Multiplex Heterogeneous Graph Convolutional Network

Pengyang Yu
Ocean University of China
Qingdao, China
ypy@stu.ouc.edu.cn

Chaofan Fu
Ocean University of China
Qingdao, China
fuchaofan@stu.ouc.edu.cn

Yanwei Yu*
Ocean University of China
Qingdao, China
yuyanwei@ouc.edu.cn

Chao Huang*
The University of Hong Kong
Hong Kong, China
chaohuang75@gmail.com

Zhongying Zhao
Shandong University of Science and Technology
Qingdao, China
zyzhao@sdust.edu.cn

Junyu Dong
Ocean University of China
Qingdao, China
dongjunyu@ouc.edu.cn

ABSTRACT

Heterogeneous graph convolutional networks have gained great popularity in tackling various network analytical tasks on heterogeneous network data, ranging from link prediction to node classification. However, most existing works ignore the relation heterogeneity with multiplex network between multi-typed nodes and different importance of relations in meta-paths for node embedding, which can hardly capture the heterogeneous structure signals across different relations. To tackle this challenge, this work proposes a **M**ultiplex **H**eterogeneous **G**raph **C**onvolutional **N**etwork (MHGCN) for heterogeneous network embedding. Our MHGCN can automatically learn the useful heterogeneous meta-path interactions of different lengths in multiplex heterogeneous networks through multi-layer convolution aggregation. Additionally, we effectively integrate both multi-relation structural signals and attribute semantics into the learned node embeddings with both unsupervised and semi-supervised learning paradigms. Extensive experiments on five real-world datasets with various network analytical tasks demonstrate the significant superiority of MHGCN against state-of-the-art embedding baselines in terms of all evaluation metrics. The source code of our method is available at: https://github.com/NSSSJSS/MHGCN.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Network Embedding; Graph Representation Learning; Multiplex Heterogeneous Networks; Graph Convolutional Networks

*Yanwei Yu and Chao Huang are the corresponding authors.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539482

ACM Reference Format:
Pengyang Yu, Chaofan Fu, Yanwei Yu, Chao Huang, Zhongying Zhao, and Junyu Dong. 2022. Multiplex Heterogeneous Graph Convolutional Network. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539482

1 INTRODUCTION

Network representation learning has emerged as a new learning paradigm to embed complex network into a low-dimensional vector space while preserving the proximities of nodes in both network topological structures and intrinsic properties. Effective network representation advances various network analytical tasks, ranging from link prediction [3, 17, 22], node classification [6, 18, 32], to recommendation [12, 13, 25]. In recent years, Graph Convolutional Networks (GCNs) [15], a class of neural networks designed to learn graph representation for complex networks with rich feature information, have been applied to many online services, such as E-commerce [35], social media platforms [33] and advertising [8].

While many efforts have been made to study the representation learning over homogeneous graphs [6, 15, 23, 26], the exploration of preserving network heterogeneous properties in graph representation paradigms has attracted much attention in recent studies, *e.g.*, metapath2vec [4] and HERec [25]. Inspired by the strength of Graph Neural Networks (GNNs) in aggregating contextual signals from neighboring nodes, various graph neural models have been introduced to tackle the challenge of heterogeneous graph learning, such as HAN [28], MAGNN [5] and HetGNN [39].

Albeit the effectiveness of existing heterogeneous network embedding methods [4, 13, 20, 29], these works are generally designed for heterogeneous networks with a single view. In real-world scenarios, however, many networks are much more complex, comprising not only multi-typed nodes and diverse edges even between the same pair-wise nodes but also a rich set of attributes [1]. For example, in E-commerce networks, there are two types of nodes (*i.e.*, users and items), and multiple relations (*e.g.*, click, purchase, add-to-cart, or add-to-preference) between the same pairs of users and items [34]. The connections between multiple types of nodes in such networks are often heterogeneous with relation diversity, which yields networks with multiple different views. It is worth noting that the multiplicity of the network is fundamentally different from the heterogeneity of the network. Two types of nodes, users and items, in a E-commerce network reflect the heterogeneity

of the network. At the same time, users may have several types of interactions (*e.g.*, click, purchse, review) with items [31], which reflects the multiplex relationships of the network. Because different user-item interactions exhibit different views of user and item, and thus should be treated differently. We term this kind of networks with both multiplex network structures with multi-typed nodes and node attribute information as ***attributed multiplex heterogeneous networks*** (AMHENs).

Performing representation learning on the AMHENs is of great importance to network mining tasks, yet it is very challenging due to such complicated network structures and node attributes. While some recent studies propose to solve the representation learning problem on multiplex heterogeneous network [1, 18, 21, 36, 38], several key limitations exist in those methods. i) The success of current representation learning models largely relies on the accurate design of meta-paths. How to design an automated learning framework to explore the complex meta-path-based dependencies over the multiplex heterogeneous graphs, remains a significant challenge. ii) Unlike the homogeneous node aggregation scheme, with the heterogeneous node types and multiplex node relationships, each meta-path can be regarded as relational information channel. An effective meta-path dependency encoder is a necessity to inject both the relation heterogeneity and multiplexity into the representations. iii) In real-world graph representation scenarios, efficiency is an important factor to handle the graph data with large number of heterogeneous nodes and multiplex edges. However, most current methods are limited to serve the large-scale network data, due to their high time complexity and memory consumption.

To address the aforementioned challenges, we propose a new **M**ultiplex **H**eterogeneous **G**raph **C**onvolutional **N**etwork, named **MHGCN**, for AMHEN embedding. Specifically, we first decouple the multiplex network into multiple homogeneous and bipartite sub-networks, and then re-aggregate the sub-networks with the exploration of their importance (*i.e.*, weights) in node representation learning. To automatically capture meta-path information across multi-relations, we tactfully design a multilayer graph convolution module, which can effectively learn the useful heterogeneous meta-path interactions of different lengths in AMHENs through multilayer convolution aggregation in both unsupervised and semi-supervised learning paradigms. To improve the model efficiency, we endow our MHGCN with a simplified graph convolution for feature aggregation, in order to significantly reduce the model computational cost. Our evaluations are conducted on several real-world graph datasets to evaluate the model performance in both link prediction and node classification tasks. Experimental results show that our MHGCN framework can obtain the substantial performance improvement compared with state-of-the-art graph representation techniques. With the designed graph convolution module, our MHGCN achieves better model efficiency when competing with state-of-the-art GNN baselines for AMHENs by up to two orders of magnitudes (see efficiency analysis in the supplemental material). .

We summarize the contributions of this paper as follows:

- We propose an effective multiplex heterogeneous graph neural network, MHGCN, which can automatically capture the useful relation-aware topological structural signals between nodes for heterogeneous network embedding.
- MHGCN integrates both network structures and node attribute features in node representations, and gains the capability to efficiently learn network representation with a simplified convolution-based message passing mechanism.
- We conduct extensive experiments on five real-world datasets to verify the superiority of our proposed model in both link prediction and node classification when competing with state-of-the-art baselines.

2 RELATED WORK

Graph Neural Networks. The goal of a GNN is to learn a low-dimensional vector representation for each node, which can be used for many downstream network mining tasks. Kipf *et al.* [15] proposes to perform convolutional operations over graph neighboring node for information aggregation. GraphSAGE [7] is an inductive GNN framework, which uses the general aggregating functions for efficient generation of node embeddings. To differentiate the influence of neighboring nodes, GAT [27] has been proposed as an attentive message passing mechanism to learn the explicit weights of neighbor node embeddings. R-GCN [24] considers the influence of different edge types on nodes, and uses weight sharing and coefficient constraints to apply to multi-graphs with large numbers of relations. To simplify the design of graph convolutional network, LightGCN [9] omits the embedding projection with non-linearity during the message passing. Additionally, AM-GCN [30] is proposed to adaptively learn deep correlation information between topological structures and node features. However, all algorithms mentioned above are developed for the homogeneous networks, and thus cannot effectively preserve the heterogeneous and multiplex graph characteristics for the network representation task.

