This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

Heterogeneous Hypergraph Variational Autoencoder for Link Prediction

Haoyi Fan, Fengbin Zhang, Yuxuan Wei, Zuoyong Li, Changqing Zou, Yue Gao, *Senior Member*, *IEEE*, Qionghai Dai, *Senior Member*, *IEEE*

Abstract—Link prediction aims at inferring missing links or predicting future ones based on the currently observed network. This topic is important for many applications such as social media, bioinformatics and recommendation systems. Most existing methods focus on homogeneous settings and consider only low-order pairwise relations while ignoring either the heterogeneity or high-order complex relations among different types of nodes, which tends to lead to a sub-optimal embedding result. This paper presents a method named Heterogeneous Hypergraph Variational Autoencoder (HeteHG-VAE) for link prediction in heterogeneous information networks (HINs). It first maps a conventional HIN to a heterogeneous hypergraph with a certain kind of semantics to capture both the high-order semantics and complex relations among nodes, while preserving the low-order pairwise topology information of the original HIN. Then, deep latent representations of nodes and hyperedges are learned by a Bayesian deep generative framework from the heterogeneous hypergraph in an unsupervised manner. Moreover, a hyperedge attention module is designed to learn the importance of different types of nodes in each hyperedge. The major merit of HeteHG-VAE lies in its ability of modeling multi-level relations in heterogeneous settings. Extensive experiments on real-world datasets demonstrate the effectiveness and efficiency of the proposed method.

Index Terms—Heterogeneous information network, hypergraph, hyperedge attention, link prediction, variational inference

1 Introduction

Link prediction, as a fundamental task in network analysis, aims to predict whether two nodes in a network are likely to have a link which is ubiquitous in a multitude of application domains, such as social relation finding [1, 2], knowledge graph completion [3, 4], protein interaction prediction [5, 6], and product recommendation [7, 8]. Therefore, link prediction approaches have been extensively studied in past decades because of their numerous applications in various network related domains.

Existing link prediction methods focus on either the plain topological networks [9, 10, 11, 12, 13] where only topological information is available, or auxiliary information enhanced networks [14, 15, 16, 17], in which extra attribute information are utilized in the form of node attributes in order to learn a low-dimensional latent representation of the network. Those methods are mainly designed for homogeneous information networks which assume that all nodes and edges in the network are of the same kind. However, the relations and attributes of different types of nodes do not necessarily share a common representation, yet they are semantically related. Therefore, as a powerful information representation, a heterogeneous information network (HIN), consisting of multiple types of nodes and edges, has attracted more and more attention recently in many network data analysis tasks. More recently, lots of HIN based methods are proposed [18, 19, 20, 21, 22], nevertheless, these methods mainly consider the low-order pairwise relations between nodes in HINs which might result in a sub-optimal node embedding. For example, as shown in Figure 1, in a conventional heterogeneous graph/network, different types of nodes are connected by different types of pairwise links which are not capable of capturing the high-order complex relation among those nodes explicitly. Inspired by the observation in [23] that many interactions in a HIN happen due to some event and the objects in each event form a complete semantic unit, in this paper, we introduce two kinds of definitions of objects (nodes) in

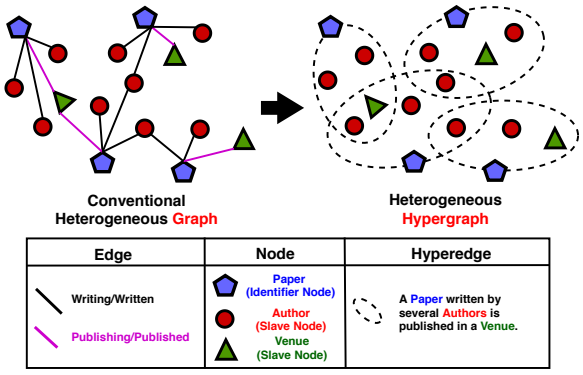

Fig. 1: A conventional heterogeneous graph (e.g. bibliographic network) and its corresponding heterogeneous hypergraph.

