigma(\cdot)$ is the non-linear activation, such as ReLU [41]. For simplicity, we denote $f(\cdot; \mathbf{\Theta}_e)$ as the encoder network where parameters $\mathbf{\Theta}_e = \{\mathbf{W}_{ae}^{(m)}, \mathbf{b}_{ae}^{(m)}\}|_{m=1}^{\frac{M}{2}}$ and $f(\cdot; \mathbf{\Theta}_d)$ as the decoder network, where parameters $\mathbf{\Theta}_d = \{\mathbf{W}_{ae}^{(m)}, \mathbf{b}_{ae}^{(m)}\}|_{m=\frac{M}{2}}^{M}$. $f(\cdot)$ denotes general the multilayer perceptron architecture.

First, in intra-view, we use view-specific autoencoders to further compress the representations of each view to be more compact. Assuming the autoencoder under view P_j is $AE(\mathbf{H}^{P_j}; \mathbf{\Theta}^{P_j})$, then the compressed representations, i.e., the output of the encoder network is:

$$\mathbf{Z}^{(\frac{M}{2}, P_j)} = f(\mathbf{H}^{P_j}; \mathbf{\Theta}_e^{P_j}). \tag{8}$$

The reconstruction of the decoder network is:

$$\mathbf{Z}^{(M, P_j)} = f(\mathbf{Z}^{(\frac{M}{2}, P_j)}; , \mathbf{\Theta}_d^{P_j}). \tag{9}$$

Therefore, the reconstruction loss for all view-specific autoencoders is:

$$\mathcal{L}_{re}^{intra} = \frac{1}{2}\sum_{P_j} ||\mathbf{H}^{P_j} - \mathbf{Z}^{(M, P_j)}||_F^2. \tag{10}$$

Second, in inter-view, we use a 2-layer supervised autoencoder [43] to encode all information into a latent representation comprehensively. We first concatenate the compressed representations from all views, denoted as $\mathbf{Z}^{\frac{M}{2}} = [\mathbf{Z}^{(\frac{M}{2}, P_1)}, \ldots, \mathbf{Z}^{(\frac{M}{2}, \mathcal{P}_{A_i})}]$. Assuming the supervised autoencoder network is $AE(\mathbf{Z}^{\frac{M}{2}}; \mathbf{\Theta}^s)$, where $\mathbf{\Theta}^s = \{\mathbf{W}_s^1, \mathbf{W}_s^2, \mathbf{b}_s^1, , \mathbf{b}_s^2\}$ and $\mathbf{W}_s^1 \in \mathbb{R}^{(d^{\frac{M}{2}} \cdot |\mathcal{P}_{A_i}|) \times d}$ and $\mathbf{W}_s^2 \in \mathbb{R}^{d \times (d^{\frac{M}{2}} \cdot |\mathcal{P}_{A_i}|)}$ and $\mathbf{b}_s^1 \in \mathbb{R}^d$ and $\mathbf{b}_s^2 \in \mathbb{R}^{d^{\frac{M}{2}} \cdot |\mathcal{P}_{A_i}|}$, then the output of the encoder network is:

$$\mathbf{H} = f(\mathbf{Z}^{\frac{M}{2}}; \mathbf{\Theta}_e^s) = \sigma(\mathbf{Z}^{\frac{M}{2}}\mathbf{W}_s^1 + \mathbf{b}_s^1). \tag{11}$$

The reconstruction of the decoder network is:

$$\mathbf{Z}_{re}^{\frac{M}{2}} = f(\mathbf{H}; \mathbf{\Theta}_d^s) = \sigma(\mathbf{H}\mathbf{W}_s^2 + \mathbf{b}_s^2). \tag{12}$$

Therefore, the reconstruction loss for the supervised autoencoders is:

$$\mathcal{L}_{re}^{inter} = \frac{1}{2}||\mathbf{Z}_{re}^{\frac{M}{2}} - \mathbf{Z}^{\frac{M}{2}}||_F^2. \tag{13}$$

The supervised information is the downstream task loss and will be discussed in Section 4.4.

Ideally, minimizing Eq. (10) and Eq. (13) will offer versatility to the multi-view representations $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}_{A_i}| \times d}$, i.e., the final representations of all nodes of type A_i. Here we give a brief theoretical proof:

Proposition 1. (Versatility of the Multi-View Representation $\mathbf{H}$) *There exists a solution to Eq. (13) and Eq. (10) which holds the versatility.*

Proof. First, we partition the weight parameter matrix of the decoder of the 2-layer supervised autoencoder as:

$$\mathbf{W}_s^2 = [(\mathbf{W}_s^2)^{P_1}, \cdots, (\mathbf{W}_s^2)^{P_{|\mathcal{P}_{A_i}|}}], \tag{14}$$

where $(\mathbf{W}_s^2)^{P_j} \in \mathbb{R}^{|\mathcal{V}_{A_i}| \times d^{\frac{M}{2}}}$. According to Eq. (13), it is easy to show that there exists $\mathbf{Z}^{(\frac{M}{2}, P_j)} = \sigma(\mathbf{H}(\mathbf{W}_s^2)^{P_j} + \mathbf{b}_s^2) = f(\mathbf{H}; \mathbf{\Theta}_d^{(s, P_j)})$, where $f(\cdot; \mathbf{\Theta}_d^{(s, P_j)})$ is the mapping from the multi-view representation $\mathbf{H}$ to the compressed single view representation $\mathbf{Z}^{(\frac{M}{2}, P_j)}$. Further, according to Eq. (10), there exists $\mathbf{H}^{P_j} = f(\mathbf{Z}^{(\frac{M}{2}, P_j)}; \mathbf{\Theta}_d^{P_j})$. Consequently, there exists:

$$\mathbf{H}^{P_j} = f(f(\mathbf{H}; \mathbf{\Theta}_d^{(s, P_j)}); \mathbf{\Theta}_d^{P_j}\}) = f(\mathbf{H}; \{\mathbf{\Theta}_d^{(s, P_j)}, \mathbf{\Theta}_d^{P_j}\}), \tag{15}$$

where $f(\cdot; \{\mathbf{\Theta}_d^{(s, P_j)}, \mathbf{\Theta}_d^{P_j}\})$ is the mapping from $\mathbf{H}$ to $\mathbf{H}^{P_j}$. Hence, $\forall\ \varphi(\cdot)$ with $\mathbf{Y}^{P_j} = \varphi(\mathbf{H}^{P_j})$, there exists a mapping $\psi(\cdot)$ satisfying $\mathbf{Y}^{P_j} = \psi(\mathbf{H})$ by defining $\psi(\cdot) = \varphi(f(\cdot; \{\mathbf{\Theta}_d^{(s, P_j)}, \mathbf{\Theta}_d^{P_j}\}))$. □

4.3.2 Orthogonal Regularization

There exist two issues when applying hierarchical autoencoders. First, the representations from different views contain much redundant information as mentioned before. For example, the *Co-author* view and *Co-conference* view may share many paper and author nodes. Second, the

autoencoders may suffer from the over-parameterization problem. Since the graph datasets are often semi-supervised where only a few labels can be accessed, the optimization of autoencoders might be problematic. In order to address these issues, we introduce orthogonal regularization to the encoder network of each autoencoder. On the one hand, the orthogonal column vectors can be regarded as orthogonal bases, ensuring each encoded dimension represents a unique meaning, i.e., independent of the other dimensions [44]. On the other hand, due to the orthogonality constraint, only informative weights are non-zero.

Specifically, considering all the first $\frac{M}{2}$ layers of all autoencoders ae, the orthogonal loss is:

$$\mathcal{L}_{ortho} = \sum_{ae} \sum_{m}^{\frac{M}{2}} ||(\mathbf{W}_{ae}^{m})^{T}(\mathbf{W}_{ae}^{m}) \odot (\mathbf{1}_{d_m} - \mathbf{I}_{d_m})||_1, \tag{16}$$

where $\odot$ refers to the element-wise product. We do not set orthogonal constraints on the decoder networks because the decoder needs to reconstruct the original hidden state that may contain many redundant features.

4.4 Optimization

In MV-HetGNN, we adopt a task-guided training strategy to optimize the network parameters. There are several strategies to optimize MV-HetGNN. For example, optimize view-specific ego graph encoder and auto multi-view fusion layer sequentially according to the downstream tasks. However, for the scalability and flexibility of the model, we optimize the MV-HetGNN in an *end-to-end* paradigm.

Algorithm 1. The Overall Learning Algorithm of MV-HetGNN

Input: The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the node features $\{\mathbf{H}_{A_i}, \forall A_i \in \mathcal{A}\}$, the view (metapath) set $\mathcal{P}$, the ego graphs $\mathcal{EG}_v^P$ of each node v under each view P.

Output: The node Embeddings $\mathbf{H}$.

```
for node type A_i ∈ A do
    Node feature transformation: H'_{A_i} = σ(X_{A_i} W_{A_i});
end
for metapath P_j in P do
    for node v ∈ V do
        Given EG_v^{P_j}, calculate the view-specific representation h_v^{P_j}
        by the view-specific ego graph encoder;
    end
end
Calculate multi-view representation H by Eq. (8, 11);
Calculate the reconstruction loss by Eq. (10, 13);
Calculate the orthogonal regularization loss by Eq. (16);
Backpropagation and update parameters according to Eq. (19);
```

Specifically, according to the node labels' availability of downstream tasks, we divide the them into two categories: semi-supervised and unsupervised downstream tasks. For semi-supervised learning (e.g., node classification and node clustering), we first define the labeled node set as $\mathcal{Y}_L$ and the the downstream classifier as $\mathbf{W}_C \in \mathbb{R}^{C \times d}$, where C is the number of classes. Then the downstream loss is:

$$\mathcal{L}_{ds} = -\sum_{v \in \mathcal{Y}_L} \mathbf{y}_v \ln(\mathbf{W}_C \mathbf{h}_v^T), \tag{17}$$

where the $\mathbf{h}_v \in \mathbb{R}^{1 \times d}$ and $\mathbf{y}_v \in \mathbb{R}^{1 \times C}$ is the embedding vector and one-hot label vector of labeled node v. For unsupervised learning (e.g., link prediction), we design the loss function of preserving the graph structure, i.e., the adjacency relationship of nodes. We optimize the model parameters by minimizing the following loss function through negative sampling [45]:

$$\mathcal{L}_{ds} = -\sum_{(u,v) \in \Omega} \log \sigma(\mathbf{h}_u^T \cdot \mathbf{h}_v) - \sum_{(u',v') \in \Omega^-} \log \sigma(-\mathbf{h}_{u'}^T \cdot \mathbf{h}_{v'}), \tag{18}$$

where $\sigma(\cdot)$ is the sigmoid function, Ω is the set of positive node pairs, and Ω^- is the sampled set of negative node pairs sampled from all unobserved node pairs.