Heterogeneous Graph Representation. Modeling the heterogeneous context of graphs has already received some attention [4, 19, 25, 39]. For example, some studies leverage random walks to construct meta-paths over the heterogeneous graph for node embeddings, including metapath2vec [4] and HERec [25]. As graph neural networks (GNNs) have become a popular choice for encoding graph structures, many heterogeneous graph neural network models are designed to enhance the GNN architecture with the capability of capturing the node and edge heterogeneous contextual signals. For example, HetGNN [39] jointly encodes the graph topology and context heterogeneity for representation learning. HeGAN [10] incorporates generative adversarial networks (GAN) for heterogeneous network embedding. NARS [37] first generates relation subgraphs, learns node embeddings by 1D convolution on the subgraphs and then aggregates the learned embeddings. Fu *et al.* [5] performs both the intra- and inter-metapath aggregation so as to distill the metapath-based relational context for learning node representations. However, most of those approaches rely on selecting the useful metapaths to guide the process of heterogeneous representation, which may need the external domain knowledge for constructing relevant metapaths.

In addition, there exist some recent studies attempting to relax the requirement of metapath construction for heterogeneous graph

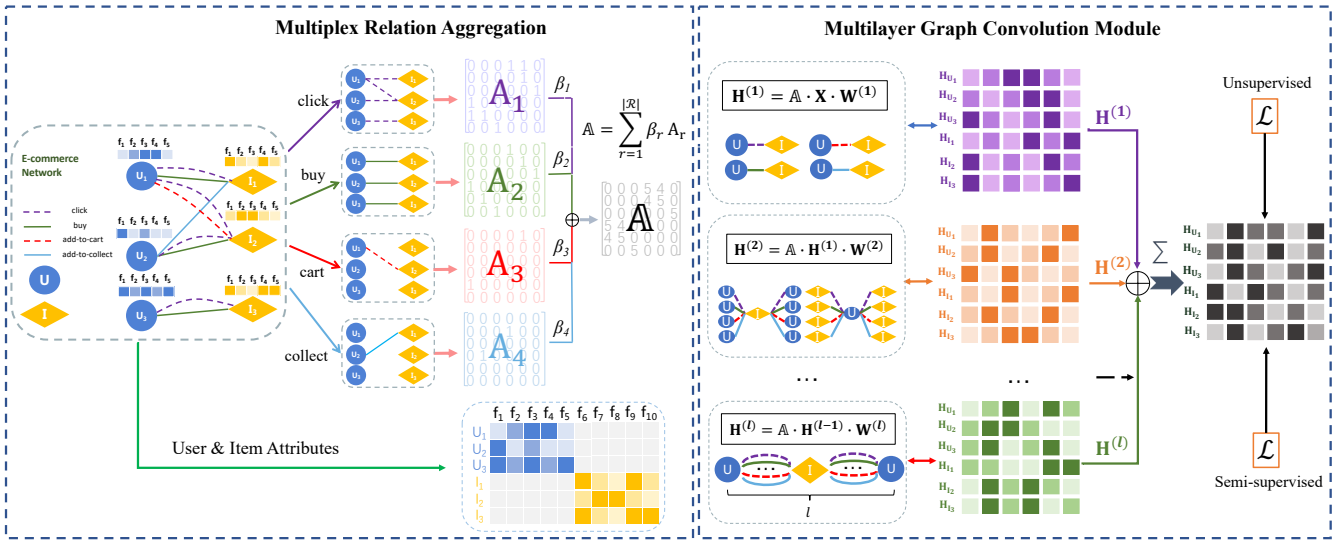


Figure 1: The overview of the proposed MHGCN.

representations. In particular, HGT [11] proposes to incorporate the self-attention into the graph-based message passing mechanism for modeling the dynamic dependencies among heterogeneous nodes. HPN [13] eliminates semantic confusion by mapping nodes in meta-path to semantic space, and then aggregates the embeddings of nodes under different metapaths to obtain the final representation. However, most of the above heterogeneous graph embedding models ignore the multiplex relational context of real-life graph data, in which multi-typed relationships exist among nodes.

Multiplex Heterogeneous Network Embedding. Real-world graphs are often inherently multiplex, which involves various relations and interactions between two connected nodes. To tackle this challenge, many multiplex network embedding techniques are proposed to project diverse node edges into latent representations. For example, MNE [40] introduces a global transformation matrix for each layer of the network to align the embeddings with different dimensions for each relation type. GATNE [1] splits the node representation by learning base embedding, edge embedding as well as attribute embedding. The self-attention is utilized to fuse neighborhood information for generating edge representation. Motivated by the mutual information maximization scheme, DMGI [21] is proposed as an unsupervised learning approach which aims to minimize the difference among relation-aware node representations. HGSL [42] first obtains the node representation based on meta-paths, and then uses GNN to jointly train the heterogeneous graph, node representation and node attributes to obtain the final embedding. However, the generality of the above methods is limited by their manual construction of meta-paths.

Recently, FAME [18] develops a spectral graph transformation component to aggregate information from sub-networks by preserving relation-aware node dependencies. However, this model is built on the random projection and sacrifices the adaptive parameter learning in exchange for fast embedding projection. Furthermore, to learn the node embeddings of multiplex bipartite graph, DualHGCN [36] firstly generates two sets of homogeneous hypergraphs and then perform the information propagation with the spectral hypergraph convolutions. In HDI [14], Jing *et al.* explores the high-order mutual information to construct the supervision signals for enhancing the node representations.

3 PROBLEM DEFINITION

We define graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with the set of nodes $\mathcal{V}$ and edges $\mathcal{E}$. Each edge in $\mathcal{E}$ represents the connections among nodes.

Definition 1 (Attributed Multiplex Heterogeneous Network, or AMHEN). *Given the defined graph $\mathcal{G}$, we further associate all nodes in $\mathcal{V}$ with the attribute feature vectors $\mathbf{X} \in \mathbb{R}^{n \times m}$. Here, the size of node set $\mathcal{V}$ and attribute vector is represented by n and m, respectively. With the consideration of node and edge heterogeneity, we define the node type and edge type mapping function as $\phi : \mathcal{V} \rightarrow \mathcal{O}$ and $\psi : \mathcal{E} \rightarrow \mathcal{R}$. Here, the set of node types and edge types is set with the size of $\mathcal{O}$ and $\mathcal{R}$, respectively. Each node $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$ belong to a certain type in $\mathcal{O}$ and $\mathcal{R}$, respectively. Additionally, with the consideration of edge multiplexity (i.e., $|\mathcal{O}| + |\mathcal{R}| > 2$), the same pair of nodes can be connected through multi-typed edges.*

Definition 2 (Meta-path). *A meta-path $\mathcal{P}$ is defined as a path in the form of $O_1 \xrightarrow{r_1} O_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{l-1}} O_l$ which describes a composite relation $R = r_1 \circ r_2 \cdots r_{l-1}$ between node types O_1 and O_l, where $\circ$ denotes the composition operator on relations.*

For example, $U_1 \xrightarrow{click} I_2 \xrightarrow{buy} U_2$ is a meta-path sample of meta-path $User \xrightarrow{click} Item \xrightarrow{buy} User$. Based on the above definitions, we formally present the representation learning task over the multiplex heterogeneous graph as follows:

Problem (Attributed Multiplex Heterogeneous Graph Representation). *The objective of our representation learning task over the attributed multiplex heterogeneous graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ is to learn low-dimensional latent embedding (with the hidden dimensionality of d $d \ll |\mathcal{V}|$) for each node $v \in \mathcal{V}$, with the preservation of node and edge heterogeneity and multiplexity.*

We summarize the key notations of our technical solution in Table 4 presented in the supplementary material.

4 METHODOLOGY

This section describe our framework MHGCN with the overall architecture shown in Figure 1. Particularly, our MHGCN contains two key learning modules: (i) *multiplex relation aggregation* and (ii) *multilayer graph convolution module*. *Multiplex relation aggregation* aims to aggregate the multi-relations among heterogeneous nodes in multiplex heterogeneous networks by differentiating each relation with importance. *Multilayer graph convolution module* can automatically capture the heterogeneous meta-paths of different lengths across multi-relations by aggregating neighboring nodes' characteristics to learn the low-dimensional representation of nodes.

