Computers, Materials & Continua CMC, vol.65, no.3, pp.2277-2294, 2020

Heterogeneous Hyperedge Convolutional Network

Yong Wu[1], Binjun Wang[1,*] and Wei Li[2]

Abstract: Graph convolutional networks (GCNs) have been developed as a general and powerful tool to handle various tasks related to graph data. However, current methods mainly consider homogeneous networks and ignore the rich semantics and multiple types of objects that are common in heterogeneous information networks (HINs). In this paper, we present a Heterogeneous Hyperedge Convolutional Network (HHCN), a novel graph convolutional network architecture that operates on HINs. Specifically, we extract the rich semantics by different metastructures and adopt hyperedge to model the interactions among metastructure-based neighbors. Due to the powerful information extraction capabilities of metastructure and hyperedge, HHCN has the flexibility to model the complex relationships in HINs by setting different combinations of metastructures and hyperedges. Moreover, a metastructure attention layer is also designed to allow each node to select the metastructures based on their importance and provide potential interpretability for graph analysis. As a result, HHCN can encode node features, metastructure-based semantics and hyperedge information simultaneously by aggregating features from metastructure-based neighbors in a hierarchical manner. We evaluate HHCN by applying it to the semi-supervised node classification task. Experimental results show that HHCN outperforms state-of-the-art graph embedding models and recently proposed graph convolutional network models.

Keywords: Graph convolutional networks, heterogeneous information networks, metastructure.

1 Introduction

Convolutional neural networks (CNNs) have been widely used in *Euclidean* data (e.g., images [Luo, Qin, Xiang et al. (2020); Peng, Long, Lin et al. (2019)], text [Liu, Yang, Lv et al. (2019); Zeng, Dai, Wang et al. (2019)] and video [Zhang, Jin, Sun et al. (2018); Xiang, Shen, Qin et al. (2019)]). As an extension, graph convolutional networks (GCNs) have attracted much attention for the purpose of designing CNNs on graphs in recent

[1] School of Information Technology and Cyber Security, People's Public Security University of China, Beijing, 100038, China.

[2] School of Chemistry and Physics, Queensland University of Technology, Brisbane, Queensland, 4001, Australia.

[*] Corresponding Author: Binjun Wang. Email: wangbinjun@ppsuc.edu.cn.

Received: 21 May 2020; Accepted: 03 July 2020.

years. Because graphs can be irregular, the goal of graph convolution is to model a target node by a layer-wise feature propagation based on its neighbors. Due to the ability to capture both graph structure and node features, GCNs have shown superiority in many machine learning tasks, such as semi-supervised node classification [Chiang, Liu, Si et al (2019)], graph classification [Ma, Wang, Aggarwal et al. (2019)] and link prediction [Nathani, Chauhan, Sharma et al. (2019)].

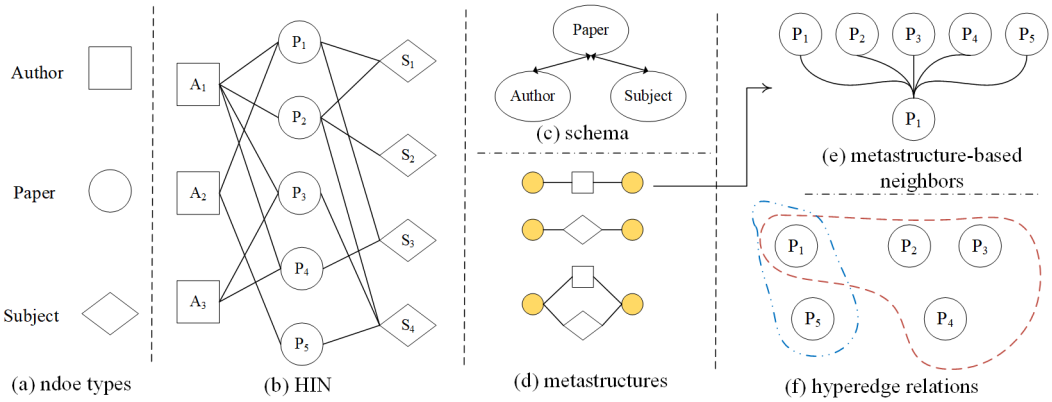

Figure 1: An example of HIN. (a) Three types of nodes: Author, Paper and Subject. (b) A HIN modeled by these three types of nodes. (c) The schema of the HIN. (d) Three metastructures with different semantics. (e) The neighbors of node P1 based on the first metastructure. (f) Two hyperedges composed of the metastructure-based neighbors of P1

Previous GCNs mostly focused on homogeneous information graphs. However, in real-world systems, graphs often contain multiple types of objects and links, which are called heterogeneous information networks (HINs). Compared with homogeneous graphs, the relationships between objects in a HIN are much more complex and have richer semantics. So, it is inappropriate to simply extend previous homogeneous GCNs to HINs. Fig. 1 shows a concrete HIN example containing three types of nodes (Author, Paper and Subject) where the same type nodes are not connected with each other directly (e.g., a paper link to another one through an author or a subject). To capture rich semantics, a network schema of the HIN should be built first. For example, in Fig. 1(c), the network schema is defined over the entity types: Author, Paper and Subject. Then, we can extract semantics by the connections between nodes. A widely used method, metapath [Sun, Han, Yan et al. (2011)], is defined as a sequence of entity types and can obtain semantics effectively. As a common situation, some special semantics are beyond the ability of meta-path. For example, in Fig. 1(d), the semantic of the last illustration is that two papers have the same author and belong to a common subject that cannot be described by the sequence of entity types. Therefore, we propose to use metastructure [Huang, Zheng, Cheng et al. (2016)] to obtain more flexible semantics. The first and second illustrations in Fig. 1(d) are two metastructures that can also be obtained by metapaths P-A-P and P-S-P, while the last illustration in Fig. 1(d) can be described by metastructure but not metapath. It is worth noting that the metastructure is similar to metagraph [Zhang, Yin, Zhu et al. (2018)] conceptually, which has extended metapath to model more complex

semantics. Metagraph can obtain richer structural contexts during the processes of random walks performed on HINs, while metastructures focus more on specific connections among different node types.

