Hypergraph Transformer Neural Networks

MENGRAN LI, YONG ZHANG, XIAOYONG LI, YUCHEN ZHANG, and BAOCAI YIN,
Beijing Key Laboratory of Multimedia and Intelligent Software Technology, Faculty of Information Technology, Beijing Institute of Artificial Intelligence, Beijing University of Technology

Graph neural networks (GNNs) have been widely used for graph structure learning and achieved excellent performance in tasks such as node classification and link prediction. Real-world graph networks imply complex and various semantic information and are often referred to as heterogeneous information networks (HINs). Previous GNNs have laboriously modeled heterogeneous graph networks with pairwise relations, in which the semantic information representation for learning is incomplete and severely hinders node embedded learning. Therefore, the conventional graph structure cannot satisfy the demand for information discovery in HINs. In this article, we propose an end-to-end hypergraph transformer neural network (HGTN) that exploits the communication abilities between different types of nodes and hyperedges to learn higher-order relations and discover semantic information. Specifically, attention mechanisms weigh the importance of semantic information hidden in original HINs to generate useful meta-paths. Meanwhile, our method develops a multi-scale attention module to aggregate node embeddings in higher-order neighborhoods. We evaluate the proposed model with node classification tasks on six datasets: DBLP, ACM, IBDM, Reuters, STUD-BJUT, and Citeseer. Experiments on a large number of benchmarks show the advantages of HGTN.

CCS Concepts: • **Networks** → *Network architectures*; *Network dynamics*; *Network properties;*

Additional Key Words and Phrases: Hypergraph, transformer, meta-paths, attention, node classification

ACM Reference format:
Mengran Li, Yong Zhang, Xiaoyong Li, Yuchen Zhang, and Baocai Yin. 2023. Hypergraph Transformer Neural Networks. *ACM Trans. Knowl. Discov. Data.* 17, 5, Article 63 (April 2023), 22 pages.
https://doi.org/10.1145/3565028

The research project is partially supported by National Key R&D Program of China No. 2021ZD0111902, National Natural Science Foundation of China under Grant No. 62072015, U21B2038, U1811463, 61906011, Beijing Municipal Science and Technology Project No. KM202010005014.

Authors' address: M. Li, Y. Zhang (corresponding author), X. Li, Y. Zhang, and B. Yin, Beijing Key Laboratory of Multimedia and Intelligent Software Technology, Faculty of Information Technology, Beijing Institute of Artificial Intelligence, Beijing University of Technology, Chaoyang 100021, Beijing, China; emails: 1127147088@qq.com, zhangyong2010@bjut.edu.cn.

1556-4681/2023/04-ART63 $15.00
https://doi.org/10.1145/3565028

1 INTRODUCTION

In recent years, the application of deep neural networks to non-Euclidean structure data processing tasks has attracted extensive attention [14, 31]. The emergence of **graph neural networks (GNNs)** makes a breakthrough in the processing of unstructured data [27, 35]. In graph theory, one edge of a conventional graph can only represent the relationship between two nodes (one edge links two nodes). However, the data structure in the real-world may exceed pairwise connection, which leads to more complex data correlation [7]. In a citation network, papers are nodes. For a conventional graph network, if two papers belong to the same author, they can establish a connection relationship, which reflects the communication between the two nodes. But in fact, an author may have three or more papers. To represent this kind of high-order relationship among multiple papers belonging to the same author, it is necessary to introduce a more practical expression to establish communication between multiple papers. Furthermore, the graph networks in the real-world are usually heterogeneous and contain various semantic information, such as social networks [2], citation networks [15], and knowledge graphs [25, 33]. **Heterogeneous information networks (HINs)** contain different types of nodes or edges, which imply rich semantic information. Figure 1 shows a heterogeneous network with inconsistent edge types. The nodes are connected with the semantic relationship of different types of edges ("paper-author-paper", "paper-keyword-paper", and "paper-conference-paper").

For different types of nodes and semantic relations in HINs, multi-hop meta-paths of different lengths can be extracted. For example, the target node of the meta-path "paper-author-conference-author-paper" (see Figure 1) is paper and the semantic relation is that the authors of two papers jointly attend the same conference. Previous studies use a meta-path-based search method to compute similarity measures for heterogeneous graphs, searching for nodes with the same category [22]. Some scholars extend random walk [17] to HINs to obtain graph sequences, and then use skip-gram [16] to learn the embedding of graphs [3]. However, these methods rely too much on topological features of meta-path, ignoring node attribute embedded features, which result in suboptimal results.

As GNNs are widely used in HINs, a large amount of work such as HAN [26], HGT [9], GTN [32], and other **heterogeneous graph neural networks (HGNNs)** enable the combination of meta-path, which obtain the state-of-the-art results on many tasks. However, there are still two main challenges in learning and representing with HINs:

Challenge 1: Discover higher-order information implied by multivariate relationships. An edge of a simple graph can only connect two nodes, thus the high-order relations among nodes are ignored. In order to improve the representation of high-order information, it is necessary to strengthen the communication between nodes.

Challenge 2: Weigh rich semantic information between different types. In heterogeneous networks, there exist different types of nodes and edges, which contain much semantic information. To improve the multi-hop representation of node embedding, weighted semantic information needs to be incorporated into the feature space.

In fact, compared with the conventional graph structure, the hypergraph structure has a stronger ability to mine the nonlinear higher-order relationships among data samples. The hypergraph's Laplacian matrix extends the node neighborhoods and can aggregate richer higher-order information. Therefore, the hypergraph can model multivariate relationships more accurately and prevent the loss of original information caused by the process of forcing multivariate relationships into binary relationships. Hypergraphs are more flexible in handling multimodal and heterogeneous data, and more convenient for the fusion of heterogeneous data.

In this article, we propose a **hypergraph transformer neural network (HGTN)**, which could independently learn semantic information and represent high-order relations for the problems

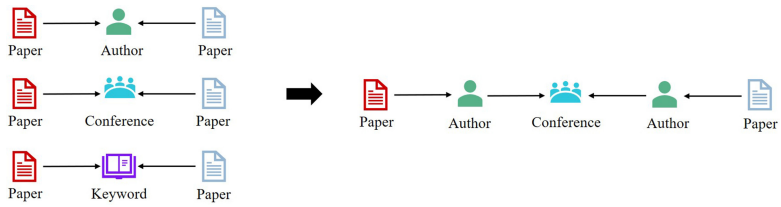


Fig. 1. Meta-paths of heterogeneous information in citation networks.

above. HGTN first constructs heterogeneous hypergraphs in HINs using raw meta-paths. Then, the raw meta-paths in the heterogeneous hypergraph are weighted and new meta-paths are generated by an attention aggregation module. Finally, a multi-scale attention module is used on the hypergraph convolution to learn higher-order node embeddings. The main contributions of this article are summarized as follows.

— For HINs, an end-to-end hypergraph transformer neural network, i.e., HGTN, is proposed to obtain excellent node embedding;
— Attention mechanism is introduced to weigh the weights of different types of hypergraphs and discover useful meta-paths for discovering implicit semantic information;
— Hypergraph structure is modeled for the HINs to enhance the high-order communication ability among nodes. The feasibility of convolution in heterogeneous hypergraphs is also theoretically proved;
— Node classification experiments are carried out on heterogeneous data such as citation, text, and social networks. Compared with the baselines, the model's performance is verified.

This article is organized as follows. Section 2 reviews the related works. Section 3 will introduce the formulation of the proposed HGTN. Section 4 assesses the performance of the proposed method on several datasets. Finally, conclusions are discussed in Section 5.

2 RELATED WORK

This section introduces the related work of hypergraph neural network and heterogeneous network representation learning.

2.1 Hypergraph Neural Network

In recent years, the extension of the neural network to graph structure has attracted extensive attention from researchers. Hypergraph structure can represent the high-order correlation between data, so it is applied to GNNs. In the field of graph node classification, [5] proposes a **hypergraph neural network** (**HGNN**) framework for data representation learning. The framework can encode high-level data correlation in hypergraph structure and realize complex data learning in citation networks and visual objects. Reference [12] improves HGNN and proposes a **dynamic hypergraph neural network** (**DHGNN**), which uses the DHG module to update the hypergraph structure of each layer dynamically. In order to effectively learn the deep embedding of high-order graph structure data, two end-to-end trainable operators are introduced, namely hypergraph convolution and hypergraph attention, to enhance the learning performance. Reference [21] splits the heterogeneous hypergraph into a series of snapshots and uses wavelet basis instead of Fourier basis to perform local hypergraph convolution, which reduces the computational cost and dramatically improves the training speed. HyperGCN [29] and other models use a GCN model to model complex relationships. Reference [34] proposes a hypergraph attention module, which further enhances the representation learning ability of hypergraphs.

Hypergraph has an excellent performance in dealing with high-order relations, and the research of HGNNs has also made phased progress in some fields. This article believes that finding the semantic information between different types of hypergraphs is very helpful in learning heterogeneous relationships.