Therefore, the complete loss function of MV-HetGNN is:

$$\mathcal{L} = \mathcal{L}_{ds} + \lambda(\mathcal{L}_{re}^{intra} + \mathcal{L}_{re}^{inter}) + \mathcal{L}_{ortho}, \tag{19}$$

where λ is a critical hyper-parameter that controls the degree of versatility. It is often a relatively small number, such as 0.1, 0.05. This is mainly because: i) At the beginning of training, the representation of each view is initialized almost randomly, so the reconstruction loss (i.e., loss of generality) is meaningless. Therefore, in order to optimize the network parameter in the right direction, the downstream task loss needs to play a major role. ii) In practical cases, it is usually difficult to guarantee the exact versatility. Moreover, exactly versatile multi-view representation will loss flexibility for various datasets and downstream tasks.

We optimize the model parameters by minimizing $\mathcal{L}$ via backpropagation and gradient descent. The overall learning algorithm is shown in Algorithm 1.

5 Experiments

5.1 Experimental Setup

Dataset. As shown in Table 2, three heterogeneous graph datasets from different domains are used to evaluate the performance of MV-HetGNN. Following [9], [10], we use three widely used evaluation tasks: node classification, node clustering, and link prediction.

- *DBLP* [2]: We adopt a subset of DBLP extracted by [46]. The authors are divided into four areas. DBLP is used for node classification and node clustering with data partition of 400 (9.86%), 400 (9.86%), and 3257 (80.28%) for training, validation, and testing.
- *IMDb* [3]: We adopt a subset of IMDb extracted by [10]. Each movie is labeled as one of three classes. IMDb is used for both node classification and node clustering with data partition of 400 (9.35%), 400 (9.35%), and 3478 (81.30%) nodes for training, validation, and testing, respectively.
- *Last.fm* [4]: We adopt a subset of Last.fm released by [47]. No label or features are included. Last.fm dataset is used for the link prediction task. All links are divided into training, validation, and testing set with data partition of 18567 (20%), 9283 (10%), and 64984 (70%), respectively.

2. https://dblp.uni-trier.de/
3. https://www.imdb.com/
4. https://www.last.fm/

TABLE 2
Statistics of Datasets

Dataset	# Node	# Edge	Metapaths
DBLP	Author(A): 4,057 Paper(P): 2,081 Term(T): 7,723 Conference(C): 20	A-P: 19,645 P-T: 85,810 P-C: 14,328	APA APTPA APCPA
IMDb	Movie(M): 4,278 Director(D): 2,081 Actor(A): 5,257	M-D: 4,278 M-A: 12,828	MDM, MAM DMD, DMAMD AMA, AMDMA
Last.fm	User(U): 1,892 Artist(A): 17,632 Tag(T): 1,088	U-U: 12,717 U-A: 92,834 A-T: 23,253	UU, UAU UATAU, AUA AUUA, ATA

Baselines. We compare MV-HetGNN with the following baselines:

- *Metapath2vec* [48] uses metapath-guided random walks and skip-gram model [49] to generates node embeddings. We test Metapath2vec++ [48] on all metapaths separately and report the best results.
- *HERec* [50] applies DeepWalk [51] on multiple metapath-based homogeneous graphs, and proposes an fusion algorithm for rating prediction. For node classification/clustering, we report the best result of all metapaths.
- *GCN* [6] performs convolutional operations in the graph Fourier domain.
- *GAT* [26] conducts convolutional operations in the spatial domain with the attention mechanism. We test GCN and GAT on metapath-based homogeneous graphs and report the best results.
- *HAN* [9] learns node embeddings from different metapath-based homogeneous graphs and leverages the attention mechanism to combine them into one vector for each node.
- *HGT* [14] designs node- and edge-type dependent parameters to characterize the heterogeneous attention over each edge to model heterogeneity.
- *MAGNN* [10] uses intra-metapath aggregation and inter-metapath aggregation to encode metapath instances, and learn the importance of different metapaths by an attention mechanism.

The above baselines can be divided into four categories. Metapath2vec and HERec are the shallow heterogeneous graphs embedding models. GCN and GAT are conventional GNNs designed for homogeneous graphs. HGT is the first category of heterogeneous graph neural networks (as described in Section 2.2). HAN and MAGNN are the second category of heterogeneous graph neural networks.

Implementation. For Metapath2vec and HERec, we set the window size to 5, walk length to 100, walks per node to 40, and the number of negative samples to 5. For all GNNs, we set the dropout rate to 0.5 as default. For HAN, MAGNN, and HGT, we follow the original setting reported in these papers. For MV-HetGNN, we employ the Adam optimizer, and set the learning rate to 0.005, 0.001, 0.001 for DBLP and IMDb and Last.fm dataset, respectively. We set the number of layers of view-specific auto encoder M as 2. For the dimension hyper-parameters d' (output feature dimension of node feature transformation), $d^{\frac{M}{2}}$ (the output dimension of the encoder of view-specific autoencoder) and d (the dimension of multi-view representations), we set $d' = 2d^{\frac{M}{2}} = 2d$ for DBLP, and $d' = d^{\frac{M}{2}} = d$ for IMDb and $d' = d^{\frac{M}{2}} = d$ for Last.fm datasets, respectively.