Given a metastructure, the metastructure-based neighbors can be obtained which are defined as the set of nodes that connect to a target node via the metastructure. For example, the neighbors of P1 based on the first metastructure in Fig. 1(d) are {P1, P2, P3, P4, P5}. Nevertheless, there is more information between a pair of nodes beyond the semantic contained in a metastructure. For example, considering the neighbors captured above, {P1, P2, P3, P4} are connected by node A1, while {P1, P5} are connected by node A2. Because there are multiple object types, this information is important when modeling a HIN. Therefore, it is necessary to model the high-order relations between metastructure-based neighbors. We propose to adopt hyperedge to preserve these high-order relations. For example, the relation of {P1, P2, P3, P4} can be described by a hyperedge that is defined as the neighbors of node A1, and the relation of {P1, P5} can be described as the neighbors of node A2. It is noted that different hyperedges need to be constructed in different application scenarios.

In a HIN, a target node is influenced by multiple metastructures. Since different metastructures have diverse semantics, how to assign the importance weights of metastructures influences the performance of graph analysis tasks directly. As a result, it is also necessary to assign proper importance to different metastructures. Based on the above considerations, we propose a Heterogeneous Hyperedge Convolutional Network (HHCN) model. We take node features as input and adopt the following two strategies. On the one hand, after the metastructure-based neighbors are captured, we further consider the information of hyperedges that reflect their inner relations and adopt an adaptive renormalization trick during the propagation of node features. On the other hand, to learn the importance of different metastructures on the target node, we introduce the attention mechanism into our model. By assigning proper weights to metastructures, the attention mechanism cannot only improve the quality but also propose the interpretability of our model.

The contributions of this paper can be summarized as follows.

- We propose a novel Heterogeneous Hyperedge Convolutional Network (HHCN) model that uses metastructures to capture semantics and uses hyperedges to capture the high-order relations of the metastructure-based neighbors.
- We introduce an adaptive renormalization trick in the process of hyperedge convolution and adopt an attention mechanism to differentiate the importance of different metastructures on target nodes.
- We conduct experiments on real-world datasets to demonstrate the effectiveness of our proposed HHCN framework.

2 Related work

In this section, we will review the related studies in two aspects: heterogeneous graph embedding and graph convolutional networks.

2.1 Heterogeneous graph embedding

Some previous works focus only on learning node vectors of homogeneous information graphs. For instance, DeepWalk [Perozzi, Al-Rfou and Skiena (2014)] and node2vec [Grover and Leskovec (2016)] conduct random walks and parameterized random walks on a graph respectively to sample linear sequences consisting of nodes, then Skip-gram [Mikolov, Sutskever, Chen et al. (2013)] is used to learn node representations by treating node sequences as word sentences. LINE [Tang, Qu, Wang et al. (2015)] designs objective functions to learn the representations of nodes by modeling the first- and second-order proximities.

To combine the characteristics of heterogeneous graph and graph embedding, two algorithms have been proposed recently for embedding learning in a HIN. To model the different type edges of a HIN, HNE [Chang, Liu, Si et al. (2015)] and PTE [Tang, Qu and Mei (2015)] divide a HIN into multiple bipartite graphs and learn node embeddings by capturing neighborhood relationships between nodes. Considering that the typed edges may not fully align with each other in a HIN, Aspem [Shi, Gui, Zhu et al. (2018)] proposes the concept of aspect and decomposes a HIN into multiple aspects, then the embeddings are derived from these aspects. HEER [Zhang, Lu, Zhou et al. (2016)] embeds HINs via edge representations and combines different aspects into a joint learning process. In Sun et al. [Sun, Han, Yan et al. (2011)], metapath is proposed to acquire semantics in a HIN, based on which many heterogeneous graph embedding methods are also proposed. Metapath2vec [Dong, Chawla and Swami (2017)] captures graph contexts by performing random walks based on metapaths, and a heterogeneous Skip-gram model is used to learn node embeddings. Because one metapath can describe only one type of relationship, metagraph2vec [Zhang, Yin, Zhu et al. (2018)] tries to build metagraphs that contain multiple paths between nodes. Then, metagraph2vec can capture richer structural contexts and semantics between distant nodes. Given a set of relationships specified in forms of metapaths in a HIN, HIN2vec [Fu, Lee and Lei (2017)] converts the learning of embeddings of nodes and metapaths to a relationship prediction task, through which the rich semantics of relationships and the details of the network structure can be captured.

There are also studies that use hyperedges to represent the relations among objects. HEBE [Gui, Liu, Tao et al. (2016)] models the proximities among participating objects in a hyperedge and can preserve more contextual information in embedding learning. To preserve local as well as global hyperedge structural information, DHNE [Tu, Cui, Wang et al. (2018)] proposes a new deep model to realize a non-linear tuple-wise similarity function. However, they regard the hyperedges in a HIN as events that are indecomposable and much semantics is ignored. HGNN [Feng, You, Zhang et al. (2019)] uses hyperedge to deal with more complex connections than pairwise relationships and generalizes the convolution operation to the hyperedge learning process with hyperedge *Laplacian*. Nevertheless, it does not consider the rich semantics between different nodes and is only used in homogeneous graphs.

2.2 Graph convolutional network

Since deep learning has achieved great success on Euclidean data, there is an increasing interest in utilizing neural networks to process the data represented in graph domains. In Scarselli et al. [Scarselli, Gori, Tsoi et al. (2008)], the concept of graph neural network (GNN) was first proposed that extended neural networks to graphs. With different mechanisms being applied to GNN, the methods are divided into spectral-based and spatial-based approaches.

The convolution operation of spectral approaches is formulated in the spectral domain of the graph. Henaff et al. [Henaff, Bruna and LeCun (2015)] utilizes the graph Laplacian eigenbasis to analogize the Fourier transform through which convolution can be performed on general graphs. To simplify the calculation process, a Chebyshev expansion of the graph Laplacian [Defferrard, Bresson and Vandergheynst (2016)] is used to approximate the spectral filters in the spectral domain. Kipf et al. [Kipf and Welling (2016)] approximate the Chebyshev polynomials by the localized first-order neighbors, and a graph convolutional network is proposed.