2.2 Heterogeneous Network Representation Learning

Heterogeneous networks are represented as a collection of nodes, edges, and types, which widely exist in many scenes in the real world. Heterogeneous hypergraph representation learning is of great significance for the construction, reasoning, and application of hypergraphs. Reference [10] uses the biased second-order random walk framework on hypergraph for the first time, which achieves good performance in hypergraph representation learning, and optimizes the time cost. However, this method uses unsupervised generation of node embedding, which cannot map the node embedding to the hyperedge, resulting in the inability to capture the high-order structure information of the hyperedge. Therefore, [11, 30] introduce random walk to learn entity pair relationship and hyper relationship to capture the high-order structure information of hypergraph. PathSim [22] is the first to use meta-paths to search HINs. It is represented by learning nodes based on the similarity measurement of the meta-path. However, the model relies too much on the meta-path and its weight defined by experts. To this end, hin2vec [6] uses different types of relationships between nodes, and [3, 4] automatically update meta-paths based on random walk methods. However, these methods need to use a priori knowledge to preset meta-paths. In the representation and learning of HINs, some excellent models have been proposed. HAN [26] transforms a HIN into several homogeneous graphs based on user-specified symmetric meta-paths. Then, it applies GCN separately on each obtained homogeneous graph and aggregates the output representations by attention. HetSANN [8] and HGT [9] extend GAT [24] to HINs. They directly calculate attention scores for all the neighbors of a target object and perform aggregation accordingly. GTN [32] uses iterative matrix multiplication to learn the weighted meta-path adjacency matrix. Then, the graph convolution is carried out according to the obtained adjacency matrix.

In the field of heterogeneous networks, meta-paths are not sufficiently exploited for the discovery of implicit higher-order semantic information. This article believes that the introduction of a hypergraph-based attention mechanism [18, 19] can help to solve this problem.

3 HYPERGRAPH TRANSFORMER NEURAL NETWORKS

This section describes the HGTN in detail. Firstly, the mathematical definitions of heterogeneous hypergraphs and meta-paths are given. The weight of different types of hypergraph structures is learned through the **hypergraph aggregation** (**HA**) module. Useful meta-paths are generated to learn semantic information. Then, a multi-scale hypergraph attention neural network learns node embeddings to represent high-order semantic information. The framework of our method is shown in Figure 2.

3.1 Preliminaries

Definition 1 (Heterogeneous Hypergraph). A heterogeneous hypergraph is defined as a quadruple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W}, \mathcal{T})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the nodes set and $\mathcal{E} = \{e_1, e_2, \ldots, e_M\}$ is the hyperedges set constructed by nodes. Each hyperedge has multiple nodes (greater than or equal to 2). N and M, respectively, represent the maximum values of hypergraph nodes and edges. $\mathbf{W} = \text{diag}(\mathbf{w}_{e_1}, \mathbf{w}_{e_2}, \ldots, \mathbf{w}_{e_M})$ is a diagonal matrix representing the hyperedge weight. $\mathcal{T} = \{t_1, t_2, \ldots, t_T\}$ represents the typeset of heterogeneous hypergraphs, and T is the maximum of types. Each node v and each hyperedge e are associated with a mapping functions $\varphi(v) : \mathcal{V} \rightarrow \mathcal{T}_v$

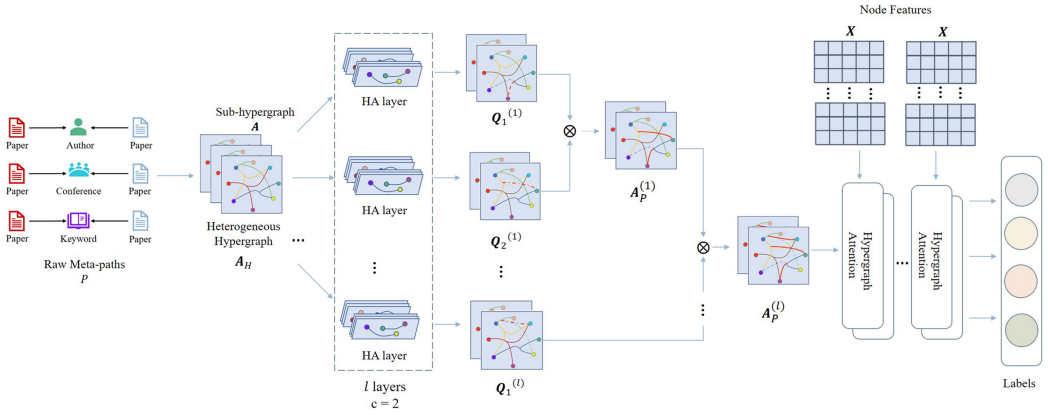


Fig. 2. The framework of HGTN. First, a heterogeneous hypergraph is constructed based on the meta-paths in the HIN (the adjacency matrix is denoted as $\mathbf{A}_H$), which includes T sub-hypergraphs (the adjacency matrix is denoted as A). Then, $\mathbf{A}_H$ is fed to HA module to learn to generate a new meta-path hypergraph (the adjacency matrix is denoted as $\mathbf{A}_P$) by attention map $\mathbf{Q}$. Finally, the learned $\mathbf{A}_P$, and the target node features are fed to the multi-scale hypergraph attention neural network to learn the node embedding to complete the node classification.

and $\varphi(e) : \mathcal{E} \rightarrow \mathcal{T}_e$, respectively. $\mathcal{T}_v$ and $\mathcal{T}_e$ denote the sets of object and relation types, where $|\mathcal{T}_v| + |\mathcal{T}_e| > 2$.

A hypergraph in the heterogeneous hypergraphs $\mathcal{G}$ can be represented by the incidence matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$. The elements in the matrix are defined as

$$\mathbf{H}(v, e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$\mathrm{D_v} \in \mathbb{R}^{N \times N}$ and $\mathrm{D_e} \in \mathbb{R}^{M \times M}$ are diagonal matrices, representing the degree matrix of nodes and edges. Specifically, the degree $\mathrm{D_v}(\mathrm{i}, \mathrm{i})$ of hypergraph node i is defined as

$$\mathbf{D}_v(i, i) = \sum_{e=1}^{M} \mathbf{W}_e \mathbf{H}(i, e) \tag{2}$$

the degree $\mathbf{D}_e(i, i)$ of hyperedge i is defined as

$$\mathbf{D}_e(i, i) = \sum_{v=1}^{N} \mathbf{H}(v, i), \tag{3}$$

the normalized hypergraph adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ represents the connection relationship between each node, which can be defined as

$$\mathbf{A} = {\mathbf{D}_v}^{-1/2} \mathbf{H} \mathbf{W} {\mathbf{D}_e}^{-1} \mathbf{H}^T {\mathbf{D}_v}^{-1/2}. \tag{4}$$

Definition 2 (Meta-path). The meta-path P represents the path through which any two nodes are connected, defined as $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} V_3 \cdots \xrightarrow{R_{l-1}} V_l$, which expresses a combination relationship $R = R_1 \circ R_2 \circ \cdots \circ R_{l-1}$ between node types $V_1 \cdots V_l$. $\circ$ represents the combination operation between relationships. Taking citation network as an example, the meta-path "APA" represents the co-authorship of two authors (A) on a paper (P), expressed as $\mathbf{A}_{APA} = \mathbf{A}_{AP} \times \mathbf{A}_{PA}$. "APCPA" denotes two authors (A) presenting a paper (P) at the same conference (C) and can be expressed

Table 1. Main Notations and Descriptions for the HGTN

Notations	Descriptions
$\mathbf{H} \in \mathbb{R}^{N\times M}$	The hypergraph incidence matrix
$\mathbf{D}_v \in \mathbb{R}^{N\times N}$	The node degree matrix of $\mathbf{H}$
$\mathbf{D}_e \in \mathbb{R}^{M\times M}$	The hyperedge degree matrix of $\mathbf{H}$
$\mathbf{W} \in \mathbb{R}^{M\times M}$	The weighted matrix of hyperedge
$\mathbf{A} \in \mathbb{R}^{N\times N}$	The sub-hypergraph adjacency matrix
$\mathbf{A}_H \in \mathbb{R}^{N\times N\times T}$	The heterogeous hypergraph adjacency matrix
$\mathbf{A}_P \in \mathbb{R}^{N\times N}$	The meta-path hypergraph adjacency matrix
$\mathbf{Q} \in \mathbb{R}^{N\times N}$	The attention map matrix
$\mathbf{X} \in \mathbb{R}^{N\times d}$	The input attribute matrix
$\mathbf{P} \in \mathbb{R}^{N\times d_i}$	The embedded feature in multi-scale hypergraph attention module
$\mathbf{Z} \in \mathbb{R}^{N\times d_i}$	The embedded feature in HGTN
$\Omega \in \mathbb{R}^{N\times d_i}$	The learnable matrix of hypergraph convolution
$\mathbf{W}^{\alpha} \in \mathbb{R}^{Vd_i\times V}$	The learnable matrix of MLP
l	The number of the meta-path
c	The number of channels

as $\mathbf{A}_{APCPA} = \mathbf{A}_{APC} \times \mathbf{A}_{CPA}$. By analogy, the semantic information between various types can be represented by meta-path of different lengths. According to the definition of graph theory, the result obtained by multiplying two adjacency matrices is the path with length 2 between the corresponding two nodes. Therefore, a meta-path with a length of l can be obtained by multiplying l adjacency matrices.