5.2 Experimental Results

Node Classification. We conduct node classification experiments [9], [10] on the DBLP and IMDb datasets, with about 10% data used for training. After obtaining embeddings of labeled data by each model, we feed the testing nodes into a linear support vector machine (SVM) classifier with varying training proportions. Since the variance of graph-structured data can be relatively high, we repeat it ten times and report the averaged *Macro-F1* and *Micro-F1* in Table 3.

As shown in the table, the bold and underlined numbers indicate the best and runner-up results in the row, respectively.

TABLE 3
Results (%) on the DBLP and IMDb Datasets for Node Classification Task

Dataset	Metrics	Train%	Unsupervised		Semi-supervised					
			Metapath2vec	HERec	GCN	GAT	HAN	HGT	MAGNN	MV-HetGNN
IMDb	Macro-F1	20%	45.94	45.31	53.63	54.74	57.52	59.38	59.48	**61.33**
		40%	47.41	46.63	53.86	56.27	57.81	59.91	59.79	**61.43**
		60%	48.23	47.07	54.22	56.97	58.28	60.32	60.02	**61.39**
		80%	50.34	48.02	54.77	57.43	58.69	60.38	60.20	**61.89**
	Micro-F1	20%	47.47	46.19	53.61	54.56	57.79	59.42	59.27	**61.31**
		40%	48.69	48.03	53.88	56.17	58.77	60.08	59.92	**61.43**
		60%	49.54	48.41	54.19	56.89	59.11	60.27	60.14	**61.36**
		80%	50.47	49.57	54.12	57.48	59.57	60.44	60.21	**61.89**
DBLP	Macro-F1	20%	89.39	90.31	90.00	91.37	91.87	92.05	93.01	**95.23**
		40%	89.99	91.15	90.11	91.70	92.36	92.57	93.24	**95.31**
		60%	90.31	92.01	90.12	91.76	92.80	92.90	93.52	**95.34**
		80%	90.94	92.37	91.07	93.81	93.01	93.40	93.79	**95.44**
	Micro-F1	20%	90.43	91.49	90.03	91.89	92.48	92.55	93.51	**95.52**
		40%	90.99	92.05	90.31	92.17	92.90	93.08	93.74	**95.64**
		60%	91.33	92.66	90.31	92.32	93.35	93.38	94.01	**95.56**
		80%	91.61	92.78	90.40	92.36	93.53	93.46	94.17	**95.80**

TABLE 4
Results (%) on the DBLP and IMDb Datasets for Node Clustering Task

Dataset	Metrics	Unsupervised		Semi-supervised					
		metapath2vec	HERec	GCN	GAT	HAN	HGT	MAGNN	MV-HetGNN
IMDb	NMI	0.93	0.41	8.10	9.98	12.46	14.50	14.41	**16.04**
	ARI	0.32	0.17	6.62	9.02	11.21	15.92	15.22	**17.67**
DBLP	NMI	74.09	69.31	72.75	75.03	77.86	77.47	80.52	**84.23**
	ARI	78.32	72.71	73.13	81.73	83.23	81.84	85.68	**89.05**

MV-HetGNN consistently achieves the best performance. The shallow models (Metapath2vec and HERec) perform worse than GCN or GAT, since they do not leverage node content features. HAN obtains better performance than GAT and GCN because it exploits multiple metapaths to explore various semantics. HGT and MAGNN are able to outperform HAN because they can utilize more node features by stacking multi-layer or metapath encoder, respectively. MV-HetGNN consistently outperforms HGT, which stacks multiple layers to catch high-order semantics. In contrast, MV-HetGNN employs the view-specific ego graph encoder to utilize higher-order information effectively. MV-HetGNN also obtains better results than MAGNN. There are two main reasons. For one thing, MV-HetGNN model the metapath-based local structure more comprehensively than MAGNN. For another thing, MV-HetGNN can comprehensively integrate the embeddings from different views to obtain more versatile embeddings than MAGNN.

Node Clustering. We conduct node clustering experiments on the DBLP and IMDb datasets, using the same setting as [9], [10]. We feed the embeddings of labeled nodes to a K-Means algorithm. The number of cluster K is set to 3 for IMDb and 4 for DBLP. Since the clustering result of the K-Means algorithm is highly dependent on the initialization of the centroids, we repeat K-Means 10 times and report the averaged *normalized mutual information* (NMI) and *adjusted Rand index* (ARI).

The results are reported in Table 4. Overall, the relative performance of node clustering task is similar to the node classification task. MV-HetGNN significantly performs much better than all baselines consistently. The experimental results demonstrate that MV-HetGNN is able to learn more effective representation for the nodes of heterogeneous graphs. We note that the performance of all evaluated models on IMDb is much worse than on DBLP because every movie node has multiple genres in the original IMDb dataset, but only the first one is chosen as its class label.

Link Prediction. We evaluate the performance of link prediction task on Last.fm, following the MAGNN model [10]. Compared to MAGNN [10], we adopt a lower training ratio (only 20% links are used for training), which is a more challenging setting. The connected user-artist pairs are treated as positive links, while unconnected user-artist pairs are regarded as negative links.

We add the same number of randomly sampled negative node pairs to the validation and testing sets. The GNNs are then optimized by minimizing Equation 18. Given the user embedding $\mathbf{h}_u$ and the artist embedding $\mathbf{h}_a$ generated by the trained model, the linking probability of u and a is calculated by $Prob_{ua} = \sigma(\mathbf{h}_u^T \cdot \mathbf{h}_a)$, where σ is the commonly used sigmoid function. The embedding models are evaluated by the *area under the ROC curve (AUC)* and *average precision* (AP) scores. In Table 5, we report the averaged results of 10 runs of each embedding model. MV-HetGNN outperforms all baseline models, indicating the superiority of MV-HetGNN. Compared with other methods, MV-HetGNN can effectively obtain high-order information, and comprehensively use the features and semantic information.