The spatial-based graph convolution updates the features of the central node by aggregating the features of its neighbors directly. GraphSAGE [Hamilton, Ying and Leskovec (2017)] assumes that different aggregation functions over a fixed-size node neighbor can be used in inductive graphs. In Veličković et al. [Veličković, Cucurull, Casanova et al. (2017)], the attention mechanism is applied, and the influence of different nodes on the target node can be described. To extend graph attention networks to HINs, HAN [Wang, Ji, Shi et al. (2019)] used node- and semantic-level attention to aggregate neighbor information based on metapaths. In contrast, our method utilizes metastructures to obtain semantics and further models richer relations between metastructure-based neighbors by hyperedges.

3 Preliminaries

Definition 1 **Heterogeneous Information Networks (HINs).** A HIN is defined as a graph $G=(V,E,U,\varphi,\phi)$, in which V and E are the sets of nodes and edges, and U is a set of object types and link types. Each node $v\in V$ and each edge $e\in E$ are associated with their mapping functions $\varphi(v):V\to U_V$ and $\phi(e):E\to U_E$ respectively. If $|U_V|+|U_E|>2$, the graph G is a heterogeneous information network. Fig. 1(b) gives an example of HIN which contains three types of objects (Author, Paper, and Subject).

Definition 2 **Network Schema.** A network schema $U_G=(U_V,U_E)$ is a directed graph defined over object types which can be seen as a template for the heterogeneous information network G. For example, Fig. 1(c) is the network schema of the HIN in Fig. 1(b).

Definition 3 **Hyperedge.** A hyperedge ε is a set of objects, representing the semantic information of relationships among multiple objects. In Fig. 1(f), {P1, P2, P3, P4} is an example of hyperedge, and the semantic presents that they all are neighbors of node A1.

Definition 4 **Incident Matrix.** An incident matrix $H_{|V|\times|\varepsilon|}$ is a matrix that shows the relationships between objects V and a hyperedge set Δ in which each row represents an

object and each column represents a hyperedge. If object $v \in V$ belongs to hyperedges $\varepsilon \in \Delta$, $h(v,\varepsilon)=1$; otherwise $h(v,\varepsilon)=0$.

4 Model

In this section, we describe our proposed Heterogeneous Hyperedge Convolutional Network (HHCN) model. After generating the metastructure-based neighbors, our model consists of three steps. The first step builds the incident matrix according to hyperedges and exploits hyperedge convolution to preserve the hyperedge information. The second step tries to combine the different node embeddings learned by hyperedge convolution based on different metastructures and utilizes the attention mechanism to learn the importance weights of each metastructure automatically. The final step involves the application of a multilayer perceptron (MLP) to predict the output and the design of the objective function.

4.1 Hyperedge convolutional network

For a graph with node set V and its corresponding adjacency matrix A, the normalized graph *Laplacian* matrix [Spielman (2007)] of the graph that is real symmetric positive semi-definite is defined as in Eq. (1).

$$L = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{1}$$

where D is a diagonal degree matrix $D_{ii} = \sum_j A_{ij}$, and n is the number of nodes.

Then, the convolution on a graph [Shuman, Narang, Frossard et al. (2013)] is defined as the multiplication of a signal x with a filter g_θ, which is shown in Eq. (2):

$$g_\theta * x = U g_\theta U^T x \tag{2}$$

where U is the eigenvector matrix of the normalized graph *Laplacian* $L = U \Lambda U^T$, $U^T x$ is the graph *Fourier* transform of the input x, Λ is the diagonal matrix of eigenvalues of L and g_θ is a function of the eigenvalues of L. To avoid computationally expensive operations, the spectral filter $g_\theta(\Lambda)$ was approximated by k_{th} order Chebyshev polynomials [Defferrard, Bresson and Vandergheynst (2016)] and the spectral convolution on the graph can be approximated as Eq. (3):

$$g_\theta * x \approx U \sum_{k=0}^{K} \theta_k^{'} T_k\left(\frac{2}{\lambda_{\max}} \Lambda - I_n\right) U^T = \sum_{k=0}^{K} \theta_k^{'} T_k\left(\frac{2}{\lambda_{\max}} L - I_n\right) x \tag{3}$$

where $T_k(\cdot)$ denote the *Chebyshev* polynomial, and $\theta^{'}$ is the Chebyshev coefficients.

In GCN [Kipf and Welling (2016)], the model was further simplified by setting $K=1$, $\lambda_{\max} \approx 2$, $\theta = \theta_0^{'} = -\theta_1^{'}$. Then, Eq. (3) can be simplified as Eq. (4):

$$g_\theta * x \approx \theta \left(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x \tag{4}$$

From the above analysis, it can be concluded that the key to graph convolution is to find the normalized graph Laplacian matrix whose eigenvectors can be used to form Fourier basis.

Considering the situation, we want to utilize hyperedge information in the process of graph convolution, and the normalized graph *Laplacian* matrix should be based on incident matrix H instead of adjacency matrix A. Inspired by Feng et al. [Feng, You, Zhang et al. (2019)], the normalized hyperedge graph Laplacian matrix is defined as Eq. (5):

$$L_o = I_n - D_v^{-\frac{1}{2}} H W D_e^{-1} H^T D_v^{-\frac{1}{2}} \tag{5}$$

where H is the incident matrix, W is a diagonal weight matrix comprising the weights of hyperedges, which is initialized as an identity matrix, n is the number of nodes and D_e is the hyperedge degree diagonal matrix. $(D_e)_{ii} = \sum_{v \in V} h(v, \varepsilon_i)$ for a hyperedge $e = \varepsilon_i$, D_v is the node degree diagonal matrix and $(D_v)_{jj} = \sum_{e \in \varepsilon} w(e) \cdot h(v_j, e)$ for a node $v = V_j$.