3.2 Heterogeneous Hypergraph Construction

For heterogeneous networks with prior knowledge, the raw element path is used as a mapping function $\varphi(\cdot)$ to construct sub-hypergraphs, and several different sub-hypergraphs are jointly represented as heterogeneous hypergraphs in this article.

For heterogeneous networks without prior knowledge, the hyperparameters k in the undirected **K-nearest neighbor** (**KNN**) hypergraph are used as mapping functions $\varphi(\cdot)$ to construct multiple sub-hypergraphs, which are finally merged. It is common to construct KNN hypergraphs based on feature distances to determine node similarity [1]. KNN hypergraph is constructed as follows.

Suppose the raw data attribute feature $\mathbf{X} \in \mathbb{R}^{N\times d}$, where each row $\mathbf{x}_i$ represents the ith node, and N is the number of samples and d is the dimension. The similarity matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ between node i and j is calculated by

$$\mathbf{S}_{ij} = \mathbf{x}_j^T \mathbf{x}_i. \tag{5}$$

After obtaining the similarity matrix $\mathbf{S}$, the first k nodes of each node are selected as neighbor nodes to form the hypergraph, and all nodes are traversed to construct the hypergraph. Concatenate all sub-hypergraphs with different k to obtain a heterogeneous hypergraph.

Finally, the hypergraph adjacency matrix is expressed as $\mathbf{A}_H \in \mathbb{R}^{N\times N\times T}$, which is an undirected symmetry matrix. The notations and descriptions are summarized in Table 1.

3.3 Hypergraph Transformer

Heterogeneous Hypergraph Aggregation Module. The introduction of hypergraphs gives nodes with more communication abilities. In order to learn the implicit semantic relationship in heterogeneous hypergraphs, a HA module is designed to learn soft selection composite relations for

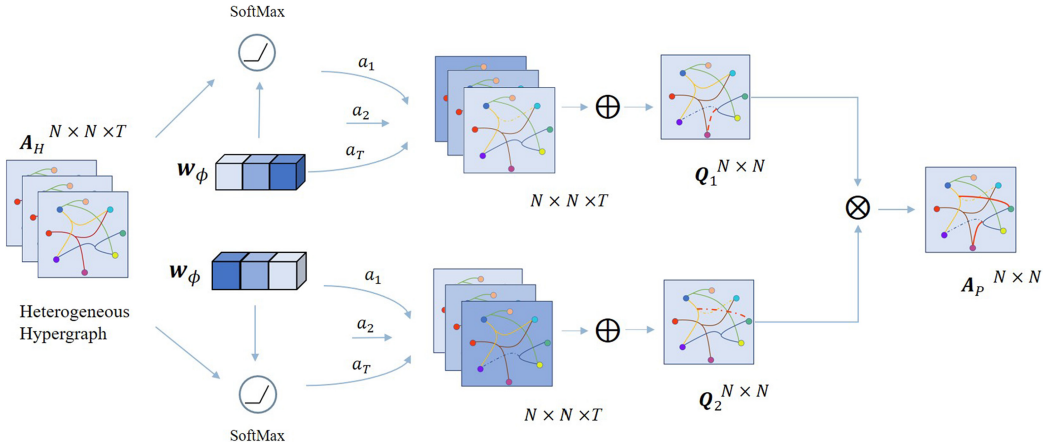


Fig. 3. The HA module first softly selects attention maps $\mathbf{Q}$ by convolving the candidate adjacency matrix $\mathbf{A}_H$ with the non-negative weights of Softmax. Then, it learns a new meta-path graph represented by $\mathbf{A}_P$ via the matrix multiplication of two selected adjacency matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ (so-called attention maps). The length of the new meta-path graph is 2, which means a second-order node relation. If l+1 attention maps are extracted, a meta-path graph of length l can be obtained.

generating useful multi-hop connections (so-called meta-paths). The details of the HA module are shown in Figure 3. The HA module softly selects an attention map $\mathbf{Q}$ by convolving the candidate adjacency matrix $\mathbf{A}_H$ with the non-negative weights of Softmax. The $\mathbf{Q}_{ij}$ is calculated as follows:

$$\mathbf{Q}_{ij} = \frac{\exp\left(-\sum_{t\in\mathcal{T}} \alpha_t \cdot \mathbf{A}_{H(ijt)}\right)}{\sum_{k\neq i} \exp\left(-\sum_{t\in\mathcal{T}} \alpha_t \cdot \mathbf{A}_{H(ikt)}\right)}, \tag{6}$$

where $\mathbf{W}_\varphi = [a_1, \ldots, a_T] \in \mathbb{R}^{1\times1\times T}$ is the learnable matrix, initialized to the standard normal distribution.

HGTN can learn arbitrary meta-paths in different edge types and path lengths. The meta-paths generation process in the HA module is essentially a stack structure. According to the meta-path definition, the attention map $\mathbf{Q}$ is combined in pairs (i.e., the matrix multiplication) to learn a new useful meta-path. The stack of l layers of the HA module allows learning l meta-path hypergraph structure, which indicates the number of generated meta-paths. The different meta-paths have different lengths, e.g., when $l = 2$, the length of the newly generated meta-path is 2. Equation (7) indicates that to generate l meta-paths $\mathbf{A}_P^{(l)}$, $l + 1$ attention maps need to be extracted to do matrix multiplication,

$$\begin{cases} \mathbf{A}_P^{(1)} = \mathbf{Q}_1^{(1)} \odot \mathbf{Q}_2^{(1)} \\ \mathbf{A}_P^{(2)} = \mathbf{A}_P^{(1)} \odot \mathbf{Q}_1^{(2)} \\ \quad \cdots \\ \mathbf{A}_P^{(l)} = \mathbf{A}_P^{(l-1)} \odot \mathbf{Q}_1^{(l)}, \end{cases} \tag{7}$$

where $\odot$ means Hadamard product.

However, there is a problem with this structure. Adding HA layers always increases the length of the meta-path, but this does not include the original edges. In some applications, both long meta-paths and short meta-paths are important. To learn meta-paths including primitive edges, we add the identity matrix $\mathbf{I}$ to $\mathbf{A}_H$, i.e., $\mathbf{A}_{H(::0)} = \mathbf{I}$. This trick allows HGTN to learn meta-paths of any length up to l+1 when stacking l layers of the HA module.

Multi-Scale Hypergraph Attention Module. Unlike conventional graphs, hyperedges in hypergraph structure can fuse the features among nodes. Thus, hypergraphs can learn higher-order relationships based on many-to-many node message passing. All nodes in the same hyperedge in the hypergraph can communicate freely, and the nodes in the same hyperedge are given the same contribution: $\frac{\mathbf{w}_{e_i}}{\mathbf{D}_e(i,i)}$, w_{e_i} is the weight of hyperedge i. At this time, the feature of node i is

$$\mathbf{E}_{v_i} = \sum_{v_i \in e_j} \mathbf{H}(v_i, e_j) \frac{\mathbf{w}_{e_j}}{\mathbf{D}_e(j,j)} \mathbf{X}_i, \tag{8}$$

if a node coexist in two or more hyperedges, the communication ability is the weighted contribution of multiple hyperedges: $\sum_{i=1}^{M} \mathbf{H}(v, e_i) \frac{\mathbf{w}_{e_i}}{\mathbf{D}_e(i,i)}$. The feature of hyperedge i is expressed as

$$\mathbf{E}_{e_i} = \sum_{v_j \in e_i} \mathbf{H}(v_j, e_i) \frac{\mathbf{w}_{e_i}}{\mathbf{D}_e(i,i)} \mathbf{X}_j. \tag{9}$$

Establish a matrix Θ to represent the probability matrix of random walk on hypergraph, and the value of each element in the matrix is expressed as

$$\Theta(v_i, v_j) = \sum_{n=1}^{M} \mathbf{H}(v_i, e_n) \frac{\mathbf{w}_{e_n}}{\mathbf{D}_v(i,i)} \frac{\mathbf{H}(v_j, e_n)}{\mathbf{D}_e(n,n)}, \tag{10}$$

after the hyperedge gathers the features of all sub-nodes, the information is transmitted back to all nodes through the weight of the hyperedge. The matrix normalized form of Θ can be expressed as

$$\Theta = \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e{}^{-1} \mathbf{H}^T \mathbf{D}_v^{-1/2}. \tag{11}$$

In the frequency domain, let the regularized hypergraph Laplacian be $\mathbf{L} = \mathbf{I} - \Theta$. According to the definition of hypergraph convolution, decomposing the hypergraph Laplacian matrix and approximating it by the first-order Chebyshev polynomial [13], the hypergraph convolution operation can be obtained by

$$\mathbf{g} * \mathbf{x} = \theta \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^T \mathbf{D}_v^{-1/2}, \tag{12}$$

θ is the approximate learning parameter in Chebyshev's inequality, the propagation mode of the HGNN can be expressed as

$$\mathbf{X}^{(s+1)} = \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^T \mathbf{D}_v^{-1/2} \mathbf{X}^{(s)} \Omega^{(s)}, \tag{13}$$

$\Omega \in \mathbb{R}^{C_1 \times C_2}$ are the parameters to be learned in training. C_1 and C_2 are the dimensions of the sth and $(s+1)$th hidden layer, respectively. Taking the meta-path adjacency matrix $\mathbf{A}_P$ learned in Equation (7) as the graph structure, the convolution operation and HGNN propagation methods are, respectively, expressed as

$$\mathbf{g} * \mathbf{x} = \theta \mathbf{A}_\mathbf{P} \mathbf{X}, \tag{14}$$

$$\mathbf{X}^{(s+1)} = \mathbf{A}_P \mathbf{X}^{(s)} \Omega^{(s)}, \tag{15}$$

where $\mathbf{A}_P \in \mathbb{R}^{N \times N}$, $\mathbf{X}^{(s)} \in \mathbb{R}^{N \times d_1}$, $\Omega^{(s)} \in \mathbb{R}^{d_1 \times d_2}$, $\mathbf{X}^{(s+1)} \in \mathbb{R}^{N \times d_2}$.