4.1 Multiplex Relation Aggregation

As defined in Sec. 3, there exit different types of nodes and multiple types of edges between these nodes in AMHENs, and each type of edge has a different role and impact on node representation. Therefore, following [18], we first generate multiple sub-graphs by differentiating the types of edge connections between nodes in the multiplex and heterogeneous graph. Afterwards, we aggregate the relation-aware graph contextual information with different importance weights.

We denote our generated sub-graph as $\{\mathcal{G}_r|r = 1, 2, \ldots, |\mathcal{R}|\}$ with the corresponding adjacent matrix $\{\mathbf{A}_r|r = 1, 2, \ldots, |\mathcal{R}|\}$. Considering the scenario of multiplex user-item relations in online retailer (*e.g.*, click, purchase, review), the decomposed sub-graph corresponds to individual type of relationship between user and item. For instance, for the graph representation learning in E-commerce platforms, different relationships (different edge types) between user and item nodes exhibit various dependency semantics. For example, the diverse behaviors of users (*e.g.*, click, add-to-favorite, purchase) reflect different preferences of users over items. Hence, multiplex user-item interactions with various relation semantics will have different impacts on the learning process of user representations. To capture such multi-typed node dependencies, our proposed MHGCN learns the relation-aware weights β_r to aggregate edge-type-specific sub-graph adjacent matrix as: $\mathbb{A} = \sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r$. Notice that the set of weights $\{\beta_r|r = 1, 2, \ldots, |\mathcal{R}|\}$ should not be a set of hyperparameter, but should be dynamically changed according to different tasks, so we set them as trainable parameters to be learned in model training.

4.2 Multilayer Graph Convolution Module

Different from homogeneous networks, heterogeneous networks contain different types of nodes and edges. The specified types of edges and nodes form a meta-path, which has an obvious effect on the representation learning of heterogeneous networks. Previous works require manually defined meta-paths and learn node representations on the sampled heterogeneous meta-paths. However, setting and sampling meta-paths artificially is a complex task. In a large-scale network, the number of meta-paths is very large. It takes a long time to sample such a large number of meta-paths. At the same time, aggregating meta-paths into meta-path graph also requires a lot of memory overhead. Additionally, the type of meta-paths has an important impact on node representation, which almost determines the performance of network embedding in various downstream tasks. The number of types of heterogeneous meta-paths is also very large, involving different lengths and different relation interactions. Therefore, it is difficult to select the appropriate meta-path types for heterogeneous network embedding methods based on meta-path aggregation. Our MHGCN effectively solves the above problems. We now present our multilayer graph convolution module that automatically captures the the short and long meta-paths across multi-relations in AMHENs.

It is worth noting that our model uses a multi-layer fusion GCN. As shown in Figure 1, our graph convolution module consists of multiple graph convolutional layers. Its purpose is to capture meta-path information of different lengths. Next, we take a two-layer GCN as an example to illustrate how our model capture meta-path information. For a single layer GCN:

$$\mathbf{H}^{(1)} = \mathbb{A} \cdot \mathbf{X} \cdot \mathbf{W}^{(1)}, \tag{1}$$

where $\mathbf{H}^{(1)} \in \mathbb{R}^{n \times d}$ is the output of first layer (*i.e.*, hidden representation of network), $\mathbf{X} \in \mathbb{R}^{n \times m}$ is the node attribute matrix, and $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}$ is the learnable weight matrix. Notice that our convolution adopts the idea of simplifying GCN [32], that is, no non-linear activation function is used.

For the two-layer GCN, the message passing process can be represented as below:

$$\begin{aligned} \mathbf{H}^{(2)} &= \mathbb{A} \cdot \mathbf{H}^{(1)} \cdot \mathbf{W}^{(2)} \\ &= \mathbb{A} \cdot (\mathbb{A} \cdot \mathbf{X} \cdot \mathbf{W}^{(1)}) \cdot \mathbf{W}^{(2)} \\ &= \mathbb{A}^2 \cdot \mathbf{X} \cdot \mathbf{W}^{(1)} \cdot \mathbf{W}^{(2)}, \end{aligned} \tag{2}$$

where $\mathbf{W}^{(2)} \in \mathbb{R}^{d \times d}$ is the learnable weight matrix for second layer.

A toy example of E-commerce network is illustrated in Figure 2, we only consider two relations (*i.e.*, buy and click) between user and item nodes in this case. As shown in Figure 2, aggregated matrix $\mathbb{A}$ can be regarded as a meta-path graph matrix generated by the 1-length meta-paths with importance (*i.e.*, all linked node pairs across all edge types with weights). For example, $\mathbb{A}_{(1,3)} = 1.5$ contains two 1-length meta-path samples with weights, *i.e.*, $U_1 \xrightarrow{1*buy} I_1 : 1$ and $U_1 \xrightarrow{0.5*click} I_1 : 0.5$. Therefore, the single-layer GCN can effectively learn the node representation that contains 1-length meta-path information. Similarly, the second power of $\mathbb{A}$ automatically captures the 2-length meta-path information with importance weights for all node pairs, including original sub-network high-order structures. For example, $\mathbb{A}^2_{(1,1)} = 2.5$ implies five 2-length meta-path samples across multi-relations with importance, *i.e.*, $U_1 \xrightarrow{1*buy} I_1 \xrightarrow{1*buy} U_1 : 1$, $U_1 \xrightarrow{1*buy} I_1 \xrightarrow{0.5*click} U_1 : 0.5$, $U_1 \xrightarrow{0.5*click} I_1 \xrightarrow{0.5*click} U_1 : 0.25$, $U_1 \xrightarrow{0.5*click} I_1 \xrightarrow{1*buy} U_1 : 0.5$, and $U_1 \xrightarrow{0.5*click} I_2 \xrightarrow{0.5*click} U_1 : 0.25$. The sum of the importance of these five meta-path samples is 2.5.

At the same time, considering that the influence of meta-paths with different lengths on embedding should also be different, the learnable weight matrices $\mathbf{W}^{(l)}$ in our multilayer graph convolution module can just play this role. Eventually, we fuse the outputs of

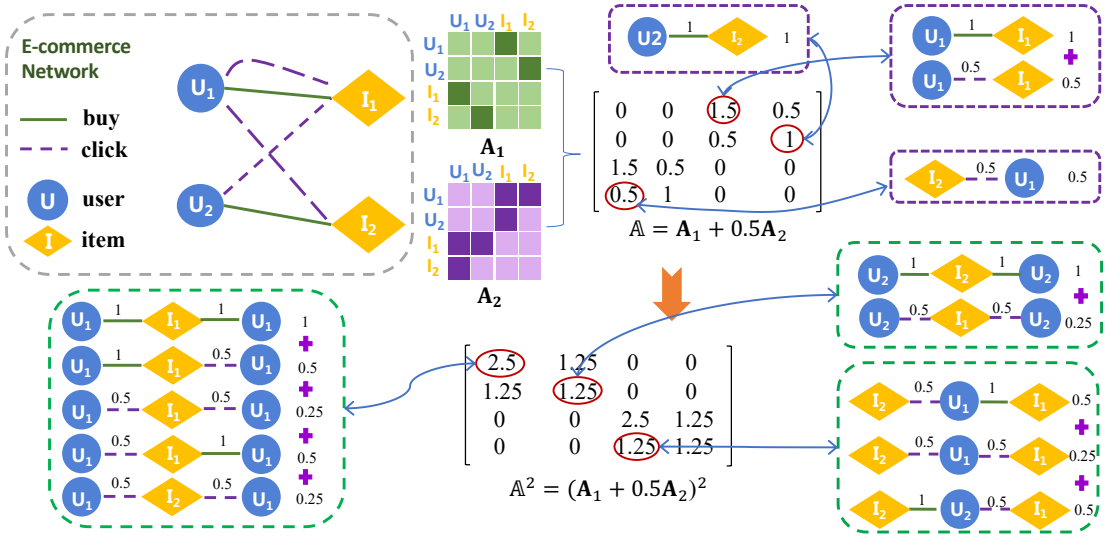


Figure 2: Illustration of meta-paths with importance for a toy example

single-layer GCN and two-layer GCN:

$$\mathbf{H} = \frac{1}{2}(\mathbf{H}^{(1)} + \mathbf{H}^{(2)}). \tag{3}$$

The final embedding $\mathbf{H} \in \mathbb{R}^{n \times d}$ contains all 1-length and 2-length meta-path information.