5.3 Study on View-Specific Ego Graph Encoder

In this section, we conduct extensive experiments to evaluate the effectiveness of view-specific ego graph encoder. First of all, we compare the view-specific ego graph encoder with other local structure encoding methods, including metapath-based neighborhoods encoder used in HAN [9] and metapath instance encoder used in MAGNN [10]. Specifically, we replace the view-specific ego graph encoder with the two approaches mentioned above, resulting in two variants, MV-HetGNN$_{\mathrm{HAN}}$ and MV-HetGNN$_{\mathrm{MAGNN}}$. Second, we evaluate the impact of modeling the representations of relations and the mapping function between heterogeneous nodes by a variant MV-HetGNN$_{\mathrm{w/o\ TransE}}$, which removes the TransE mapping function and aggregates information directly. Finally, benefiting from modeling the local structure under each semantics as graph structure, we can further improve the performance of view-specific ego graph encoder by setting $\frac{1}{C_i}$ in Eq. (3) learnable. Specifically, we learn $\frac{1}{C_i}$ by graph attention mechanism [26], resulting in a variant MV-HetGNN$_{\mathrm{GAT}}$. Due to space limitations, we omit the details of these variants. Interested readers can refer to their original papers [9], [10], [26].

The results are shown in Table 6. MV-HetGNN consistently outperforms MV-HetGNN$_{\mathrm{HAN}}$ and MV-HetGNN$_{\mathrm{MAGNN}}$, validating the superiority of the view-specific ego graph encoder. Furthermore, the results of MV-HetGNN$_{\mathrm{w/o\ TransE}}$ indicate that modeling the mapping function plays a positive role in HGs embedding. Last but not least, MV-HetGNN$_{\mathrm{GAT}}$ outperforms MV-HetGNN in node classification and link prediction tasks, which indicates that the view-specific graph modeling provides us with more flexibility to design models.

5.4 Study on Auto Multi-View Fusion Layer

In this section, we aim to evaluate the effectiveness of the auto multi-view fusion layer. First of all, we replace it with other popular metapath fusion methods, including simple concatenation, mean pooling, and the attention mechanism [9], [10], [15], [16], [20], resulting in three variants, MV-HetGNN$_{\mathrm{concat}}$, MV-HetGNN$_{\mathrm{mean}}$, and MV-HetGNN$_{\mathrm{attn}}$. Specifically, the MV-HetGNN$_{\mathrm{concat}}$ simply concatenates the representations from multiple views without any transformation, the MV-HetGNN$_{\mathrm{mean}}$ conducts mean pooling on these representations,

TABLE 5
Results (%) on the Last.fm Datasets for Link Prediction Task

Dataset	Metrics	metapath2vec	HERec	GCN	GAT	HAN	HGT	MAGNN	MV-HetGNN
Last.fm	AUC	74.32	73.98	76.59	80.03	81.00	86.53	87.68	**92.80**
	AP	74.11	72.53	75.51	81.44	82.03	88.77	89.25	**94.03**

TABLE 6
Results (%) of the Study on View-Specific Ego Graph Encoder and Auto Multi-View Fusion Layer on Three Datasets

Variants	IMDb				DBLP				Last.fm	
	Macro-F1	Micro-F1	NMI	ARI	Macro-F1	Micro-F1	NMI	ARI	AUC	AP
MV-HetGNN	61.51	61.50	**16.04**	**17.67**	95.33	95.63	**84.23**	**89.05**	92.80	94.03
MV-HetGNN$_{\text{HAN}}$	59.32	59.57	14.04	14.01	93.34	93.80	79.91	85.16	80.06	83.60
MV-HetGNN$_{\text{MAGNN}}$	60.48	60.47	14.21	15.27	94.37	94.66	81.85	86.83	92.69	94.31
MV-HetGNN$_{\text{w/o TransE}}$	60.80	60.83	13.44	14.32	95.17	95.38	83.09	87.93	92.56	93.37
MV-HetGNN$_{\text{GAT}}$	**61.87**	**61.90**	15.89	16.93	**95.65**	**95.93**	83.83	88.68	**93.31**	**95.74**
MV-HetGNN$_{\text{concat}}$	54.47	55.14	9.72	10.01	94.54	94.96	80.08	85.94	81.40	85.27
MV-HetGNN$_{\text{mean}}$	60.43	60.35	15.18	16.10	94.27	94.70	81.77	86.95	82.23	85.95
MV-HetGNN$_{\text{attn}}$	60.11	60.28	12.45	14.66	94.42	94.83	81.54	86.41	92.59	93.28
MV-HetGNN$_{\text{w/o ae}}$	61.08	61.15	15.33	16.28	94.82	95.09	82.17	86.59	83.35	86.02
MV-HetGNN$_{\text{w/o reg}}$	60.47	60.43	11.16	13.24	93.58	94.26	80.22	85.83	79.84	83.10

and the attention mechanism adopts the implementation in MAGNN [10]. Second, we conduct ablation studies by two variants, MV-HetGNN$_{\text{w/o ae}}$ and MV-HetGNN$_{\text{w/o reg}}$, which remove the hierarchical autoencoder (replaced with a linear layer) and orthogonal regularization, respectively.