By virtue of the property of being real symmetric positive semidefinite of L_o [Feng, You, Zhang et al. (2019)], we can perform *Fourier* transform in an orthonormal space, which is formed by the eigenvectors of L_o. As a result, a convolution formula can be obtained by taking similar steps to GCN, which is shown in Eq. (6):

$$g_\theta * x = \theta(I_n + D_v^{-\frac{1}{2}} H W D_e^{-1} H^T D_v^{-\frac{1}{2}}) x \tag{6}$$

In GCN, a renormalization trick is also assumed to alleviate the problem of numerical instabilities, which is shown in Eq. (7):

$$I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \to \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \tag{7}$$

where $\tilde{A} = A + I_n$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

In Eq. (7), the adjacency matrix is added to an identity matrix, which can be regarded as increasing the interactions of the nodes themselves. However, because the interactions of the nodes themselves have been involved in hyperedges, the strategy may not be suitable. for hyperedge convolution. Here, we adopt an adaptive renormalization trick by setting a learnable weight parameter α, and the hyperedge convolution is defined as Eq. (8):

$$g_\theta * x = \theta(\tilde{D}_v^{-\frac{1}{2}} \tilde{H} W \tilde{D}_e^{-1} \tilde{H}^T \tilde{D}_v^{-\frac{1}{2}}) x \tag{8}$$

where $\tilde{H} = H + \alpha \cdot I_n$, and $\tilde{D}_v$ and $\tilde{D}_e$ are the corresponding diagonal hyperedge matrix and diagonal node degree matrix. Then, the parameter α can be learned automatically.

Eventually, the hyperedge convolution layer can be expressed as Eq. (9):

$$Z^{(l+1)} = \sigma(\tilde{D}_v^{-\frac{1}{2}} \tilde{H} W \tilde{D}_e^{-1} \tilde{H}^T \tilde{D}_v^{-\frac{1}{2}} \theta^l Z^{(l)}) \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid activation function, and $\theta^{(l)}$ is the parameter matrix of the l^{th} layer. $Z^{(l)}$ is the activation matrix in the l^{th} layer, and $Z^{(0)}$ is the input feature matrix.

For the convolution operation in Eq. (8), the difference between HHCN and GCN is the node connection matrix, which is $\tilde{H}\tilde{H}^T$ in HHCN and adjacency matrix A in GCN apart from the normalized term. In practice, the hyperedge incident matrix $\tilde{H}$ is sparse, and the node connection matrix $A_h = \tilde{H}\tilde{H}^T$ of HHCN can be computed efficiently. Then, A_h can also be sparse matrix where a nonzero item means that a pair of nodes have at least a common hyperedge. Considering the input $x \in R^{N \times D}$ with N nodes and D dimensional feature vector for every node features and a matrix of filter parameters $\theta \in R^{D \times F}$ with F filters, the convolution operation of HHCN has complexity $\mathcal{O}(|E_h|DF)$ in comparison to GCN with $\mathcal{O}(|E|DF)$, where $|E_h|$ and $|E|$ are the number of nonzero items in A_h and A respectively.

4.2 Combining multiple metastructures

If we select a set of T network metastructures $\{M_1,...,M_T\}$, then the comprehensive node embedding should be based on all these metastructure-based embeddings. Because different metastructures reflect distinct semantic information, the specific importance of each metastructure is expected. Here, a metastructure attention layer is leveraged to assign different weights to different metastructures and fuse them together automatically. At first, a trainable attention context vector τ is used to denote the preference. A higher attention coefficient will be assigned to a metastructure if it is similar to the context vector.

There are two strategies for learning the attention coefficient of different metastructures here. One strategy is to treat each node in the HIN as an independent individual and assign different weights to the metastructures associated with the node. The other strategy is to regard the nodes extracted from a metastructure as a whole and learn an overall attention coefficient for each metastructure. To facilitate the display of the impacts of the attention mechanism in the following experiments, we select the second strategy for our model. The metastructure-based embedding first undergoes a linear transformation with non-linear activation. Then the attention coefficient of a node is based on the similarity between context vector τ and the transformed embedding. For a metastructure M_i, its attention coefficient w_{M_i} can be achieved by averaging the attention coefficients of the nodes extracted from it, which is shown in Eq. (10):

$$w_{M_i} = \frac{1}{|M_i|} \sum_{j \in M_i} \tau^T \cdot \tanh(W \cdot z_j^{M_i} + b) \tag{10}$$

where W is the weight, b is the bias parameter of the linear transformation and $|M_i|$ is the number of nodes extracted from M_i.

In addition, the softmax function is used to normalize the importance of metastructure M_i, and the weight β^{M_i} is obtained as Eq. (11):

$$\beta^{M_i} = soft\max(w_{M_i}) = \frac{\exp(w_{M_i})}{\sum_{k=1}^{T} \exp(w_{M_i})} \tag{11}$$

Lastly, the comprehensive node embedding can be obtained by weighted combining of all the metastructure-based embeddings, which is shown in Eq. (12):

$$Z = \sum_{k=1}^{T} \beta^{M_k} \cdot Z_{M_k} \tag{12}$$

4.3 Loss function

Given a HIN G with n nodes, a labeling function $\gamma(v): V \to L$ can be used to map a node to one of the labels in $\{1,...,L\}$.

When all or part of the nodes are labeled, a one-layer MLP is applied on the node embeddings and predict the labels. The objective is to minimize the cross-entropy loss between the ground truth and the predictions, which is shown in Eq. (13):

$$J = -\sum_{v \in M} \sum_{l=1}^{L} Y_{vl} \cdot \ln(P_{vl}) \tag{13}$$

where M is the set of labeled nodes, Y_{vl} is a binary value indicating the true label of node v (i.e., Y_{vl}=1 if the true label of node v is l, zero otherwise) and P_{vl} is a binary value indicating the predicted label of node v. After computing the loss, the parameters are updated by back-propagating the gradient.

5 Experiments

In this section, we compare our proposed model with state-of-the-art graph embedding models and recently developed graph convolutional network models.

5.1 Datasets

We evaluate our models on two datasets, which are also used in Wang et al. [Wang, Ji, Shi et al. (2019)].

- IMDB[3]. This is a heterogeneous information network with movies 4430(M), 5627 actors (A), 2137 directors (D), 11,841relations (P-A) and 4430 relations (P-S). The features of a movie consist of the elements of a bag-of-words represented by plots, and the dimension is 1232. The label is the corresponding genre, which is divided into three classes, and the labeling ratio is 66% in our experiments. We select the metastructure set as M-A-M, M-D-M and M<A, D>M, in which the first two metastructures can also be seen as metapath and used in the metapath-based models.
- ACM[4]. This is a heterogeneous information network with 2835 papers (P), 3087 authors (A), 45 subjects (S), 8504 relations (P-A) and 2835 relations (P-S) from three areas: Database, Wireless Communication and Data Mining. The feature of a paper consists of the elements of a bag-of-words represented by keywords, and the dimension is 1830. The label is the corresponding conference that was held, and all the papers are

[3] https://www.imdb.com/

[4] http://dl.acm.org/

labeled in our experiments. For a fair comparison, we select the metastructure set as P-A-P, P-S-P and P<A, S>P, in which the first two metastructures can also be seen as metapaths and used in the metapath-based models.