In the propagation learning of the HGNN, the transfer equation for learning the next layer using the feature $\mathbf{P}$ obtained from the s layer can be expressed as

$$\mathbf{P} = \mathbf{A}_P \mathbf{X} \Omega. \tag{16}$$

In order to learn more high-order semantic information, a multi-scale attention mechanism is introduced into the hypergraph for feature embedded representation, as shown in Figure 4. The embedded features $\mathbf{P}$ are first extracted through a hypergraph convolution of multiple shared parameters (see Equation (16)). All embedded features are concatenated as $[\mathbf{P}_1 || \mathbf{P}_2 || \cdots || \mathbf{P}_V] \in \mathbb{R}^{N \times V d_i}$,

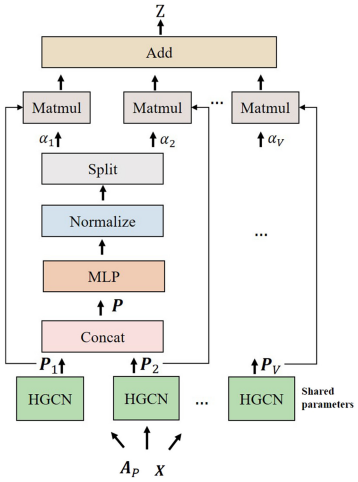


Fig. 4. The illustrations of the multi-scale hypergraph attention. Specifically, we first learn the weights through the proposed attention-based mechanism and then integrate the multi features $\mathbf{P}_i$ through the weighted fusion. Finally, the multi-scale weighted features are combined in a feature concatenation manner.

where d_i is the dimension of the ith hidden layer, and V is the number of hypergraph convolutions. Then, a **multilayer perceptron** (**MLP**) layer, which is parametrized by a weight matrix $\mathbf{W}^{\alpha} \in \mathbb{R}^{Vd_i \times V}$, is introduced to capture the attention for the concatenated features of the fused feature. Next, the LeakyReLU activation function is applied to the output of the normalized MLP layer. The output of LeakyReLU is normalized by the Softmax activation function and the ℓ_2 regularization function. Finally, the attention coefficients of each hypergraph convolution unit after feature sharing $[\alpha_1, \alpha_2, \ldots, \alpha_V]$ are denoted as Attention $(\alpha) \in \mathbb{R}^{N \times V}$,

$$\begin{aligned} \text{Attention}\,(\alpha) &= [\alpha_1, \alpha_2, \ldots, \alpha_V] \\ &= \ell_2(\text{softmax}(\text{LeakyReLU}([\mathbf{P}_1||\mathbf{P}_2||\cdots||\mathbf{P}_V]\,\mathbf{W}^{\alpha}))). \end{aligned} \tag{17}$$

Perform the Hadamard product ($\odot$) on the embedded features $\mathbf{P}_V$ to obtain the attention coefficients Attention (α) for each output of the hypergraph convolution unit, and sum to obtain the multiscale attention feature $\mathbf{Z}$,

$$\mathbf{Z} = \alpha_1 \odot \mathbf{P_1} + \alpha_2 \odot \mathbf{P_2} + \cdots + \alpha_V \odot \mathbf{P}_V. \tag{18}$$

In order to learn more high-order semantic information, we add multiple channels to the learning of node embedded feature $\mathbf{Z}$ to express more possibilities of attention weight. The final transfer equation of the multi-scale hypergraph attention neural network is expressed as

$$\mathbf{Z}^* = \sigma(||_{i=1}^{c}\mathbf{Z}), \tag{19}$$

where σ means activation function, c means the number of channels, and $||$ means concatenate operation. Equation (19) is a strategy similar to the multi-headed attention in GAT [24], and the same attention mechanism is used in this article. A richer node embedding is desired to be learned through multiple channels. $\mathbf{Z}$ contains node representations from c different channels of variable length up to l+1. The method in this article performs gradient descent using a cross-entropy loss function.

3.4 Hypergraph Transformer Neural Networks Analysis

This section mathematically proves the rationality of the application of hypergraph adjacency matrix $\mathbf{A}$ in HGNN. In Section 3.3, we learn the multi-hop adjacency matrix from Equation (7) as the graph structure. $\mathbf{A}_P$ is evolved from a heterogeneous hypergraph adjacency matrix, which can be regarded as a more effective hypergraph adjacency matrix.

Graph convolution is a particular case of hypergraph convolution. The representation of graph convolution is proved in literature [13] and in the frequency domain. Therefore, we compare it with graph convolution to illustrate the general representation of hypergraph convolution. The learning process of HGNN can be expressed as

$$\mathbf{X}^{(s+1)} = \sigma\left(1/2\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-1/2}\mathbf{X}^{(s)}\Omega^{(s)}\right). \tag{20}$$

According to the definition of graph, the above equation is converted into conventional graph structure, the representation of graph is actually to make the edge degree matrix of hypergraph $\mathbf{D}_e = 2\mathbf{I}$, weight matrix $\mathbf{W} = \mathbf{I}$, then Equation (20) is simplified as

$$\begin{aligned}\mathbf{X}^{(s+1)} &= \sigma\left(1/2\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{H}^T\mathbf{D}_v^{-1/2}\mathbf{X}^{(s)}\Omega^{(s)}\right)\\ &= \sigma\left(1/2\mathbf{D}_v^{-1/2}(\mathbf{A}+\mathbf{D}_v)\mathbf{D}_v^{-1/2}\mathbf{X}^{(s)}\Omega^{(s)}\right)\\ &= \sigma\left(1/2(\mathbf{I}+\mathbf{D}_v^{-1/2}\mathbf{A}\mathbf{D}_v^{-1/2})\mathbf{X}^{(s)}\Omega^{(s)}\right),\end{aligned} \tag{21}$$

if the constant 1/2 is not considered. Equation (21) is completely equivalent to the definition of unnormalized graph convolution network [13].

4 EXPERIMENT RESULTS AND ANALYSIS

We evaluate the performance of HGTN in node classification tasks on six datasets. Several baselines are selected for comparison. The performance of these methods is compared from many aspects. The GPU used in the experiment is NVIDIA GeForce RTX 3090 with 24G memory. Our source codes can be obtained on GitHub.[1]

4.1 Datasets

In order to comprehensively evaluate the performance of HGTN on heterogeneous networks, we select six datasets from the citation, text, and social networks. The statistical data used in our experiment are shown in Table 2.

- DBLP[2]: The paper citation network extracted from DBLP website contains three node graphs: co-authors (two authors jointly publish the same paper), common conference (two authors jointly participate in the same conference), and common theme (two authors use the same theme).
- ACM[3]: Papers in ACM library reference network. There are four node graphs: co-authors, common laboratory, common conference, and common theme.
- IMDB[4]: The data are from IMDB film introduction and scoring website. There are three node graphs: co-actors (two films contain the same actor), co-directors (two films contain the same director), and common year (two films are released in the same year).

[1] https://github.com/limengran98/HGTN.
[2] https://dblp.uni-trier.de.
[3] https://dl.acm.org.
[4] https://www.imdb.com.

Table 2. Dataset Statistics

Dataset	Nodes	Edge types	Features	Classes
DBLP	4057	A-P-A: 3528 A-P-C-P-A: 2498219 A-P-T-P-A: 3386139	334	4
ACM	3025	P-A-P: 13128 P-L-P: 1103868 P-C-P: 1109337 P-T-P: 9147568	1870	3
IMDB	4780	M-A-M: 46615 M-D-M: 8119 M-Y-M: 809072	1232	3
Reuters	10000	k = 5: 50000 k = 10: 100000	2000	4
STUD-BJUT	4060	information: 8120 library: 8120 gateway: 8120 shopping: 8120	105	3
Citeseer	3327	P-A-P: 9104	3703	6

— Reuters [20]: The data comes from many short news and related topics released by Reuters in 1986. It is a simple and widely used text classification dataset. Two node relation k-uniform hypergraphs are constructed according to the features.
— STUD-BJUT: The data is collected from the student activity information at Beijing University of Technology. The feature extraction is performed to obtain the description of the student's behavior features. According to different behaviors, it can be divided into four types: information, library, gateway, and shopping. We use this dataset to construct student association networks: students are nodes, and k-uniform hypergraphs ($k = 2$) are constructed according to the similarity of features. Use the student's grade points (good, medium, poor) as the node classification label.
— Citeseer[5]: The paper citation network describes the citation between the papers of the world's top conferences.