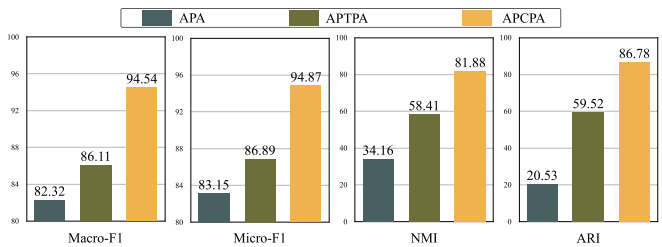
The results are shown in Table 6. MV-HetGNN consistently outperforms MV-HetGNN$_{\text{concat}}$, MV-HetGNN$_{\text{mean}}$, and MV-HetGNN$_{\text{attn}}$, indicating auto multi-view fusion layer's effectiveness. Compared with other approaches, MV-HetGNN$_{\text{concat}}$ performs worst. Although it preserves all information from multiple views intuitively, there is much redundant information as discussed in Section 4.3.2, which significantly hinders the performance in downstream tasks. Furthermore, the MV-HetGNN$_{\text{attn}}$ can not consistently outperform MV-HetGNN$_{\text{mean}}$, especially on the IMDb dataset. Similar results can also be found in a recent work GIAM [52], which further shows that the main reason for this phenomenon is overfitting. In addition, the results of MV-HetGNN$_{\text{w/o ae}}$ and MV-HetGNN$_{\text{w/o reg}}$ indicate that both hierarchical autoencoders and orthogonal regularization are critical. The orthogonal regularization is essential for the success of network optimization, while the auto encoders can further strengthen the performance.

5.5 Impact of the Length and Number of MetaPaths

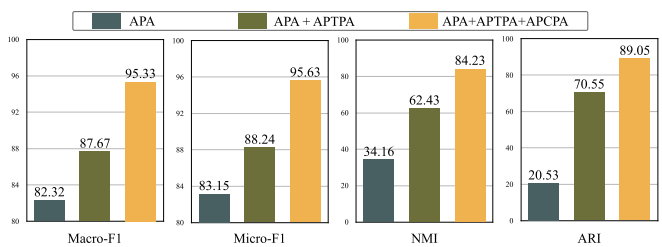
In this section, we experiment with the DBLP dataset to evaluate the impact of the length and number of metapaths.

First, we compare the performance on three metapaths (APA, APCPA, and APTPA) individually to evaluate the impact of the length of metapaths. Specifically, we remove the auto multi-view fusion layer and train MV-HetGNN solely on one of the metapaths. Second, we compare the performance on different numbers of metapaths to evaluate the impact of the number of metapaths. Specifically, we experiment on three groups of metapaths, APA, APA+APTPA, and APA+APTPA+APCPA. We report the average Macro-F1, average Micro-F1, NMI, and ARI.

From Fig. 4a, we have two observations. First, the performance on metapaths APTPA and APCPA outperforms APA, indicating that the performance of view-specific ego graph encoder does not get worse with the increase of metapath length. Second, although the lengths of metapath APCPA and APTPA are the same, their performance is quite different. Therefore, compared with the length of metapaths, the semantics behind them are more crucial factors affecting the performance in downstream tasks. In addition, some related works [53], [54] that can automatically learn metapaths also show that refined metapaths (e.g., 2∼4 order) are generally better.

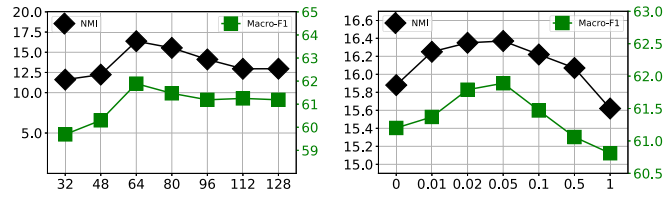


(a) Performance on metapaths of different lengths and semantics.



(b) Performance on different numbers of metapaths.

Fig. 4. Study on the impact of the length and number of metapaths.



(a) Dimension of the $\mathbf{H}$ (b) Value of λ in Eq. (19)

Fig. 5. Parameter sensitivity of MV-HetGNN.

Local structures under a too-short metapath cannot include enough structural and feature information, while local structures under a too-long metapath introduce too much noise, and they both cannot be used to produce better node embeddings.

As shown in Fig. 4b, benefiting from auto multi-view fusion layer, the performance of MV-HetGNN increases consistently with the number of metapaths and is at least as good as any single metapath. Moreover, we find that view-specific ego graph encoder based on a single metapath APCPA (Fig. 4a) outperforms the MV-HetGNN$_{attn}$ (Table 6), which integrates all three metapaths by the attention mechanism used in many works [9], [10]. This phenomenon again validates the superiority of our method.

5.6 Parameter Sensitivity

In this section, we conduct experiments to analyze the impacts of two critical hyper-parameters.