5.2 Baselines

Eight models are adopted to evaluate HHCN: DeepWalk [Perozzi, Al-Rfou and Skiena (2014)], LINE [Tang, Qu, Wang, et al. (2015)], metapath2vec [Dong, Chawla and Swami (2017)], HIN2vec [Fu, Lee and Lei (2017)], GCN [Kipf and Welling (2016)], HGNN [Feng, You, Zhang et al. (2019)], GAT [Veličković, Cucurull, Casanova et al. (2017)] and HAN [Wang, Ji, Shi et al. (2019)]. The first four models are methods based on representation learning, and the other four are methods based on graph convolutional networks. We also propose three variants of our model.

- HHCN-f. This variant does not perform the fusing operation. For the experiments, we selected the metastructure that achieves the best performance.
- HHCN-a. This variant does not utilize the attention mechanism and regards all the metastructures as equally important.
- HHCN-s. This variant uses the {M-A-M, M-D-M} and {P-A-P, P-S-P} as the metastructure set, through which fair comparisons with metapath-based models can be performed.

5.3 Setup

A percentage of the labeled nodes per class are selected for training. The selection ratio ranged from 10% to 90% and the remaining labeled nodes were used for testing. A two-layer HGNN was applied in our model, and accuracy was used as the evaluation metric. The hyperedge set of each metastructure was defined as the neighbors of the middle node combination in the metastructure. For nodes that had no neighbors based on M<A, D>M in IMDB and P<A, S>P in ACM, we used neighbors based on M-D-M and P-A-P to fill them. We initialized parameters randomly and optimized the model with Adam [Kingma and Ba (2014)]. We set learning rate=0.001, regularization coefficient=0.0005 and dropout rate=0.5. For the compared methods, we used the code provided by authors. For unsupervised methods, we first learned embedding for each node, then used the training set to train a logistic regression classifier and output the results through the testing set. For the random walk-based method, we set the number of walks per node=40, the walk length=100, window size=5 and the size of negative sampling=5. In the HGNN, the hyperedge was set as the set of node neighbors that was used in the original paper. In LINE, the learning rate of the starting value=0.025, the number of negative samples=5 and the total number of samples=100 million. In DeepWalk and LINE, the heterogeneous graphs were regarded as homogeneous graphs. For all the methods, we set the vector dimension=64, the dimension of context vector =100, the vector dimension of all the methods=64 and the dimension of attention vector τ =100.

5.4 Experimental results

For both datasets, the results are reported in Tab. 1, in which the highest value of each row has been bolded. For metapath2vec, GCN, HGNN and GAT, we select the metapath with the best classification performance and report the classification results. From Tab. 1, we can see that:

Table 1: Results of node classification on two datasets

Datasets	Train	DeepWalk	LINE	Hin2vec	metapath2vec	GCN	HGNN	GAT	HAN	HHCN-r	HHCN-h	HHCN-s	HHCN
ACM	10%	0.7285	0.6545	0.7252	0.7513	0.8840	0.8844	0.8859	0.8931	0.9041	0.9012	0.9036	**0.9045**
	20%	0.7653	0.6569	0.7459	0.7516	0.8968	0.8950	0.8908	0.9048	0.9120	0.9086	0.9129	**0.9141**
	30%	0.7735	0.6603	0.7463	0.7518	0.8971	0.8982	0.8972	0.9092	0.9143	0.9197	0.9213	**0.9219**
	40%	0.7742	0.6676	0.7473	0.7536	0.8997	0.8978	0.9019	0.9096	0.9163	0.9318	0.9312	**0.9325**
	50%	0.7755	0.6688	0.7496	0.7543	0.9003	0.8990	0.9028	0.9103	0.9187	0.9329	0.9325	**0.9340**
	60%	0.7772	0.6711	0.7560	0.7554	0.9008	0.9001	0.9032	0.9113	0.9193	0.9343	0.9348	**0.9361**
	70%	0.7784	0.6763	0.7591	0.7560	0.9037	0.9007	0.9057	0.9222	0.9232	0.9353	0.9362	**0.9370**
	80%	0.7786	0.6774	0.7594	0.7589	0.9139	0.9117	0.9125	0.9327	0.9354	0.9425	0.9430	**0.9436**
	90%	0.7789	0.6783	0.7696	0.7696	0.9404	0.9437	0.9358	**0.9552**	0.9466	0.9485	0.9535	0.9544
IMDB	10%	0.3313	0.3511	0.3521	0.3639	0.3841	0.3900	0.3870	0.3989	0.3880	0.4194	0.4220	**0.4231**
	20%	0.3404	0.3510	0.3626	0.3667	0.4486	0.4466	0.4473	0.4533	0.4503	0.4596	0.4686	**0.4694**
	30%	0.3526	0.3521	0.3670	0.3669	0.4523	0.4592	0.4546	0.4653	0.4625	0.4817	0.4828	**0.4852**
	40%	0.354	0.3538	0.3839	0.3751	0.4627	0.4698	0.4684	0.4760	0.4775	0.4992	0.5035	**0.5048**
	50%	0.3694	0.3626	0.3983	0.3776	0.4685	0.4669	0.4746	0.4898	0.4832	0.5132	0.5154	**0.5170**
	60%	0.3733	0.3713	0.3987	0.3786	0.4819	0.4806	0.4879	0.5168	0.5083	0.5208	0.5260	**0.5286**
	70%	0.3756	0.3733	0.4005	0.3861	0.5160	0.5166	0.5196	0.5231	0.5259	0.5419	0.5444	**0.5452**
	80%	0.3807	0.3843	0.4130	0.3960	0.5333	0.5360	0.5346	0.5520	0.5476	0.5583	0.5626	**0.5644**
	90%	0.3858	0.3936	0.4234	0.4026	0.5504	0.5566	0.5565	0.5704	0.5595	0.5861	0.5957	**0.5966**

(1) Graph neural network-based methods that combine the structure and feature information usually perform better than graph embedding methods. In particular, our model HHCN achieved the best performance of all the compared algorithms.