To capture the more length heterogeneous meta-paths, we can extend it to l-layer:

$$\begin{aligned} \mathbf{H}^{(l)} &= \mathbb{A} \cdot \mathbf{H}^{(l-1)} \cdot \mathbf{W}^{(l)} \\ &= \mathbb{A} \cdot (\mathbb{A} \cdot \mathbf{H}^{(l-2)} \cdot \mathbf{W}^{(l-1)}) \cdot \mathbf{W}^{(l)} \\ &= \underbrace{\mathbb{A} \cdots (\mathbb{A}}_{l} \cdot \mathbf{X} \cdot \underbrace{\mathbf{W}^{(1)}) \cdots \mathbf{W}^{(l)}}_{l} \\ &= \mathbb{A}^l \cdot \mathbf{X} \cdot \underbrace{\mathbf{W}^{(1)} \cdots \mathbf{W}^{(l)}}_{l} \end{aligned} \tag{4}$$

Therefore, our multilayer graph convolution module fuses outputs of all layers to capture all meta-path information of different length across multi-relations:

$$\begin{aligned} \mathbf{H} &= \frac{1}{l}\sum_{i=1}^{l} \mathbf{H}^{(i)} \\ &= \frac{1}{l}\sum_{i=1}^{l} \mathbb{A} \cdot \mathbf{H}^{(i-1)} \cdot \mathbf{W}^{(i)}, \end{aligned} \tag{5}$$

where $\mathbf{H}^{(0)}$ is the node attribute matrix $\mathbf{X}$.

4.3 Model Learning

In this section, we present the objective function to train our model to learn the final node representation. Depending on the requirements of different downstream tasks and the availability of node labels, we can train MHGCN in two major learning paradigms, *i.e.*, unsupervised learning and semi-supervised learning.

For unsupervised learning, we can optimize the model parameters by minimizing the following binary cross-entropy loss function through negative sampling:

$$\mathcal{L} = -\sum_{(u,v)\in\Omega} \log\sigma(< \mathbf{H}_u^\top, \mathbf{H}_v >) - \sum_{(u',v')\in\Omega^-} \log\sigma(- < \mathbf{H}_{u'}^\top, \mathbf{H}_{v'} >), \tag{6}$$

where $\mathbf{H}_v$ is the representation of node v, T denotes matrix transposition, $\sigma(\cdot)$ is the sigmoid function, $<,>$ can be any vector similarity measure function (*e.g.*, inner product), Ω is the set of positive node pairs, Ω^- is the set of negative node pairs sampled from all unobserved node pairs. That is, we use the loss function to increase the similarities between the node representations in the positive samples and decrease the similarities between the node representations in the negative samples simultaneously.

For semi-supervised learning, we can optimize the model parameters by minimizing the cross entropy via backpropagation and gradient descent. The cross entropy loss over all labeled nodes between the ground-truth and the prediction is formulated as:

$$\mathcal{L} = -\sum_{i\in\mathcal{V}_{ids}} \mathrm{Y}_i \ln(\mathbf{C} \cdot \mathbf{H}_i), \tag{7}$$

where $\mathcal{V}_{ids}$ is the set of node indices that have labels, Y_i is the label of the i-th node, $\mathbf{C}$ is the node classifier parameter, and $\mathbf{H}_i$ is the representation of the i-th node. With the guide of a small fraction of labeled nodes, we can optimize the proposed model and then learn the embeddings of nodes for semi-supervised classification.

Notice that $\{\mathbf{W}^{(i)}|i = 1, 2, \ldots, l\}$ and $\{\beta_r|r = 1, 2, \ldots, |\mathcal{R}|\}$ in our model can be learned during training phase. The pseudo-code of our proposed MHGCN is shown in Algorithm 1 in the supplement.

Table 1: Statistics of Datasets (n-type: node type, e-type: edge type, feat.: features, and Mult.: Multiplex network)

Dataset	#nodes	#edges	#n-type	#e-type	#feat.	Mult.
Alibaba	21,318	41,676	2	4	19	✓
Amazon	10,166	148,865	1	2	1,156	✓
AMiner	58,068	118,939	3	3	4	×
IMDB	12,772	18,644	3	2	1,256	×
DBLP	26,128	119,783	4	3	4,635	×

5 EXPERIMENT

5.1 Datasets

Five publicly available real-world datasets are used in experimental evaluation, *i.e.*, Alibaba[1], Amazon[2], AMiner[3], IMDB[4], and DBLP[5]. Detailed dataset description can be found in the supplement. Since some of the baselines cannot scale to the whole Alibaba network, we evaluate all models on a sampled dataset from Alibaba. The statistics of these five datasets are summarized in Table 1.

5.2 Baselines

We compare our MHGCN against the following eighteen graph learning baselines, which are divided into three categories.

Homogeneous network embedding methods:

- **node2vec** [6] - node2vec is a network embedding method which samples short biased random walks.
- **RandNE** [41] - RandNE is a network embedding approach based on Gaussian random projection, which preserves the high-order proximities between nodes.
- **FastRP** [2] - FastRP is an extension of RandNE by using sparse random projection.
- **SGC** [32] - SGC is a simplified version of GCN, which only uses the product of high-order adjacency matrices and attribute matrix, without nonlinear transformation.
- **AM-GCN** [30] - AM-GNN is a state-of-the-art graph convolutional network, which is an adaptive multi-channel graph convolutional networks for semi-supervised classification.

Heterogeneous network embedding methods:

- **R-GCN** [24] - R-GCN further considers the influence of different edge types on nodes, and uses weight sharing and coefficient constraints to apply to heterogeneous networks.
- **HAN** [28] - HAN applies graph attention network on multiplex network considering the inter- and intra-network interactions, which exploit manually selected meta-paths to learn node embedding.
- **NARS** [37] NARS decouples heterogeneous networks according to the type of edge, and then aggregates neighbor features on the decoupled subgraph.
- **MAGNN** [5] - MAGNN is a metapath aggregated graph neural network for heterogeneous graphs.
- **HPN** [13] - HPN designs a semantic propagation mechanism to alleviate semantic confusion and a semantic fusion mechanism to integrate rich semantics.

Multiplex Heterogeneous network embedding methods:

- **PMNE** [16] - PMNE contains three different models to merge the multiplex network to generate one overall embedding for each node, which are denoted as PMNE-n, PMNE-r, and PMNE-c, respectively.
- **MNE** [40] - MNE obtains the final embedding by combining the high-dimensional common embedding and the low-dimensional hierarchical embedding.
- **GATNE** [1] - GATNE includes two variants GATNE-T and GATNE-I. We use GATNE-I as our baseline method in experiments.
- **GTN** [38] - GTN transforms a heterogeneous graph into multiple meta-path graphs and then learns node embeddings via GCN on the meta-path graphs.
- **DMGI** [21] - DMGI integrates node embeddings from multiple graphs by introducing a consensus regularization framework and an universal discriminator.
- **FAME** [18] - FAME is a random projection-based network embedding for AMHENs, which uses spectral graph transformation to capture meta-paths, and significantly improves efficiency through random projection.
- **HGSL** [42] - HGSL is a state-of-the-art heterogeneous GNN, which jointly performs heterogeneous graph structure learning and GNN parameter learning for classification.
- **DualHGNN** [36] - DualHGCN uses dual hypergraph convolutional network to learn node embeddings for multiplex bipartite networks.

The network types handled by the competitor methods are summarized in Table 5 in the supplemental material.