4.2 Baselines

In the experimental comparison, this article selects nine baselines, which belong to four benchmarks: random walk, GNN, HGNN, and heteorgeneous information neural network.

— Deepwalk [17]: A graph structure data discovering algorithm combining random walk and word2vec. The algorithm can learn the hidden information of the network and represent the nodes in the graph as a vector containing potential information.
— LINE [23]: A graph embedding method based on breadth first random walk to obtain context information and learn the network representation through first- and second-order similarity.
— GCN [13]: GNN with convolution operation can be applied to graph embedding representation learning.

[5]https://citeseerx.ist.psu.edu/index.

— GAT [24]: GNN aggregates neighbor nodes through self-attention mechanism to realize adaptive matching of weights of different neighbors.
— HGNN [5]: Hypergraph neural network with convolution operation can represent and learn high-order embedded features.
— HWNN [21]: The Fourier transform in hypergraph convolution is replaced by wavelet transform to represent heterogeneous hypergraphs.
— HAN [26]: A heterogeneous graph neural network based on hierarchical attention. Heterogeneous graphs are processed by extracting node level attention and semantic level attention.
— HGT [9]: The relevant parameters of node and edge types are designed to represent the neural network model of heterogeneous structure.
— GTN [32]: Graph transformation network, which updates the graph by generating meta-paths, uses convolution to represent the node features of the graph structure.

4.3 Experimental Setup and Metrics

The experimental setup is as follows: Each dataset is divided into a training set and a test set, which account for 80% and 20%. We set the learning rate to 5e-3, the regularization parameter to 5e-4, the dropout rate to 0.6, and the maximum epoch to 500. For our proposed HGTN, unless otherwise specified, the number of channels c to 2 and the number of meta-paths l to 3. For DeepWalk [17], we set window size to 5, walk length to 100, and walks per node to 40. For LINE [23], we set the number of negative samples used in negative sampling as 5. For models designed for isomorphic graphs, such as GCN [13], GAT [24], and HGNN [5], we first slice the heterogeneous graph into multiple homogeneous graphs containing heterogeneous information according to the meta-path and then superimpose these homogeneous graphs. For the HAN [26], we set the dimension of the semantic-level attention vector to 128, and the number of attention head k to 8. For the GTN [32], we set the meta-path length l to 2, and the number of channels c to 2. For HGT [9], we set the heterogeneous graph as the original input and the head number as 8. The node embeddings obtained from the model training are fed to the classification layer to predict the classes. HGTN introduces cross-entropy loss and uses Adam optimizer to update the learnable parameters. To ensure the fairness of the experiments, we perform multiple pieces of training and choose the mean of the best results obtained in each piece.

HGTN is used to classify different network nodes. Therefore, four evaluation metrics of macro-precision (Pre), macro-recall (Rec), accuracy (ACC), and macro-F1 (F1) score[6] are selected for comparison experiments. ACC can directly calculate the probability that the predicted value is correctly classified with the true value without being affected by the number of classes. For Pre, Rec, and F1, we denote the true positive, false positive, true negative, and false negative as TP, FP, TN, and FN, respectively [28]. The definitions of these metrics are

$$\begin{aligned} Pre &= \frac{TP}{TP + FP}, \\ Rec &= \frac{TP}{TP + FN}, \\ F1 &= \frac{2 \times Pre \times Rec}{Pre + Rec}, \end{aligned} \tag{22}$$

where Pre is intuitively the ability of the classifier not to label negative samples as positive. Rec is the ability of the classifier to find all the positive samples. F1 can be interpreted as a harmonic mean of the Pre and Rec. The relative contributions of Pre and Rec to F1 are equal. For multilabel

[6] https://scikit-learn.org/stable/modules/model_evaluation.html.

Table 3. Node Classification Comparison Experiment

Dataset	Metrics	Random Walk		Graph		Hypergraph		Information			ours
		DeepWalk	LINE	GCN	GAT	HGNN	HWNN	HAN	HGT	GTN	HGTN
DBLP	Pre	0.825	0.873	0.914	0.905	0.917	0.936	0.939	0.934	0.909	**0.940**
	Rec	0.839	0.876	0.921	0.910	0.904	0.900	0.934	0.932	0.914	**0.939**
	ACC	0.838	0.872	0.926	0.908	0.916	0.934	0.940	0.935	0.916	**0.945**
	F1	0.834	0.872	0.919	0.908	0.909	0.915	0.939	0.934	0.911	**0.939**
ACM	Pre	0.669	0.516	0.901	0.903	0.927	0.953	0.945	0.944	0.945	**0.975**
	Rec	0.685	0.482	0.911	0.898	0.927	0.953	0.949	0.934	0.945	**0.976**
	ACC	0.671	0.504	0.905	0.904	0.927	0.951	0.945	0.943	0.944	**0.975**
	F1	0.669	0.499	0.905	0.904	0.927	0.953	0.947	0.949	0.945	**0.976**
IMDB	Pre	0.521	0.495	0.534	0.544	0.564	0.546	0.542	0.542	0.544	**0.613**
	Rec	0.483	0.351	0.503	0.529	0.552	0.558	0.553	0.553	0.555	**0.582**
	ACC	0.481	0.471	0.522	0.564	0.627	0.625	0.614	0.619	0.622	**0.663**
	F1	0.507	0.307	0.491	0.519	0.557	0.554	0.544	0.543	0.548	**0.589**
Reuters	Pre	0.719	0.793	0.884	0.901	0.931	0.943	0.940	–	0.948	**0.952**
	Rec	0.606	0.789	0.874	0.892	0.919	0.955	0.943	–	0.941	**0.962**
	ACC	0.657	0.777	0.873	0.916	0.942	0.952	0.954	–	0.956	**0.965**
	F1	0.614	0.788	0.868	0.928	0.925	0.943	0.941	–	0.944	**0.957**
STUD-BJUT	Pre	0.529	0.676	0.530	0.711	0.707	0.735	0.739	–	0.731	**0.800**
	Rec	0.531	0.414	0.531	0.622	0.708	0.735	0.736	–	0.722	**0.790**
	ACC	0.584	0.622	0.585	0.649	0.708	0.739	0.739	–	0.723	**0.792**
	F1	0.530	0.406	0.531	0.622	0.707	0.726	0.732	–	0.724	**0.791**
Citeseer	Pre	0.547	0.588	0.731	0.735	0.711	0.755	0.753	–	0.750	**0.764**
	Rec	0.563	0.600	0.656	0.652	0.708	0.731	0.735	–	**0.743**	0.740
	ACC	0.560	0.592	0.710	0.713	0.739	0.751	0.764	–	0.764	**0.776**
	F1	0.550	0.583	0.714	0.711	0.710	0.742	0.732	–	0.739	**0.743**

classification, the metrics for each class are summed and averaged to obtain the final evaluation results.

4.4 Node Classification Results and Analysis

The node classification results of HGTN and the baseline method on the six datasets are shown in Table 3. We divide the baseline methods into four categories (graph embedding, GNN, HGNN, and heterogeneous graph network) to verify the improvement effect of our method. At the same time, six datasets can also be subdivided into three categories: Heterogeneous semantic graph structure: ACM, DBLP, IMDB; K-uniform graph structure: Reuters, STUD-BJUT; Homogeneous graph structure: Citeseer. The results show that in most cases, the results of HGTN are better than the baseline. HGTN performs best on these datasets with an average improvement rate of about 3% due to the multiple semantic graph structures in ACM, DBLP, and IMDB with different meta-paths. For Reuters and STUD-BJUT, the k-uniform hypergraph constructed using node features, the improvement is about 1%–2%, which is not as good as the former. The improvement is minimal on the Citeseer dataset since the input graph is only an isomorphic graph. The above results show that HGTN can mine the high-order semantic information of nodes, which is more suitable for processing heterogeneous graph structures with semantic information.