(2) GCN, HGNN, GAT and HHCN-f are graph convolutional network models that consider a single metastructure. The performance of HGNN is similar to that of GCN, because HGNN is used in homogeneous networks, and no additional and more complex information is merged into the generated hypergraph structure. GAT, which learns weights for neighbors, showed a slight improvement over GCN, while HHCN-f displayed the best performance. This demonstrates that more meaningful relation information can be captured by the hyperedges considered in our method for a HIN and better results can be achieved.

(3) Compared with HAN, which uses the attention mechanism on both of the nodes and semantics, our model HHCN-s, which uses the same metapaths, achieves better performance most of the time. This also suggests that it is necessary to preserve the information of hyperedges in HINs.

(4) Our model also shows better performance than the variants HHCN-a and HHCN-s, which indicates there are benefits to the attention mechanism and metastructure. Compared with HHCN-a, it is reasonable for our model to consider the different

influences of metastructures. Compared with HHCN-s, reasonable metastructure can provide more meaningful semantic information.

5.5 Attention analysis

Because different weights for different metastructures can be obtained through the attention mechanism, the analysis on attention was conducted here. We selected 50% of the labeled nodes as the training set, and the remaining 50% as the testing set.

Tab. 2 lists the attention coefficients of the different metastructures and the performance when taking into consideration a single metastructure. As we can see, metastructure with higher accuracy has a larger attention coefficient, which shows they are correlated positively in both datasets. Specifically, metastructure P-A-P is assigned a larger weight than metastructure P-S-P, and P<A, S>P has the largest weight in ACM. This is explained by the fact that papers with the same authors tend to have the same labels more often than papers with a common subject, and papers with both aspects show a larger trend to have the same labels. In IMDB, metastructure M-D-M has the larger weight, meaning that having a common director determines the genre of a movie more than having a common actor, and metastructure M<A, D>M has the largest weight for its tighter restriction, which is also reasonable. To summarize, the learned attention coefficients can properly identify the quality metastructures.

Table 2: Quality of single motifs and the corresponding attention coefficients

Dataset	ACM			IMDB		
Meta structure	P-A-P	P-S-P	P<A, S>P	M-A-M	M-D-M	M<A, D>M
Accuracy	0.9089	0.7568	0.9187	0.4429	0.4796	0.4832
Attention coefficient	0.3432	0.1677	0.4891	0.2959	0.3325	0.3716

5.6 Adaptive renormalization trick

The renormalization trick used in our model is $\tilde{H} = H + \alpha \cdot I_n$. Here, we compare different renormalization tricks, which are shown in Tab. 3. HHCN-1 uses the same trick as GCN by adding an identity matrix, while HHCN-0 considers the first-order term only.

Table 3: Different renormalization tricks

HHCN	$\theta(\tilde{D}_v^{-\frac{1}{2}}\tilde{H}\tilde{D}_e^{-1}\tilde{H}^T\tilde{D}_v^{-\frac{1}{2}})x$ $\tilde{H} = H + \alpha \cdot I_n$
HHCN-1	$\theta(\tilde{D}_v^{-\frac{1}{2}}\tilde{H}\tilde{D}_e^{-1}\tilde{H}^T\tilde{D}_v^{-\frac{1}{2}})x$ $\tilde{H} = H + I_n$
HHCN-0	$\theta(D_v^{-\frac{1}{2}}HWD_e^{-1}H^TD_v^{-\frac{1}{2}})x$

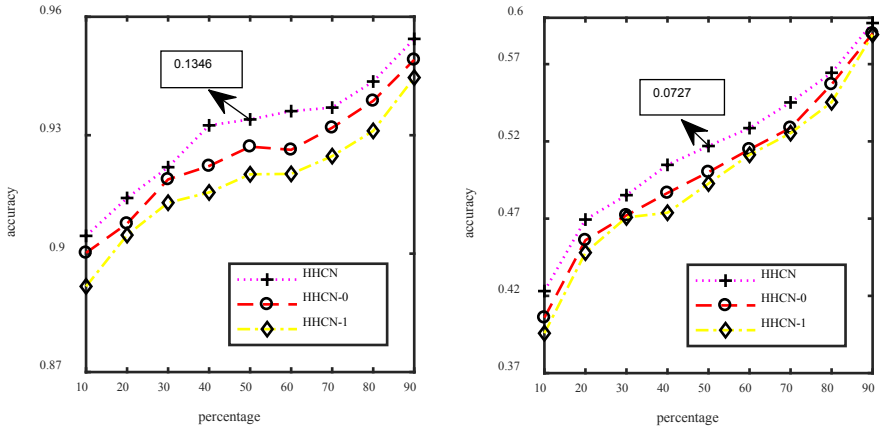

Figure 2: Qualities of different renormalization tricks on ACM and IMDB

The comparison results are demonstrated in Fig. 2. We can see that

- HHCN-1 and HHCN-0 both consider the hyperedge information, but HHCN-1 performs worse than HHCN-0. These results show that using the trick employed in GCN for direct hyperedge convolution is inappropriate.
- HHCN achieves the best performances, which demonstrates the necessity to assign the proper proportion of incident matrix and identity matrix in the design of the propagation model. For example, when the training ratio is set to 50%, the learned coefficient α is 0.1346 on ACM and 0.0727 on IMDB. As a result, the corresponding effects are improved significantly.

5.7 Parameter and efficiency analysis

The number of hyperedge convolution layers and the dimension of node embedding can affect the results of node classification. We select 50% labeled nodes are selected as the training set and the remaining half as the testing set. When analyzing the number of hyperedge convolution layers, we set the node embedding dimension $d = 64$. When analyzing the dimension of node embedding, we set the number of hyperedge convolution layers as $k = 2$.

5.7.1 Hyperedge convolution layer

Fig. 3 shows that HHCN achieves the best performance when there are two hyperedge convolution layers on both graphs. However, when more layers are added, the performance drops sharply. This phenomenon can be explained by the fact that the layers control the receptive field of a target node, and more information can be obtained when layers are initially increased, but too many layers lead to oversmoothing.