5.3 Experimental Setting

Following [18], we set d to 200 for all the methods. For baselines, we use the source code released by their authors or OpenHGNN[6], and adopt the parameter settings recommended in their papers and fine-tune them to be optimal. For our MHGCN, we set the number of convolution layers l to 2. For fair comparison, we uniformly set the number of training rounds to 500 for link prediction and the number of training rounds to 200 for node classification. More detailed experimental settings can be found in the supplement.

5.4 Link Prediction

We first evaluate the model performance by comparing our MHGCN with fifteen baselines on link prediction task in an unsupervised learning manner. The results are shown in Table 2, where the best is shown in bold. The first seven baselines are homogeneous or heterogeneous network embedding methods, and the last eight are multiplex network embedding methods.

We can see that MHGCN significantly outperforms all baselines in terms of all evaluation metrics on five datasets. Specifically, MHGCN achieves average gains of 5.68% F1 score in comparison to the best performed GNN baselines across all datasets (*i.e.*, FAME,

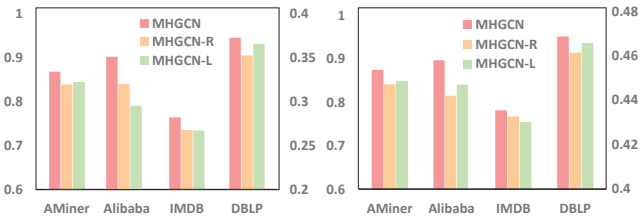
[1]https://tianchi.aliyun.com/competition/entrance/231719/information/
[2]http://jmcauley.ucsd.edu/data/amazon/
[3]https://github.com/librahu/
[4]https://github.com/seongjunyun/Graph_Transformer_Networks
[5]https://www.dropbox.com/s/yh4grpeks87ugr2/DBLP_processed.zip?dl=0
[6]https://github.com/BUPT-GAMMA/OpenHGNN

Table 2: Link prediction performance comparison of different methods on five datasets

Method	AMiner			Alibaba			IMDB			Amazon			DBLP		
	R-AUC	PR-AUC	F1	R-AUC	PR-AUC	F1	R-AUC	PR-AUC	F1	R-AUC	PR-AUC	F1	R-AUC	PR-AUC	F1
node2vec	0.594	0.663	0.602	0.614	0.580	0.593	0.479	0.568	0.474	0.946	0.944	0.880	0.449	0.452	0.478
RandNE	0.607	0.630	0.608	0.877	0.888	0.826	0.901	0.933	0.839	0.950	0.941	0.903	0.492	0.491	0.493
FastRP	0.620	0.634	0.600	0.927	0.900	0.926	0.869	0.893	0.811	0.954	0.945	0.893	0.515	0.528	0.506
SGC	0.589	0.585	0.567	0.686	0.708	0.623	0.826	0.889	0.769	0.791	0.802	0.760	0.601	0.606	0.587
R-GCN	0.599	0.601	0.610	0.674	0.710	0.629	0.826	0.878	0.790	0.811	0.820	0.783	0.589	0.592	0.566
MAGNN	0.663	0.681	0.666	0.961	0.963	0.948	0.912	0.923	0.887	0.958	0.949	0.915	0.690	0.699	0.684
HPN	0.658	0.664	0.660	0.958	0.961	0.950	0.900	0.903	0.892	0.949	0.949	0.904	0.692	0.710	0.687
PMNE-n	0.651	0.669	0.677	0.966	0.973	0.891	0.674	0.683	0.646	0.956	0.945	0.893	0.672	0.679	0.663
PMNE-r	0.615	0.653	0.662	0.859	0.915	0.824	0.646	0.646	0.613	0.884	0.890	0.796	0.637	0.640	0.629
PMNE-c	0.613	0.635	0.657	0.597	0.591	0.664	0.651	0.634	0.630	0.934	0.934	0.868	0.622	0.625	0.609
MNE	0.660	0.672	0.681	0.944	0.946	0.901	0.688	0.701	0.681	0.941	0.943	0.912	0.657	0.660	0.635
GATNE	OOT	OOT	OOT	0.981	0.986	0.952	0.872	0.878	0.791	0.963	0.948	0.914	OOT	OOT	OOT
DMGI	OOM	OOM	OOM	0.857	0.781	0.784	0.926	0.935	0.873	0.905	0.878	0.847	0.610	0.615	0.601
FAME	0.687	0.747	0.726	0.993	0.996	0.979	0.944	0.959	0.897	0.959	0.950	0.900	0.642	0.650	0.633
DualHGNN	/	/	/	0.974	0.977	0.966	/	/	/	/	/	/	/	/	/
MHGCN	**0.711**	**0.753**	**0.730**	**0.997**	**0.997**	**0.992**	**0.967**	**0.966**	**0.959**	**0.972**	**0.974**	**0.961**	**0.718**	**0.722**	**0.703**

OOT: Out Of Time (36 hours). OOM: Out Of Memory; DMGI runs out of memory on the entire AMiner. R-AUC: ROC-AUC.

MAGNN and HPN). Our MHGCN realizes a high accuracy of more than 96% on three datasets (Alibaba, Amazon, and IMDB), especially more than 99% prediction performance on Alibaba network. This is because MHGCN automatically captures effective multi-relational topological structures through multiplex relation aggregation and multilayer graph convolution on the generated meta-paths across multiplex relations. Especially, compared with GATNE and MAGNN, our model has achieved better results, showing the ability of our model in automatically capturing meta-paths compared with manually setting meta-paths. FAME that use spectral graph transformation achieving the second best performance on most datasets also verifies the ability of multiplex relation aggregation to automatically capture useful heterogeneous meta-paths. However, MHGCN obtains better performance than FAME on all networks as MHGCN learns meaning node representations for AMHENs using multilayer graph convolution in a learning manner. Additionally, MHGCN also shows significant performance advantages on general heterogeneous networks (*e.g.*, IMDB and DBLP). This may be because our MHGCN uses a weighted approach to differentiate the effects of different types of relations on node representation, which cannot be achieved by traditional meta-path sampling.



(a) Macro-F1 score (b) Micro-F1 score

Figure 3: Experimental results of ablation study

5.5 Node Classification

We next evaluate the effectiveness of our model on the node classification task compared with state-of-the-art methods. The results are shown in Table 3, where the best is shown in bold. The first eight baselines are unsupervised embedding methods, and the rest are semi-supervised embedding methods.

As we see, MHGCN also achieves state-of-the-art performance on all tested networks. Specifically, our MHGCN achieves average 11.22% and 14.49% improvement over state-of-the-art GNN model HGSL across all datasets in terms of Macro-F1 and Micro-F1, respectively. Considering that the performance gain in node classification task reported in some recent works [5, 42] is usually around 2-4%, this performance improvement achieved by our MHGCN is significant. Furthermore, we also observe that MHGCN performs much better than competitor methods on general heterogeneous network with multi-typed nodes (*e.g.*, IMDB and AMiner), achieving 23.23% and 22.19% improvement in Macro-F1 and Micro-F1 on IMDB network. The possible reason is that our MHGCN effectively learns node representations for classification by exploring all meta-path interactions across multiple relations with different importance (*i.e.*, weights), which is ignored by the heterogeneous network embedding approaches based on manually setting meta-path sampling.

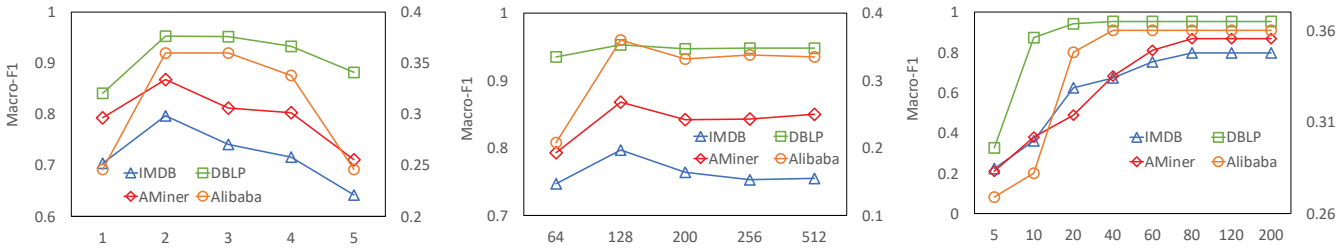
5.6 Ablation Study

To validate the effectiveness of each component of our model, we further conduct experiments on different MHGCN variations. Here MHGCN-R does not consider the importance of different relations, that is, we set the weights β_r to 1; MHGCN-L uses only a two-layer GCN to obtain the embedding, so it can only capture the 2-length meta-paths. We report the results of ablation study on four datasets for node classification in Figure 3, where the performance on Alibaba refers to the right-ordinate axis.