We give a breakdown of the performance of each class in the test set for all datasets, see Figure 5. The bar chart in Figure 5 shows the performance of each class and the line chart shows the number of each class. In fact, the performance metrics in Table 3 are summed averages of the performance metrics for each class. Also, as can be seen in Figure 5, the number of all classes is roughly evenly

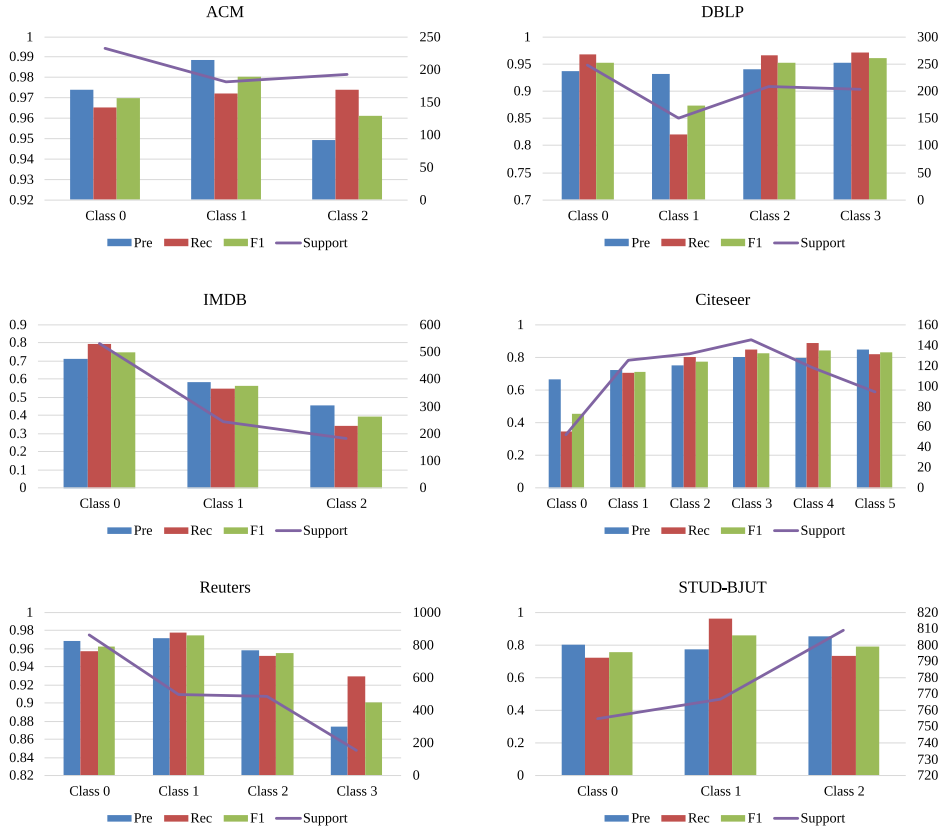


Fig. 5. The performance breakdown of all datasets for each class in the test set.

distributed, so that the evaluation metrics on the vast majority of the dataset are not affected by a particular class.

We visualize the training and testing iteration curves for all datasets, as shown in Figure 6. Our method achieves better test results within 100 epochs on ACM, DBLP, Reuters, and Citeseer datasets. However, the test metrics for the IMDB and STUD-BJUT datasets converge slowly.

To report the performance of HGTN in more detail, we introduce ROC-AUC curves. The **ROC** (**Receiver Operating Characteristic**) curve corresponds to the vertical coordinate of **True Positive Rate** (**TPR**) and the horizontal coordinate of **False Positive Rate** (**FPR**). TPR is equivalent to the recall and calculates the percentage of positive instances correctly classified by the classifier as a percentage of all positive instances. FPR is equal to 1- TPR, which calculates the proportion of negative instances that are incorrectly considered as positive by the classifier to all negative instances. **AUC** (**Area Under Curve**) is the size of the area under the ROC curve, and the larger the AUC, the better the performance of the model. Compared with other models (GCN, HGNN, GTN), the HGTN has the best performance in the ROC-AUC curves (as shown in Figure 7).

To verify the effect of each meta-path on the node classification results, we differentially input different meta-paths from three datasets (ACM, DBLP, and IMDB) into HGTN, and the results are shown in Tables 4 and 5. Table 4 shows the results for a single meta-path as input, and Table 5 shows the results from multiple meta-paths as input. Compared with all meta-paths as input, the

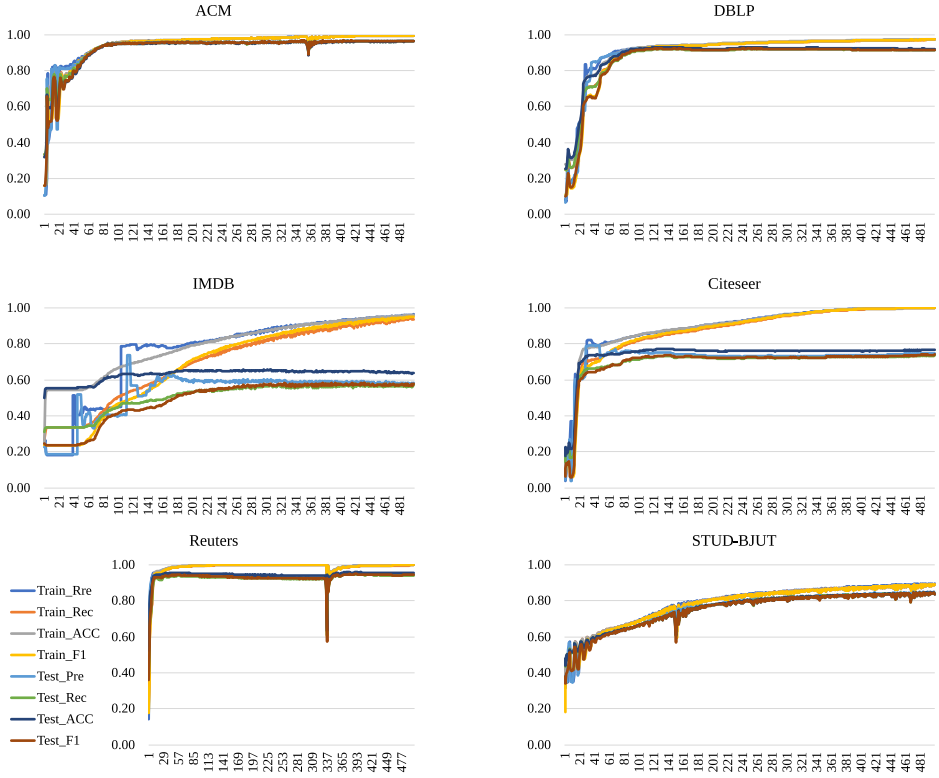


Fig. 6. Training and testing iteration curves for all datasets.

Table 4. Unit Meta-Path Influence

	Meta-path	F1	ACC	Pre	Rec
DBLP	A-P-A	0.819	0.825	0.821	0.818
	A-P-C-P-A	0.117	0.305	0.070	0.250
	A-P-T-P-A	0.764	0.778	0.766	0.762
ACM	P-A-P	0.927	0.927	0.927	0.927
	P-L-P	0.920	0.920	0.921	0.929
	P-C-P	0.934	0.934	0.934	0.933
	P-T-P	0.185	0.383	0.128	0.333
IMDB	M-A-M	0.272	0.548	0.396	0.431
	M-D-M	0.564	0.639	0.575	0.558
	M-Y-M	0.255	0.546	0.280	0.335

four evaluation metrics of Pre, Rec, ACC, and F1 for a single meta-path are decreased. Especially the poor results of "P-T-P" for ACM and "A-P-C-P-A" for DBLP indicate that this meta-path cannot support the model to train excellent node embeddings. In addition, the results of "M-A-M" and "M-Y-M" of IMDB are considerably lower. Among the multiple meta-paths, the combination of "P-L-P + P-C-P" has better results. The combination of "A-P-A" + "A-P-C-P-A" in the DBLP dataset has better results.

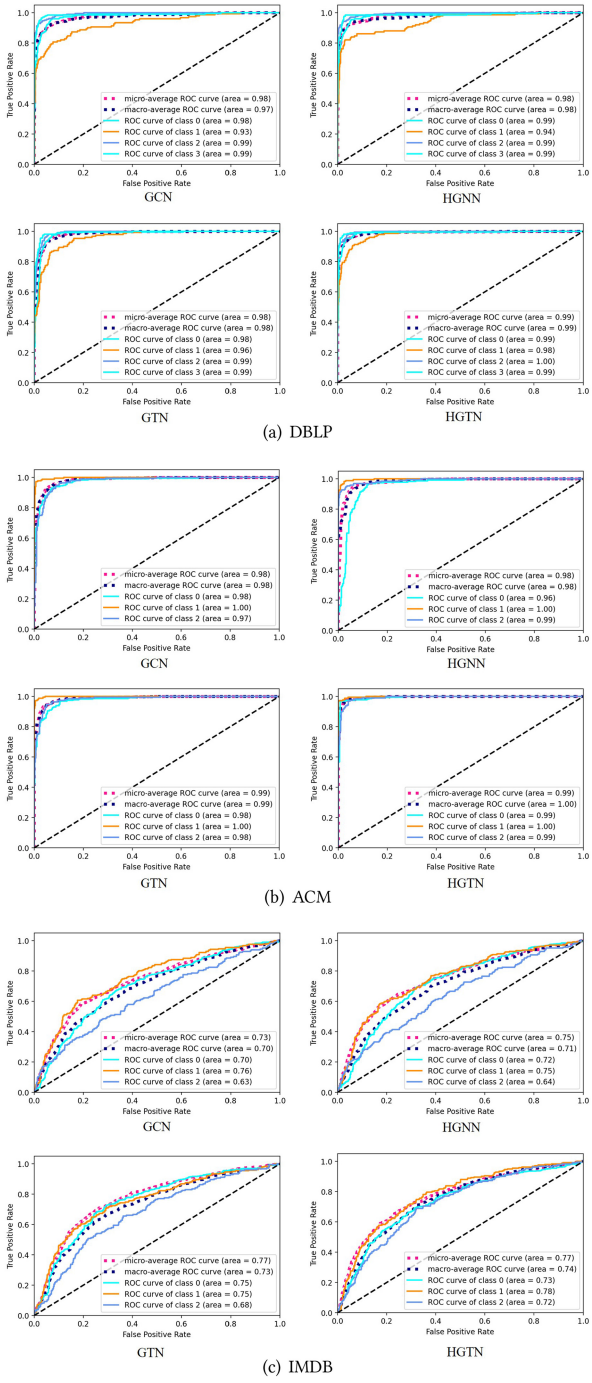


Fig. 7. The ROC and AUC of the DBLP, ACM, and IMDB dataset.