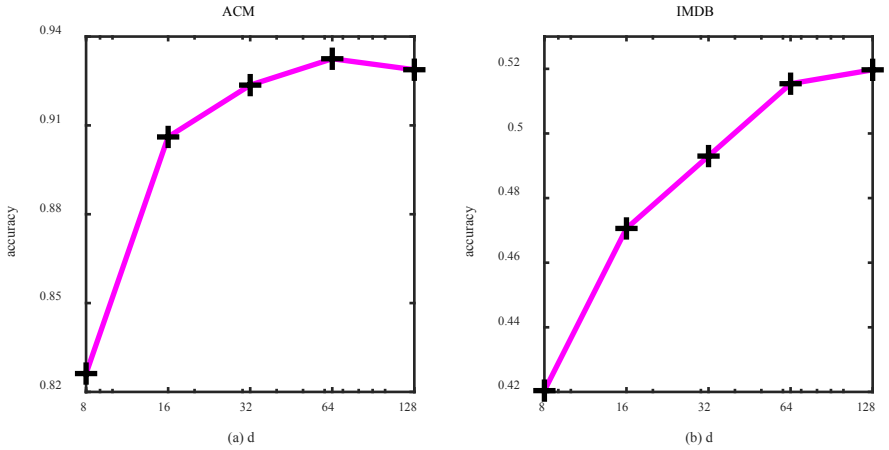

Figure 3: Results of HHCN with respect to hyperedge convolution layer

5.7.2 Node embedding dimension

As shown in Fig. 4, the performance improves with the embedding dimension increasing at first, which demonstrates that a large dimension can capture more information. Nevertheless, when the dimension increases from 64 to 128, the performance on dataset ACM begins to decline and the performance on dataset IMDB increases slightly, which indicates that the introduction of redundant information to node embedding is inevitable when the dimension is too large and may lead to overfitting as well.

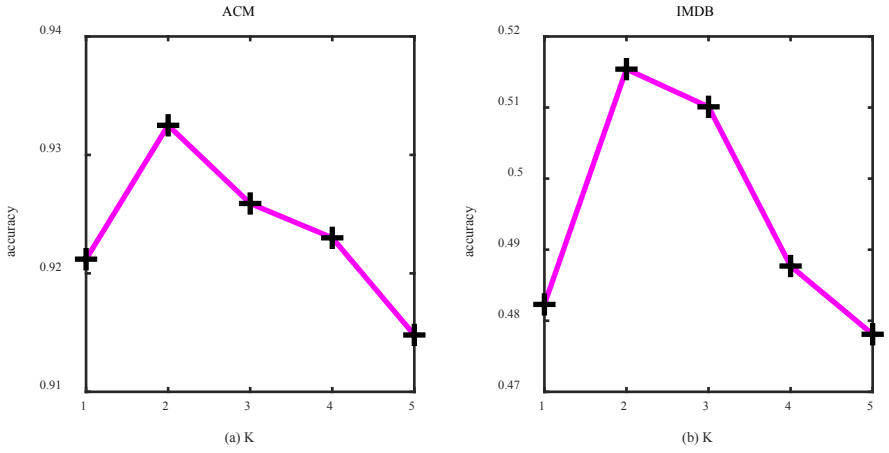

Figure 4: Results of HHCN with respect to dimension of node embedding

5.7.3 Computational efficiency

We compare the model training time per epoch of HHCN versus other graph neural network-based methods on an Intel(R) Core (TM) i5-4200H CPU @2.80 GHz system with 4 cores and 12 GB memory. For fair comparison, we choose GCN_h and GAT_h as the compared methods and set the metastructures in ACM and IMDB as {P-A-P, P-S-P}

and {M-A-M, M-D-M}. Different from HHCN, GCN_h and GAT_h use GCN and GAT to aggregate the features of meta structure-based neighbors respectively. As shown in Fig. 5, it is can be found that HHCN is quite efficient in practice and takes much less time per epoch than GAT_h that needs to perform a large amount of time-consuming attention mechanism operations on both datasets. What's more, HHCN is reasonably close to GCN_h owing to the sparseness of the hyperedge incident matrix.

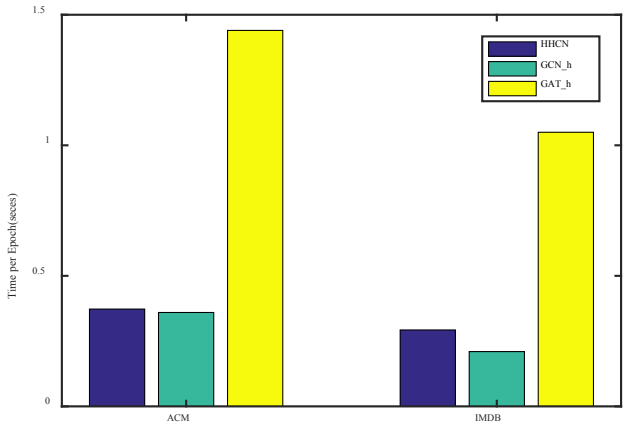

Figure 5: Comparison of time per epoch

6 Conclusion and future work

In this paper, we introduce a novel Heterogeneous Hyperedge Convolutional Network (HHCN) for HIN. Our HHCN model uses metastructures to extract semantics and construct an incident matrix based on hyperedges. An attention mechanism is also introduced to our model so that the importance of different metastructures may be taken into consideration. Taking the node feature matrix as input, the proposed HHCN model can encode node features, metastructure-based semantics and hyperedge information simultaneously. In this setting, our model achieves better performance than several recently proposed models. For our future work, we first plan to extend HHCN from a spectral-based approach to a spatial-based one, which is more efficient in many scenarios. Thereafter, more complex hyperedges will be considered in order to model a HIN better. Finally, we plan to examine the effect of HHCN on larger and more complex attributed heterogeneous graphs and adopt neighborhood sampling strategies to further enable our method to scale to very large graphs.

Acknowledgement: The authors sincerely acknowledge the reviewers for their suggestions which will help in improving the quality of the paper.

Availability of Data and Materials: The code and data involved in this paper are here: https://github.com/timeflow-lab/HHCN.