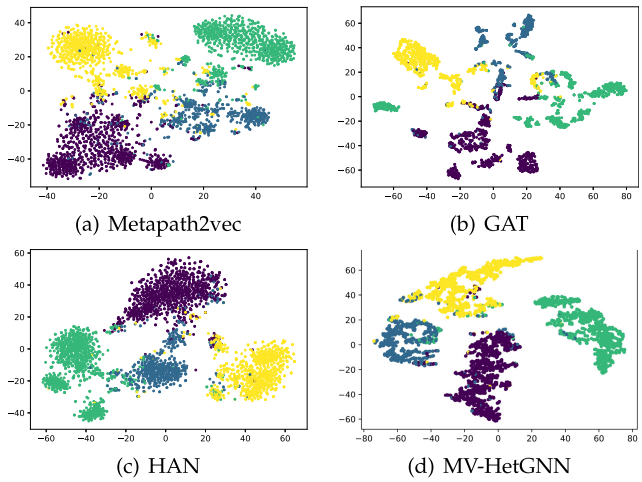
In Fig. 5, we present the results of NMI in node clustering and Macro-F1 in node classification on the IMDb dataset with different parameters.

First, we test sensitivity of the representation feature dimension d of $\mathbf{H}$. According to the setting in Section 5.1, we additionally set $d' = d^{\frac{M}{2}} = d$. Note that we only set it for the convenience of the parameter sensitivity test. The effect of dimension d of the final embedding $\mathbf{H}$ is shown in Fig. 5a. As the embedding dimension increases, the performance will gradually rise to the highest point and then drop slowly. This is because the smaller dimension is not enough to encode the heterogeneous feature information and semantic information, while the larger dimensions may introduce redundancy, hindering the model optimization.

Second, we test the sensitivity of λ in Eq. (19). Fig. 5b reports the effect of λ. As discussed in the Section 4.4, considering various data sets and downstream tasks, too strong versatility constraints (higher λ) can harm the model performance. A relatively small value will improve the performance of MV-HetGNN. For example, assuming that complementary metapaths that are meaningful to downstream tasks coexist with meaningless metapaths, it is beneficial to integrate all the information from the complementary metapaths, but it is harmful to forcibly contain information from the meaningless metapaths.

5.7 Visualization

In this part, we conduct the task of visualization to intuitively compare the embedding results on a low dimensional space. First, we get the embeddings of the nodes in the testing set and then project them into a 2-dimensional space through t-SNE [55]. We present a visualization of author node embeddings in DBLP and color the nodes based on their labels. As shown in Fig. 6, GAT performs worst, where nodes of the same type are not closely distributed, and nodes with different types are mixed. Benefiting from the use of multiple metapaths to explore comprehensive semantic information, the visualization of HAN performs better than Metapath2vec and GAT. Further, MV-HetGNN achieves the best visualization performance. The nodes of the same type are located close to each other, and the nodes from different types are well separated.



(a) Metapath2vec (b) GAT (c) HAN (d) MV-HetGNN

Fig. 6. Visualization of embeddings on DBLP. Each point denotes one author and its color indicates the research area.

6 Conclusion

In this paper, we introduce the idea of multi-view representation learning to HGs embedding and propose a *Heterogeneous Graph Neural Network with multi-view representation learning* (MV-HetGNN). The complex ego graph in HGs is decomposed into multiple view-specific ego graphs based on different semantics. Then, we employ the view-specific ego graph encoder to obtain node representation under each view. In this process, heterogeneity is addressed by learning the representation of relations and modeling the mapping relation between heterogeneous nodes. Then the auto multi-view fusion layer is developed to integrate the embeddings from diverse views. Versatile node embeddings with a theoretical guarantee are learned in this module. We conduct extensive experiments on three real-world datasets, and the results show that the proposed MV-HetGNN significantly outperforms all the baselines on various tasks.

Acknowledgments

The authors would also like to thank the anonymous reviewers for their comments on improving the quality of this paper.

References

[1] W. Fan et al., "Graph neural networks for social recommendation," in *Proc. Int. World Wide Web Conf.*, 2019, pp. 417–426.

[2] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5171–5181.

[3] 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. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 974–983.

[4] M. Wang, Y. Lin, G. Lin, K. Yang, and X.-M. Wu, "M2GRL: A multi-task multi-view graph representation learning framework for web-scale recommender systems," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 2349–2358.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.

[7] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2021, pp. 10 737–10 745.

[8] 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," 2020, *arXiv:2011.14867*.

[9] X. Wang et al., "Heterogeneous graph attention network," in *Proc. Int. World Wide Web Conf.*, 2019, pp. 2022–2032.

[10] X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding," in *Proc. Int. World Wide Web Conf.*, 2020, pp. 2331–2341.

[11] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.*, 2018, pp. 593–607.

[12] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 793–803.

[13] X. Qin, N. Sheikh, B. Reinwald, and L. Wu, "Relation-aware graph attention model with adaptive self-adversarial training," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2021, pp. 9368–9376.

[14] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proc. Int. World Wide Web Conf.*, 2020, pp. 2704–2710.

[15] Q. Zhong et al., "Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network," in *Proc. Web Conf.*, 2020, pp. 785–795.

[16] F. Xie, Z. Cao, Y. Xu, L. Chen, and Z. Zheng, "Graph neural network and multi-view learning based mobile application recommendation in heterogeneous graphs," in *Proc. IEEE Int. Conf. Serv. Comput.*, 2020, pp. 100–107.

[17] 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.

[18] F. Xu, J. Lian, Z. Han, Y. Li, Y. Xu, and X. Xie, "Relation-aware graph convolutional networks for agent-initiated social e-commerce recommendation," in *Proc. 28th Conf. Inf. Knowl. Manage.*, 2019, pp. 529–538.

[19] H. Ji et al., "Who you would like to share with? A study of share recommendation in social e-commerce," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2021, pp. 232–239.