Table 5. Multiple Meta-Paths Influence

	Meta-paths	F1	ACC	Pre	Rec
DBLP	A-P-A + A-P-C-P-A	0.916	0.921	0.915	0.916
	A-P-A + A-P-C-P-A	0.824	0.830	0.829	0.821
	A-P-C-P-A + A-P-T-P-A	0.116	0.305	0.076	0.250
ACM	P-A-P + P-L-P	0.973	0.973	0.973	0.974
	P-A-P + P-C-P	0.977	0.976	0.978	0.976
	P-L-P + P-C-P	0.988	0.988	0.988	0.988
	P-A-P + P-L-P+P-C-P	0.985	0.985	0.985	0.985
IMDB	M-A-M + M-D-M	0.253	0.550	0.316	0.335
	M-A-M + M-Y-M	0.238	0.555	0.185	0.333
	M-D-M + M-Y-M	0.238	0.555	0.185	0.333

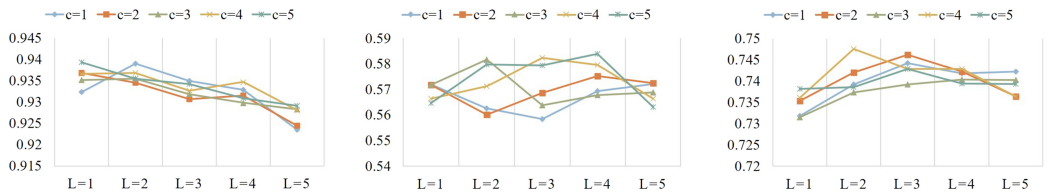


Fig. 8. F1 score of different hyperparameters on the node classification task.

4.5 Model Analysis

Hyperparameters Verification. HGTN has two hyperparameters: the number of the meta-path l and the number of channels c. In order to verify the influence of the two hyperparameters, we performed cross-validation. Choose the number of the meta-path $l = [1, 2, 3, 4, 5]$, the number of channels $c = [1, 2, 3, 4, 5]$. We performed node classification tasks on DBLP, Citeseer and IMDB, and used the F1 score as the evaluation index. The result is shown in Figure 8. On the DBLP dataset, as l increases, the overall F1 score shows a downward trend, and the effect is best when $c = 1$ and $l = 2$. The Citeseer dataset is opposite to DBLP. As l increases, the F1 score shows an upward trend, and the F1 score is the highest when $c = 2$ and $l = 2$. The F1 score is the highest when $c = 5$ and $l = 4$ in the IMDB dataset.

Ablation Experiments. In order to further analyze the effects of each part of the model, we conducted ablation experiments on all six datasets. Set up three sets of ablation models to compare with HGTN, and the results are shown in Figure 9. The three ablation implementations, respectively, verify the functions of hypergraph, HA layers, and attention.

— Hypergraph can express high-order relationships. If it is transformed into a graph, the four evaluation indicators will drop significantly in the node classification task, especially the recall rate.
— The role of the HA module is to discover the attention coefficients of different types of hypergraphs and generate multi-hop meta-paths. If this module is removed, the evaluation metrics will decrease.
— The role of attention is to discover multi-scale semantic information. Compared with the previous two, this part still plays a positive role in node classification. In general, for datasets with different semantic information such as DBLP, ACM hypergraph, and HA layers, the

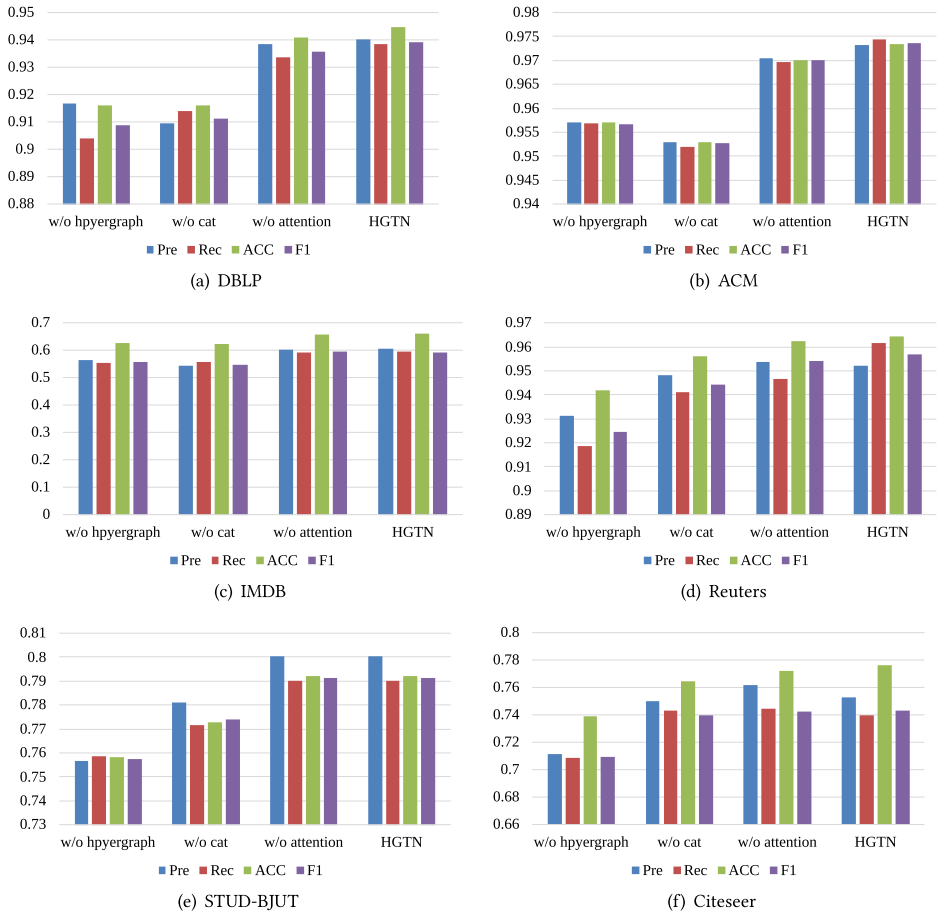
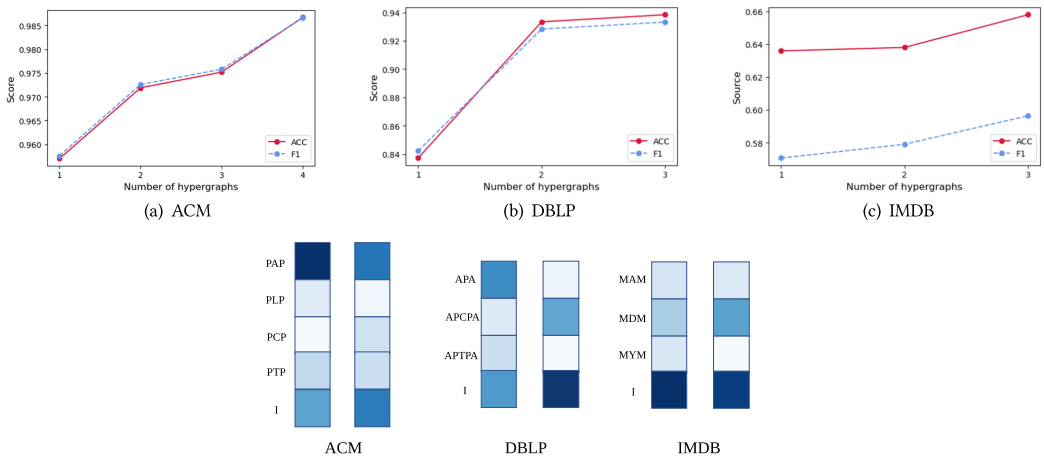


Fig. 9. F1 score of ablation model on node classification task.

impact is more significant. For k-uniform graph datasets Reuters, STUD-BJUT and single graph Citeseer, the improvement of a hypergraph is greater than that of HA layers.

Hypergraph Semantic Discovery. In order to analyze the improvement of semantic information discovered in the experimental results, we introduce different types of hypergraph meta-paths on the three datasets of DBLP, ACM, and IMDB. The results are shown in Figure 10(a)–(c). With the introduction of hypergraph meta-paths containing semantic relations, the F1 score of the model has also been improved. Experiments prove that the new meta-path can discover useful semantic information and improve accuracy. We visualize the attention coefficients of all hypergraph meta-paths, as shown in Figure 10(d). Compared with the DBLP results, the identity matrix has a higher attention coefficient in IMDB, which suggests that HGTN persists in learning shorter meta-paths at a deeper level. Also, the attention coefficient is higher for less semantic information (implicit data information needs strong attention), which suggests that HGTN is able to learn the most efficient meta-path adaptively based on the dataset.