Table 3: Node classification performance comparison of different methods on four datasets

Method	AMiner		Alibaba		IMDB		DBLP	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
node2vec	0.522 (0.0032)	0.532 (0.0051)	0.238 (0.0125)	0.347 (0.0093)	0.363 (0.0237)	0.382 (0.0703)	0.352 (0.0103)	0.351 (0.0112)
RandNE	0.641 (0.0074)	0.672 (0.0064)	0.319 (0.0170)	0.358 (0.0093)	0.373 (0.0143)	0.392 (0.0185)	0.351 (0.0153)	0.372 (0.0150)
FastRP	0.650 (0.0086)	0.690 (0.0074)	0.301 (0.0180)	0.392 (0.0119)	0.363 (0.0236)	0.381 (0.0140)	0.343 (0.0201)	0.375 (0.0199)
MNE	0.643 (0.0069)	0.686 (0.0045)	0.289 (0.0155)	0.390 (0.0021)	0.374 (0.0153)	0.382 (0.0680)	0.366 (0.0117)	0.384 (0.0109)
GATNE	OOT	OOT	0.291 (0.0086)	0.390 (0.0014)	0.369 (0.0132)	0.333 (0.0005)	OOT	OOT
DMGI	0.473 (0.0155)	0.626 (0.0093)	0.220 (0.0214)	0.392 (0.0026)	0.548 (0.0190)	0.544 (0.0189)	0.781 (0.0303)	0.787 (0.0235)
FAME	0.722 (0.0114)	0.727 (0.0091)	0.323 (0.0154)	0.393 (0.0060)	0.593 (0.0135)	0.594 (0.0143)	0.842 (0.0183)	0.868 (0.0127)
DualHGNN	/	/	0.347 (0.0114)	0.402 (0.0127)	/	/	/	/
SGC	0.516 (0.0047)	0.587 (0.0157)	0.286 (0.0231)	0.361 (0.0175)	0.489 (0.0106)	0.563 (0.0133)	0.622 (0.0009)	0.623 (0.0009)
AM-GCN	0.702 (0.0175)	0.713 (0.0223)	0.307 (0.0232)	0.399 (0.0156)	0.610 (0.0021)	0.640 (0.0013)	0.867 (0.0105)	0.878 (0.0112)
R-GCN	0.690 (0.0078)	0.692 (0.0106)	0.265 (0.0326)	0.381 (0.0125)	0.544 (0.0172)	0.572 (0.0145)	0.862 (0.0053)	0.870 (0.0070)
HAN	0.690 (0.0149)	0.726 (0.0086)	0.275 (0.0327)	0.392 (0.0081)	0.552 (0.0112)	0.568 (0.0078)	0.806 (0.0078)	0.813 (0.0100)
NARS	0.722 (0.0103)	0.721 (0.0097)	0.297 (0.0201)	0.392 (0.0195)	0.565 (0.0037)	0.574 (0.0048)	0.794 (0.0255)	0.804 (0.0320)
MAGNN	0.755 (0.0105)	0.757 (0.0133)	0.348 (0.0488)	0.398 (0.0405)	0.614 (0.0073)	0.615 (0.0089)	0.881 (0.0284)	0.895 (0.0396)
HPN	0.710 (0.0612)	0.732 (0.0490)	0.263 (0.0346)	0.392 (0.0405)	0.578 (0.0023)	0.584 (0.0021)	0.822 (0.0201)	0.830 (0.0201)
GTN	OOM	OOM	0.255 (0.0420)	0.392 (0.0071)	0.615 (0.0108)	0.616 (0.0093)	0.852 (0.0137)	0.868 (0.0125)
HGSL	0.754 (0.0100)	0.758 (0.0103)	0.338 (0.0121)	0.398 (0.0238)	0.620 (0.0048)	0.638 (0.0030)	0.893 (0.0284)	0.902 (0.0396)
MHGCN	**0.868 (0.0160)**	**0.875 (0.0200)**	**0.351 (0.0204)**	**0.458 (0.0160)**	**0.764 (0.0145)**	**0.782 (0.0138)**	**0.945 (0.0221)**	**0.952 (0.0203)**

OOT: Out Of Time (36 hours), OOM: Out Of Memory. The standard deviations are reported in the parentheses.



(a) Macro-F1 score *w.r.t.* #layers (b) Macro-F1 score *w.r.t.* dimension d (c) Macro-F1 score *w.r.t.* #rounds

Figure 4: Parameter sensitivity of proposed method *w.r.t.* #layers, dimension d, and #rounds.

It can be seen from the results that the two key components both contribute to performance improvement of our MHGCN. The comparison between MHGCN-R and MHGCN highlights the effectiveness of the importance of different relations. We can observe that MHGCN-R performs worse than MHGCN on all datasets in terms of both Macro-F1 and Micro-F1 metrics, reducing 9.68% performance in Macro-F1 score on Alibaba, which demonstrates the crucial role of our designed multiplex relation aggregation module in capturing the importance of different relations for node representation learning. The comparison between MHGCN-L and MHGCN reflects the importance of our multilayer graph convolution module. Compared with MHGCN-L, MHGCN improves 2.97%, 18.98%, 4.09% and 1.51% over MHGCN-L in terms of Macro-F1 on AMiner, Alibaba, IMDB, and DBLP, respectively. This indicates that our proposed multilayer graph convolution module effectively captures useful meta-paths of different lengths across multiplex relations.

5.7 Parameter Sensitivity

We finally investigate the sensitivity of MHGCN with respect to the important parameters, including the number of layers l, embedding dimension d, and the number of training rounds. We report Macro-F1 score on node classification task with different parameter settings on four datasets in Figure 4. Notice that the performance on Alibaba refers to the ordinate on the right.

As shown in Figure 4(a), at first, the performance of MHGCN increases as l increases, and then the performance begins to decline when $l \geq 2$. This is mainly because 1-length and 2-length meta-path interactions already effectively capture the topological structures of network for node classification, while longer meta-paths would not lead to performance improvement. With the growth of GCN layers, the representation of nodes would be flattened after multiple convolutions, resulting in performance degradation. From the results in Figure 4(b), we can see that the performance of MHGCN gradually rises and then decreases slightly as dimension d increases, and achieves the best performance when embedding dimension $d = 128$. This is because the features of all nodes are compressed into a small

embedding space when dimension d is small, thus it is difficult to retain the characteristics proximities of all node pairs. Conversely, a larger dimension would also flatten the distance between all node embeddings. Figure 4(c) illustrates the performance of our MHGCN with respect to the number of training rounds in learning model weights. We can find that our MHGCN can converge quickly and efficiently achieve stable performance within 80 rounds on all tested datasets.

6 CONCLUSION

In this paper, we propose an embedding model MHGCN for attributed multiplex heterogeneous networks. Our model mainly includes two key components: multiplex relation aggregation and multilayer graph convolution module. Through multiplex relation aggregation, MHGCN can distinguish the importance of the relations between different nodes in multiplex heterogeneous networks. Through multilayer graph convolution module, MHGCN can automatically capture the short and long meta-path interactions across multi-relations, and learn meaning node embeddings with model parameter learning during training phase. Experiments results on five real-world heterogeneous networks show the superiority of the proposed MHGCN in both link prediction and node classification.

ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foundation of China under grant Nos. 62176243, 62072288, 61773331 and 41927805, and the National Key Research and Development Program of China under grant Nos. 2018AAA0100602 and 2019YFC1509100.