[20] Z. Qiao, P. Wang, Y. Fu, Y. Du, P. Wang, and Y. Zhou, "Tree structure-aware graph representation learning via integrated hierarchical aggregation and relational metric learning," 2020, *arXiv:2008.10003*.

[21] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.

[22] C. Zhang, Y. Liu, and H. Fu, "AE2-Nets: Autoencoder in autoencoder networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2577–2585.

[23] C. Zhang, Y. Cui, Z. Han, J. T. Zhou, H. Fu, and Q. Hu, "Deep partial multi-view learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 5, pp. 2402–2415, May 2022.

[24] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 1–14.

[25] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[26] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[28] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[29] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "AM-GCN: Adaptive multi-channel graph convolutional networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1243–1253.

[30] P. Li, Y. Wang, H. Wang, and J. Leskovec, "Distance encoding: Design provably more powerful neural networks for graph representation learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 4465–4478.

[31] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.

[32] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1358–1368.

[33] C. Zhang, A. Swami, and N. V. Chawla, "SHNE: Representation learning for semantic-associated heterogeneous networks," in *Proc. Int. Conf. Web Search Data Mining*, 2019, pp. 690–698.

[34] C. Yue et al., "HTGN-BTW: Heterogeneous temporal graph network with bi-time-window training strategy for temporal link prediction," 2022, *arXiv:2202.12713*.

[35] Y. Fu, Y. Xiong, S. Y. Philip, T. Tao, and Y. Zhu, "Metapath enhanced graph attention encoder for hins representation learning," in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 1103–1110.

[36] X. Chu, X. Fan, D. Yao, C.-L. Zhang, J. Huang, and J. Bi, "Noise-aware network embedding for multiplex network," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[37] S. Zhu, C. Zhou, S. Pan, X. Zhu, and B. Wang, "Relation structure-aware heterogeneous graph neural network," in *Proc. IEEE Int. Conf. Data Mining*, 2019, pp. 1534–1539.

[38] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye, "An attention-based graph neural network for heterogeneous structural learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2020, pp. 4132–4139.

[39] S. Vashishth, S. Sanyal, V. Nitin, and P. P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–16.

[40] B. Yang, W.-T. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," 2014, *arXiv:1412.6575*.

[41] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[42] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2017, pp. 1506–1515.

[43] L. Le, A. Patterson, and M. White, "Supervised autoencoders: Improving generalization performance with unsupervised regularizers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 107–117.

[44] C. Ranjan, *Understanding Deep Learning Application in Rare Event Prediction*, Atlanta, GA, USA: Connaissance Publishing, 2020.

[45] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[46] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao, "Graph regularized transductive classification on heterogeneous information networks," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2010, pp. 570–586.

[47] I. Cantador, P. Brusilovsky, and T. Kuflik, "2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011)," in *Proc. ACM Recommender Syst. Conf.*, 2011.

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

[49] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learn. Representations*, 2013, pp. 1–12.

[50] 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. 2019.

[51] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
[52] D. Jin, Z. Yu, D. He, C. Yang, P. Yu, and J. Han, "GCN for HIN via implicit utilization of attention and meta-paths," *IEEE Trans. Knowl. Data Eng.*, early access, Nov. 25, 2021, doi: 10.1109/TKDE.2021.3130712.
[53] Y. Yang, Z. Guan, J. Li, W. Zhao, J. Cui, and Q. Wang, "Interpretable and efficient heterogeneous graph convolutional network," *IEEE Trans. Knowl. Data Eng.*, early access, Aug. 06, 2021, doi: 10.1109/TKDE.2021.3101356.
[54] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 11 983–11 993.
[55] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.

Zezhi Shao received the BE degree from Shandong University, Jinan, China, in 2019. He is currently working toward the PhD degree with the Institute of Computing Technology, Chinese Academy of Sciences, China. His research interests include graph and data mining.

Yongjun Xu received the BEng and PhD degrees in computer communication from the Xi'an Institute of Posts & Telecoms (China) in 2001 and Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China in 2006, respectively. He is a professor with the Institute of Computing Technology, Chinese Academy of Sciences (ICT-CAS) in Beijing, China. His current research interests include artificial intelligence systems, and Big Data processing.

Wei Wei received the PhD degree from the Huazhong University of Science and Technology, Wuhan, China, in 2012. He is currently an associate professor with the Department of Computer of Science and Technology, Huazhong University of Science and Technology. He was a research fellow with Nanyang Technological University, Singapore, and Singapore Management University, Singapore. His current research interests include information retrieval, natural language processing, social computing and recommendation, cross-modal/multimodal computing, deep learning, machine learning and artificial intelligence.

Fei Wang born in 1988, PhD, associate professor. He received the BS degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2011 and the PhD degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences in 2017. From 2017 to 2020, he was a research assistant with the Institute of Technology, Chinese Academy of Sciences. Since 2020, he has been working as an associate professor with the Institute of Computing Technology, Chinese Academy of Sciences. His main research interests include spatiotemporal data mining, Information fusion, graph neural networks.

Zhao Zhang received the BE degree in computer science and technology from the Beijing Institute of Technology (BIT) in 2015, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences in 2021. He is a research associate with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His current research interests include data mining and knowledge graphs.

Feida Zhu received the BS degree in computer science from Fudan University, Shanghai, China, and the PhD degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA. He is currently an assistant professor with the School of Information Systems, Singapore Management University, Singapore. His current research interests include large-scale graph pattern mining and social network analysis, with applications on Web, management information systems, business intelligence, and bioinformatics.