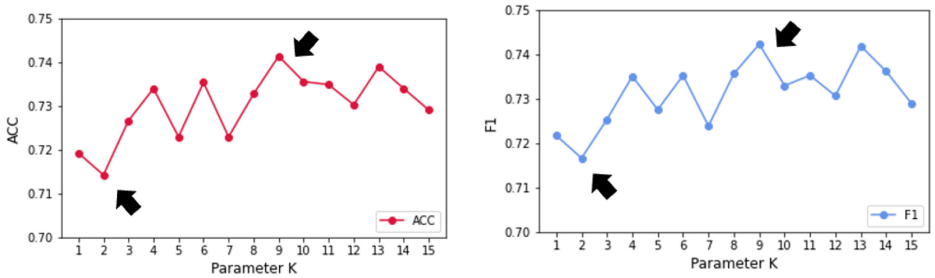
Robustness Verification. We use STUD-BJUT after noise attack to verify the robustness of HGTN. According to the study and rest time of the school and the function of each location, the dataset has



(a) ACM (b) DBLP (c) IMDB

(d) Visualization of the attention coefficients of the attention map for three heterogeneous datasets (number of channels c=2).

Fig. 10. F1 score of adding multi semantic hypergraph meta-paths and the attention coefficients of the attention map.

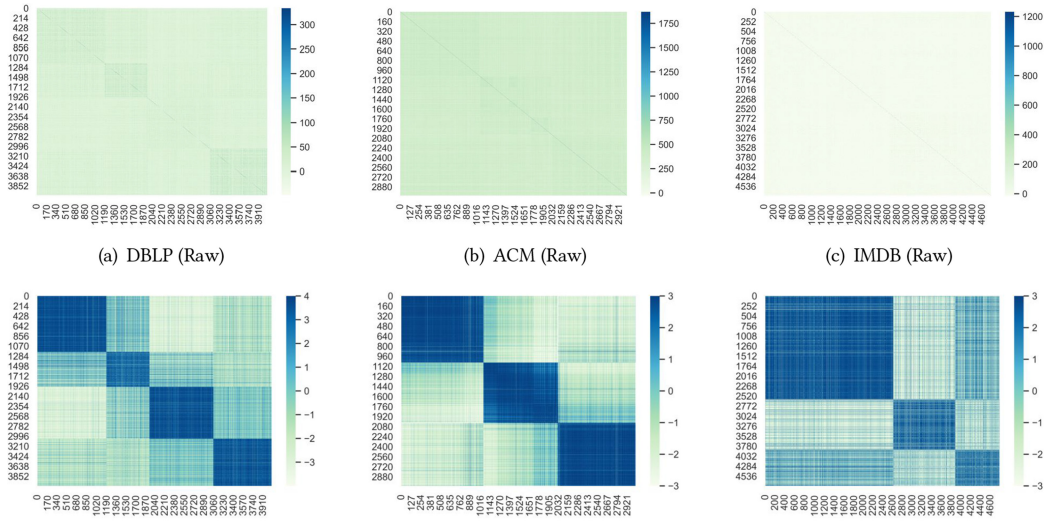


(a) ACC (b) F1

Fig. 11. Influence of neighbor number k in STUD-BJUT on model robustness.

four different types of student behavior features. Hypergraphs are constructed based on feature similarity. Due to the contingency and aggregation of students' behavior in the campus scene, it may be inaccurate to measure social relations through the feature similarity between nodes. Another problem is that it is impossible to determine how many neighbors the central node should connect because the number of nearest neighbors may affect the performance of hypergraph learning. Therefore, we assume that the change in the number of neighbors of the central node k is interference. In the experiment, we analyze the structure of different k values in the evaluation index. Due to the number of neighbors, the graph structure will introduce much noise that does not match the downstream tasks. Figure 11 shows that the variation range of ACC and F1 is minimal under noise interference, and the difference between the best and worst is about 3%, which proves that HGTN has good robustness.

Embedded Feature Similarity Comparison. It is generally believed that nodes with similar features are more closely connected [1]. We first obtain the original sample features and the embedded features learned by the proposed method. Then, we construct the element similarity matrix by calculating the cosine similarity on the three datasets of DBLP, ACM, and IMDB (see Figure 12,



(a) DBLP (Raw) (b) ACM (Raw) (c) IMDB (Raw)

(d) DBLP (HGTN) (e) ACM (HGTN) (f) IMDB (HGTN)

Fig. 12. The heat maps of node similarity matrices in the latent space of raw data and HGTN.

the coordinates are expressed as nodes). The results show that the feature similarity between the original data is insignificant. HGTN can learn a better representation of the embedding space. The feature similarity between different categories is distinguished sharply. This is attributed to the fact that HGTN learns more higher-order representations of HINs.

5 CONCLUSION

In this article, we propose an end-to-end method named HGTN for HIN learning. The model learns the high-order relationship and semantic information of heterogeneous networks to describe node embedding and improve the classification effect, with more extensive potential application in social networks and personalized recommendations. Experiments show that HGTN is superior to the baselines in evaluation metrics such as accuracy and recall. We verify the effectiveness of HGTN in high-order representation and semantic information discovery. The following work will further break through the integration of semantic knowledge networks and solve the limitations of semantic knowledge representation.

REFERENCES

[1] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural deep clustering network. In *Proceedings of the International Conference on World Wide Web*. 1400–1410.

[2] Qi Cao, Huawei Shen, Jinhua Gao, Bingzheng Wei, and Xueqi Cheng. 2020. Popularity prediction on social platforms with coupled graph neural networks. In *Proceedings of the International Conference on Web Search and Data Mining*. 70–78.

[3] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 135–144.

[4] Yongping Du, Wenyang Guo, Jingxuan Liu, and Changqing Yao. 2019. Classification by multi-semantic meta path and active weight learning in heterogeneous information networks. *Expert Systems with Applications* 123, C (2019), 227–236.

[5] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3558–3565.

[6] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the ACM on Conference on Information and Knowledge Management*. 1797–1806.
[7] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. 2020. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 5 (2020), 2548–2566.
[8] Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. 2020. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the Association for the Advance of Artificial Intelligence*, Vol. 34. 4132–4139.
[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the International Conference on World Wide Web*. 2704–2710.
[10] Jie Huang, Chuan Chen, Fanghua Ye, Jiajing Wu, Zibin Zheng, and Guohui Ling. 2019. Hyper2vec: Biased random walk for hyper-network embedding. In *Proceedings of the International Conference on Database Systems for Advanced Applications*. 273–277.
[11] Jie Huang, Xin Liu, and Yangqiu Song. 2019. Hyper-path-based representation learning for hyper-networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 449–458.
[12] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. 2019. Dynamic hypergraph neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 2635–2641.
[13] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907. Retrieved from https://arxiv.org/abs/1609.02907.
[14] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. 2019. Spam review detection with graph convolutional networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 2703–2711.
[15] Renjie Liao, Marc Brockschmidt, Daniel Tarlow, Alexander L. Gaunt, Raquel Urtasun, and Richard S. Zemel. 2018. Graph partition neural networks for semi-supervised classification. (2018).
[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781. Retrieved from https://arxiv.org/abs/1301.3781.
[17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 701–710.
[18] Xiangbo Shu, Jiawen Yang, Rui Yan, and Yan Song. 2022. Expansion-squeeze-excitation fusion network for elderly activity recognition. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 8 (2022), 5281–5292.
[19] Xiangbo Shu, Liyan Zhang, Guo-Jun Qi, Wei Liu, and Jinhui Tang. 2021. Spatiotemporal co-attention recurrent neural networks for human-skeleton motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 6 (2021), 3300–3315.
[20] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*. 127–140.
[21] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Jiuxin Cao, Yingxia Shao, and Nguyen Quoc Viet Hung. 2021. Heterogeneous hypergraph embedding for graph classification. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. 725–733.
[22] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.
[23] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the International Conference on World Wide Web*. 1067–1077.
[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph Attention Networks. arXiv:1710.10903. Retrieved from https://arxiv.org/abs/1710.10903.
[25] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 968–977.
[26] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *Proceedings of the International Conference on World Wide Web*. 2022–2032.
[27] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.
[28] Shunxin Xiao, Shiping Wang, Yuanfei Dai, and Wenzhong Guo. 2022. Graph neural networks in node classification: Survey and evaluation. *Machine Vision and Applications* 33, 1 (2022), 1–19.
[29] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. HyperGCN: A new method of training graph convolutional networks on hypergraphs. In *Proceedings of the International Conference on Neural Information Processing Systems*. 1511–1522.

[30] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudré-Mauroux. 2020. LBSN2Vec++: Heterogeneous hypergraph embedding for location-based social networks. *IEEE Transactions on Knowledge and Data Engineering* 34, 4 (2020), 1843–1855.

[31] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[32] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph transformer networks. In *Proceedings of the International Conference on Neural Information Processing Systems*, Vol. 32. 11983–11993.

[33] Mengqi Zhang, Shu Wu, Meng Gao, Xin Jiang, Ke Xu, and Liang Wang. 2020. Personalized graph neural networks with attention mechanism for session-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3946–3957.

[34] R. Zhang, Y. Zou, and J. Ma. 2020. Hyper-SAGNN: A self-attention based graph neural network for hypergraphs. In *Proceedings of the International Conference on Learning Representations*.

[35] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

Received 15 December 2021; revised 2 August 2022; accepted 21 September 2022