REFERENCES

[1] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *KDD*. 1358–1368.
[2] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. 2019. Fast and Accurate Network Embeddings via Very Sparse Random Projection. In *CIKM*. 399–408.
[3] Hongxu Chen, Hongzhi Yin, Weiqing Wang, Hao Wang, Quoc Viet Hung Nguyen, and Xue Li. 2018. PME: projected metric embedding on heterogeneous networks for link prediction. In *KDD*. 1177–1186.
[4] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*. 135–144.
[5] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW*. 2331–2341.
[6] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.
[7] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1025–1035.
[8] Li He, Hongxu Chen, Dingxian Wang, Shoaib Jameel, Philip Yu, and Guandong Xu. 2021. Click-Through Rate Prediction with Multi-Modal Hypergraphs. In *CIKM*. 690–699.
[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.
[10] Binbin Hu, Yuan Fang, and Chuan Shi. 2019. Adversarial Learning on Heterogeneous Information Networks. In *KDD*. 120–129.
[11] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*. 2704–2710.
[12] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xiao, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In *AAAI*.
[13] Houye Ji, Xiao Wang, Chuan Shi, Bai Wang, and Philip Yu. 2021. Heterogeneous Graph Propagation Network. *TKDE* (2021).
[14] Baoyu Jing, Chanyoung Park, and Hanghang Tong. 2021. Hdmi: High-order deep multiplex infomax. In *The Web Conference*. 2414–2424.
[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[16] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. 2017. Principled multilayer network embedding. In *ICDMW*. IEEE, 134–141.
[17] Zhijun Liu, Chao Huang, Yanwei Yu, and Junyu Dong. 2021. Motif-preserving dynamic attributed network embedding. In *WWW*. 1629–1638.
[18] Zhijun Liu, Chao Huang, Yanwei Yu, Baode Fan, and Junyu Dong. 2020. Fast Attributed Multiplex Heterogeneous Network Embedding. In *CIKM*. 995–1004.
[19] Xiaoling Long, Chao Huang, Yong Xu, Huance Xu, Peng Dai, Lianghao Xia, and Liefeng Bo. 2021. Social Recommendation with Self-Supervised Metagraph Informax Network. In *CIKM*. 1160–1169.
[20] Yuanfu Lu, Chuan Shi, Linmei Hu, and Zhiyuan Liu. 2019. Relation structure-aware heterogeneous information network embedding. In *AAAI*. 4456–4463.
[21] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. 2020. Unsupervised Attributed Multiplex Network Embedding. In *AAAI*. 5371–5378.
[22] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD*. 701–710.
[23] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. Netsmf: Large-scale network embedding as sparse matrix factorization. In *WWW*. 1509–1520.
[24] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.
[25] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *TKDE* 31, 2 (2018), 357–370.
[26] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.
[27] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[28] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. ACM, 2022–2032.
[29] Xiao Wang, Yuanfu Lu, Chuan Shi, Ruijia Wang, Peng Cui, and Shuai Mou. 2020. Dynamic heterogeneous information network embedding with meta-path based proximity. *TKDE* (2020).
[30] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In *KDD*. 1243–1253.
[31] Wei Wei, Chao Huang, Lianghao Xia, Yong Xu, Jiashu Zhao, and Dawei Yin. 2022. Contrastive meta learning with behavior multiplicity for recommendation. In *KDD*. 1120–1128.
[32] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, et al. 2019. Simplifying Graph Convolutional Networks. In *ICML*. 6861–6871.
[33] Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. 2020. Graph convolutional networks with markov random field reasoning for social spammer detection. In *AAAI*, Vol. 34. 1054–1061.
[34] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Bo Zhang, and Liefeng Bo. 2020. Multiplex Behavioral Relation Learning for Recommendation via Memory Augmented Transformer Network. In *SIGIR*. 2397–2406.
[35] Lianghao Xia, Yong Xu, Chao Huang, Peng Dai, and Liefeng Bo. 2021. Graph meta network for multi-behavior recommendation. In *SIGIR*. 757–766.
[36] Hansheng Xue, Luwei Yang, Vaibhav Rajan, Wen Jiang, Yi Wei, and Yu Lin. 2021. Multiplex bipartite network embedding using dual hypergraph convolutional networks. In *WWW*. 1649–1660.
[37] Lingfan Yu, Jiajun Shen, Jinyang Li, and Adam Lerer. 2020. Scalable graph neural networks for heterogeneous graphs. *arXiv preprint arXiv:2011.09679* (2020).
[38] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph Transformer Networks. In *NeurIPS*. 11960–11970.
[39] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.
[40] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable multiplex network embedding. In *IJCAI*, Vol. 18. 3082–3088.
[41] Ziwei Zhang, Peng Cui, Haoyang Li, Xiao Wang, and Wenwu Zhu. 2018. Billion-Scale Network Embedding with Iterative Random Projection. In *ICDM*. 787–796.
[42] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. 2021. Heterogeneous Graph Structure Learning for Graph Neural Networks. In *AAAI*.

A SUPPLEMENT

A.1 Notations

Key notations used in the paper and their definitions are summarized in Table 4.

Table 4: Main notations and their definitions.

Notation	Definition
$\mathcal{G}$	the input network
$\mathcal{V}, \mathcal{E}$	the node/edge set of $\mathcal{G}$
$\mathcal{O}, \mathcal{R}$	the node/edge type set of $\mathcal{G}$
$\mathbf{X}$	the node attribute matrix of $\mathcal{G}$
$\mathcal{G}_r$	the sub-network *w.r.t.* edge type r
$\mathbf{A}_r$	the adjacency matrix of $\mathcal{G}_r$
$\mathbb{A}$	the aggregated adjacency matrix
$\mathbf{H}$	the node embeddings
$\mathbf{H}^{(l)}$	the hidden representation for the l-th layer
d	the dimension of embeddings
n, m	the number of nodes/attributes
β_r	the learnable weight for edge type r
$\mathbf{W}^{(l)}$	the learnable weight matrix for the l-th layer

A.2 Algorithm Pseudo-Code

Algorithm 1 shows the pseudo-code of our proposed MHGCN framework guided by the above objective functions (*i.e.*, Eq. (6) or Eq. (7)).

Algorithm 1 The Learning Process of MHGCN

Input: Input AMHEN $\mathcal{G}$, node feature matrix $\mathbf{X}$, embedding dimension d, the number of convolution layers l
Output: Embedding results $\mathbf{H}$

1: Decouple the attributed multiplex heterogeneous network into homogeneous networks and bipartite networks to obtain the adjacency matrices $\{\mathbf{A}_r | r = 1, 2, \ldots, |\mathcal{R}|\}$
2: Calculate $\mathbb{A} = \sum_{r=1}^{|\mathcal{R}|} \beta_r \mathbf{A}_r$
3: **for** $i = 1$ to l **do**
4: Calculate $\mathbf{H}^{(i)} \leftarrow \mathbb{A} \cdot \mathbf{H}^{(i-1)} \cdot \mathbf{W}^{(i)}$
5: **end for**
6: $\mathbf{H} = \frac{1}{l}(\mathbf{H}^{(1)} + \cdots + \mathbf{H}^{(l)})$
7: Calculate $\mathcal{L}$ using Eq. (6) or Eq. (7);
8: Back propagation and update parameters in MHGCN
9: Return $\mathbf{H}$

A.3 Detailed Dataset Description

Alibaba dataset includes four types of edges between user and item nodes. We use the category of item as the class label in node classification. Amazon dataset includes one node type of products in Electronics category, and co-viewing and co-purchasing links between products. The product attributes contain the price, sales-rank, brand, category, etc. AMiner dataset is a citation network, which contains three types of nodes: author, paper and conference. The domain of papers is considered as the class label. IMDB dataset contains three types of nodes, *i.e.*, movie, actor and director, and labels are genres of movies. Node features are given as bag-of-words representations of plots. DBLP dataset contains four types of nodes, *i.e.*, author, paper, term and venue. We use the authors' research field as a label for classification.