Funding Statement: This research was funded by The Science and Technology Strengthening Police Basic Program of Ministry of Public Security (2018GABJC03) and The Technology Research Project Program of Ministry of Public Security (2018JSYJA02).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Chang, S.; Han, W.; Tang, J.; Qi, G. J.; Aggarwal, C. C. et al. (2015): Heterogeneous network embedding via deep architectures. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 119-128.

Chiang, W. L.; Liu, X.; Si, S.; Li, Y.; Bengio, S. et al. (2019): Cluster-GCN: an efficient algorithm for training deep and large graph convolutional networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 257-266.

Defferrard, M.; Bresson, X.; Vandergheynst, P. (2016): Convolutional neural networks on graphs with fast localized spectral filtering. *Proceedings of advances in Neural Information Processing Systems*, vol. 29, pp. 3844-3852.

Dong, Y.; Chawla, N. V.; Swami, A. (2017): metapath2vec: Scalable representation learning for heterogeneous networks. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 135-144.

Feng, Y.; You, H.; Zhang, Z.; Ji, R.; Gao, Y. (2019): Hypergraph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3558-3565.

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

Grover, A.; Leskovec, J. (2016): node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855-864.

Gui, H.; Liu, J.; Tao, F.; Jiang, M.; Norick, B. et al. (2016): Large-scale embedding learning in heterogeneous event data. *Proceedings of the 16th IEEE International Conference on Data Mining*, pp. 907-912.

Hamilton, W.; Ying, Z.; Leskovec, J. (2017): Inductive representation learning on large graphs. *Proceedings of advances in Neural Information Processing Systems*, pp. 1024-1034.

Henaff, M.; Bruna, J.; LeCun, Y. (2015): Deep convolutional networks on graph-structured data. arXiv:1506.05163.

Huang, Z; Zheng, Y; Cheng, R; Sun, Y, Mamoulis, N. et al. (2016): Meta structure: Computing relevance in large heterogeneous information networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1595-1604.

Kingma, D. P.; Ba, J. (2014): Adam: A method for stochastic optimization. arXiv:1412.6980.

Kipf, T. N.; Welling, M. (2016): Semi-supervised classification with graph convolutional networks. arXiv:1609.02907.

Liu, J.; Yang, Y. H.; Lv, S. Q.; Wang, J.; Chen, H. (2019): Attention-based BiGRU-CNN for Chinese question classification. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-019-01344-9.

Luo, Y. J.; Qin, J. H.; Xiang, X. Y.; Tan, Y.; Liu, Q. et al. (2020): Coverless real-time image information hiding based on image block matching and dense convolutional network. *Journal of Real-Time Image Processing*, vol. 17, no. 1, pp. 125-135.

Ma, Y.; Wang, S.; Aggarwal, C. C.; Tang, J. (2019): Graph convolutional networks with eigenpooling. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 723-731.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; Dean, J. (2013): Distributed representations of words and phrases and their compositionality. *Proceedings of advances in Neural Information Processing Systems*, pp. 3111-3119.

Nathani, D.; Chauhan, J.; Sharma, C.; Kaul, M. (2019): Learning attention-based embeddings for relation prediction in knowledge graphs. *Proceedings of 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4710-4723.

Peng, F.; Long, Q.; Lin, Z. X.; Long, M. (2019): A reversible watermarking for authenticating 2D CAD engineering graphics based on iterative embedding and virtual coordinates. *Multimedia Tools and Applications*, vol. 78, no. 19, pp. 26885-26905.

Perozzi, B.; Al-Rfou, R.; Skiena, S. (2014): Deepwalk: Online learning of social representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701-710.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; Monfardini, G. (2008): The graph neural network model. *IEEE Transactions on Neural Networks*, vol. 20 no. 1, pp. 61-80.

Shi, Y.; Gui, H.; Zhu, Q.; Kaplan, L.; Han, J. (2018): Aspem: Embedding learning by aspects in heterogeneous information networks. *Proceedings of the SIAM International Conference on Data Mining*, pp. 144-152.

Shuman, D. I.; Narang, S. K.; Frossard, P.; Ortega, A.; Vandergheynst, P. (2013): The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, vol. 30 no. 3, pp. 83-98.

Spielman, D. A. (2007): Spectral graph theory and its applications. *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pp. 29-38.

Sun, Y.; Han, J.; Yan, X.; Yu, P. S.; Wu, T. (2011): Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, vol. 4 no. 11, pp. 992-1003.

Tang, J.; Qu, M.; Mei, Q. (2015): Pte: Predictive text embedding through large-scale heterogeneous text networks. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165-1174.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J. et al. (2015): Line: Large-scale information network embedding. *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067-1077.

Tu, K.; Cui, P.; Wang, X.; Wang, F.; Zhu, W. (2018): Structural deep embedding for hyper-networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 426-433.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P. et al. (2017): Graph attention networks. arXiv: 1710.10903.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y. et al. (2019): Heterogeneous graph attention network. *Proceedings of the 28th International Conference on World Wide Web*, pp. 2022-2032.

Wu, Y.; Liu, H.; Yang, Y. (2018): Graph convolutional matrix completion for bipartite edge prediction. *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pp. 49-58.

Xiang, L. Y.; Shen, X. B.; Qin, J. H.; Hao, W. (2019): Discrete multi-graph hashing for large-scale visual search. *Neural Processing Letters*, vol. 49, no. 3, pp. 1055-1069.

Zeng, D. J.; Dai, Y.; Li, F.; Wang, J.; Sangaiah, A. K. (2019): Aspect based sentiment analysis by a linguistically regularized CNN with gated mechanism. *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 5, pp. 3971-3980.

Zhang, D.; Yin, J.; Zhu, X.; Zhang, C. (2018): Metagraph2vec: Complex semantic path augmented heterogeneous network embedding. *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 196-208.

Zhang, J. M.; Jin, X. K.; Sun, J.; Wang, J.; Sangaiah, A. K. (2018): Spatial and semantic convolutional features for robust visual object tracking. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-018-6562-8.

Zhang, J.; Lu, C. T.; Zhou, M.; Xie, S.; Chang, Y. et al. (2016): Heer: Heterogeneous graph embedding or emerging relation detection from news. *Proceedings of the IEEE International Conference on Big Data*, pp. 803-812.