A.4 Detailed Experimental Settings

For link prediction task, we treat the connected nodes in network as positive node pairs, and consider all unlinked nodes as negative node pairs. For each edge type, we divide the positive node pairs into training set, verification set and test set according to the proportion of 85%, 5% and 10%. At the same time, we randomly select the same number of negative node pairs to add into training set, validation set and test set. Notice that we predict each type of edge using all types of edges in datasets, and finally take the average of all edges as the final result. For node classification task, we first learn the representation of each node and then perform accuracy evaluation. More specifically, we take 80% of the node embeddings as the training set, 10% as the validation set, and 10% as the test set. In experiments, we train a logistic regression classifier for node classification. Notice that we repeat each experiment 10 times to report average results.

For fair comparison, we uniformly set the number of training rounds to 500 for link prediction and the number of training rounds to 200 for node classification. We set $p = 2$ and $q = 0.5$ for node2vec and set α_r and β_r to 1 for every edge type r on GATNE. For the PMNE model, we use the hyperparameters given by the original paper. For the MNE, we set the dimension of additional vectors to 10, set the length of walk as 10, set the number of walks as 20. For all deep learning methods (*e.g.*, GATNE, HAN, GTN), we tune learning rate in $\{0.01, 0.05, 0.001, 0.005, 0.0001, 0.0005\}$. We set the regularization parameter to 0.001, the number of attention head is set as 8, and the dropout ratio of attention is 0.6. For GTN, we use the sparse version of their released source code and set GT layers to 3 for all datasets. For DMGI, we set the self-connection weight $w = 3$ and tune α, β, γ in $\{0.0001, 0.001, 0.01, 0.1\}$. For FAME, we perform optuna[7] to tune the weights $\alpha_1, \ldots, \alpha_K, \beta_1, \ldots, \beta_{|\mathcal{R}|}$ as described in the original paper. For AM-GCN, we tune loss aggregation parameters β, γ in $\{0.0001, 0.001, 0.01, 0.1\}$. For MAGNN, we set the number of independent attention mechanisms $k = 4$. For HPN, we set iterations in semantic propagation $k = 3$ and value of restart probability $\alpha = 0.1$. For HGSL, we set the number of GNN layers to 2 and the hidden layer output dimension to 64. For R-GCN, we set the batch size to 126, the number of GNN layers to 2, and the hidden layer dimension to 64. For NARS, we set the number of hops to 2, and the number of feed-forward layers to 2. For DualHGNN, we use the asymmetric operator and set λ as 0.5.

For our MHGCN, we set the number of convolution layers l to 2, learning rate to 0.05, dropout to 0.5, and weight-decay to 0.0005.

We evaluate the efficiency evaluation for all methods on a machine with Intel Xeon E5-2660 (2.2GHz) CPU, 80GB memory, and 2 × GeForce RTX 2080 (8G).

The source code of our model is available at https://github.com/NSSSJSS/MHGCN.

[7] https://github.com/pfnet/optuna

Table 5: The network types handled by different methods (Heter.: Heterogeneity, Multi.: Multiplex edge type, Attr.: Attribute, Unsup.: Unsupervised, Auto.: Automatic meta-path).

Method	Heter.		Multi.	Attr.	Unsup.	Auto.
	Node	Edge				
node2vec	×	×	×	×	✓	×
RandNE	×	×	×	×	✓	×
FastRP	×	×	×	×	✓	×
SGC	×	×	×	✓	✓/×	×
AM-GCN	×	×	×	✓	×	×
R-GCN	✓	✓	×	✓	✓/×	×
HAN	✓	✓	×	✓	×	×
NARS	✓	✓	×	✓	×	×
MAGNN	✓	✓	×	✓	✓/×	×
HPN	✓	✓	×	✓	✓/×	×
PMNE	×	✓	✓	×	✓	×
MNE	×	✓	✓	×	✓	×
GATNE	✓	✓	✓	✓	✓	×
GTN	✓	✓	✓	✓	×	✓
DMGI	✓	✓	✓	✓	✓	×
FAME	✓	✓	✓	✓	✓	✓
HGSL	✓	✓	✓	✓	×	×
DualHGNN	✓	×	✓	✓	✓	×
MHGCN	✓	✓	✓	✓	✓/×	✓

A.5 Baselines

The publicly source codes of baselines can be available at the following URLs:

- **node2vec** – https://github.com/aditya-grover/node2vec
- **RandNE** – https://github.com/ZW-ZHANG/RandNE
- **FastRP** – https://github.com/GTmac/FastRP
- **SGC** – https://github.com/Tiiiger/SGC
- **AM-GCN** – https://github.com/zhumeiqiBUPT/AM-GCN
- **R-GCN** – https://github.com/BUPT-GAMMA/OpenHGNN
- **HAN** – https://github.com/Jhy1993/HAN
- **NARS** – https://github.com/BUPT-GAMMA/OpenHGNN
- **MAGNN** – https://github.com/cynricfu/MAGNN
- **HPN** – https://github.com/BUPT-GAMMA/OpenHGNN
- **PMNE** – The source code of PMNE used in this work is released by the authors of MNE at https://github.com/HKUST-KnowComp/MNE
- **MNE** – https://github.com/HKUST-KnowComp/MNE
- **GATNE** – https://github.com/THUDM/GATNE
- **GTN** – https://github.com/seongjunyun/Graph_Transformer_Networks
- **DMGI** – https://github.com/pcy1302/DMGI
- **FAME** – https://github.com/ZhijunLiu95/FAME
- **HGSL** – https://github.com/Andy-Border/HGSL
- **DualHGNN** – https://github.com/xuehansheng/DualHGCN

For homogeneous network embedding methods and heterogeneous network embedding methods to deal with multiplex networks, we feed separate graphs with a single-layer view into them to obtain different node embeddings, then perform mean pooling to generate final node embedding. Since DualHGNN is designed only for multiplex bipartite networks, it can only work on Alibaba network.

The network types handled by the baseline methods are summarized in Table 5.

A.6 Additional Experimental Results

A.6.1 Model Efficiency Analysis. We also compare the efficiency of our MHGCN with other GNN baselines for semi-supervised node classification. We report the experimental results on four datasets in Table 6.

As can be seen from Table 6, our MHGCN achieves the fourth-best performance after three heterogeneous network embedding methods (*i.e.*, R-GCN, NARS and HPN). However, from the above experimental results (Tables 2 and 3), MHGCN is significantly better than these three methods in both link prediction and node classification. MHGCN is significantly faster than the best performed GNN baseline in node classification task (*i.e.*, HGSL) on all datasets under the same number of training rounds. More specifically, our MHGCN achieves up to 135× speedup over state-of-the-art embedding method HAN. MHGCN is faster than state-of-the-art AMHEN embedding method GTN by 21.25 times on multiplex Alibaba network. MHGCN is even 2.33 times and 16.58 times faster than state-of-the-art heterogeneous GNN model MAGNN on Alibaba and AMiner, respectively. The main reason is because our MHGCN adopts the idea of simplifying graph convolutional networks, that is, omitting non-linear activation function. Therefore, the training efficiency of MHGCN can be significantly improved. In fact, according to the above experimental results in Figure 4(c), our model can converge quickly within 80 rounds for node classification on four tested datasets, that is, our model does not need to be trained for 200 rounds set in our experimental evaluation and thus can achieve faster efficiency.

Table 6: Runtime comparison of GNN methods (Second)

Method	AMiner	Alibaba	IMDB	DBLP
AM-GCN	8703.71	2519.82	24280.12	2786.73
R-GCN	**153.04**	301.25	155.40	192.85
HAN	87105.55	4226.95	70510	22315.36
NARS	172.21	**211.54**	**75.81**	**108.54**
MAGNN	10361.20	2320.62	731.03	2125.33
HPN	172.82	249.47	176.64	109.49
GTN	OOM	21166.83	4287.20	18233.64
HGSL	1684.03	2120.93	1758.21	2037.10
DualHGN	/	11295.92	/	/
MHGCN	645.20	996.52	677.23	970.29
Speedup*	**135.05**×	**4.37**×	**104.15**×	**23.01**×
Speedup**	/	**21.25**×	**6.33**×	**18.80**×

* Speedup of MHGCN over HAN.
** Speedup of MHGCN over GTN.
OOM: Out Of Memory.