Haoyi Fan and Fengbin Zhang are with the School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China. (e-mail: isfanhy@gmail.com, zhangfengbin@hrbust.edu.cn).
Yuxuan Wei and Yue Gao are with the School of Software, BNRist, THUICBS, Tsinghua University, Beijing 100084, China. (e-mail: weiyuxua19@mails.tsinghua.edu.cn, kevin.gaoy@gmail.com).
Zuoyong Li is with the Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou 350121, China. (e-mail: fzulzytdq@126.com).
Changqing Zou is with the Sun Yat-sen University, Guangzhou 510275, China (e-mail: aaronzou1125@gmail.com).
Qionghai Dai is with THUICBS, Department of Automation, BNRist, Tsinghua University, Beijing 100084, China. (e-mail: qhdai@tsinghua.edu.cn).
(Yue Gao and Fengbin Zhang are the corresponding authors. This work is finished when Haoyi Fan visited Tsinghua University)

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

each event (hyperedge) as follows:

Definition 1.1. ***Identifier Node*** *is the node (object) that uniquely determines a hyperedge (event) based on a certain kind of semantics.*

Definition 1.2. ***Slave Node*** *is the member node (object) of a hyperedge (event) which is combined with identifier nodes to represent a complete semantics. A slave node does not necessarily uniquely exist in a hyperedge, it might be shared across multiple hyperedges.*

Take a bibliographic network as an example, as shown in Figure 1, the interactions in this network include not only low-order pairwise ones, e.g., *author writing paper*, *paper being published in a venue*, but also high-order complex ones e.g., *multiple authors are co-authored in the same one paper*, *multiple papers are published in the same one venue*. Moreover, there are also some high-order semantics determined by an event like, *"A paper written by several authors is published in a venue."* in the network. Consequently, the publication of a paper could be represented as a hyperedge, which is capable of encapsulating all the participating objects in this event. In this case, each paper acts as an identifier node, which uniquely determines a hyperedge (e.g. the publication event), and other objects like authors and venue play the role of slave nodes which are combined with an identifier node to represent complete publication semantics. Besides, we find that the event-induced identifier nodes also widely exist in many other HINs such as user-movie interest networks with the movie as an identifier node while the director, actor, and user are slave nodes, and user-business networks where the business is an identifier node while the location, category and user are slave nodes. Although the representation of the heterogeneous hypergraph encapsulates more information, more challenges are imposed, and how to effectively conduct representation learning with both high-order semantics and complex relation of objects, while preserving low-order pairwise structure under the heterogeneous settings is still an open problem.

To cope with the aforementioned challenges, in this paper, we propose a method of **Hete**rogeneous **H**yper**g**raph **V**ariational **A**uto**e**ncoder (HeteHG-VAE), for representation learning on a heterogeneous hypergraph, in which, the high-order semantics that implicitly exists in HINs is explicitly preserved in the latent space. Different from other hypergraph based methods [23, 24, 25], which are designed for event (hyperedge) prediction, while original pairwise relations between nodes are ignored, HeteHG-VAE aims at the pairwise level link prediction which is common in real-world HIN based applications e.g., the author identification task [26, 27] that models the likelihood of a paper being written by an author (paper-author pairwise relationship). To this end, we firstly model the conventional HIN as a heterogeneous hypergraph based on a certain kind of semantics to capture the high-order semantics and complex relations of different types of nodes while preserving the low-order pairwise relation in the original HIN. In the constructed heterogeneous hypergraph, we model each identifier node in the form of a hyperedge together with its salve nodes as the member nodes of this hyperedge, to represent a complete high-order semantics. Also, the original pairwise relation between the identifier nodes and the slave nodes can be preserved based on the incidence relationship between nodes and hyperedges. Then, by using stochastic variational inference [28], HeteHG-VAE infers the stochastic distribution of the latent variables from the latent space instead of the observation space to learn deep latent representations of nodes (slave nodes) and hyperedges (identifier nodes) in an unsupervised manner. In this, a heterogeneous hypergraph encoder as an inference model imposes a prior distribution on the latent space by minimizing a Kullback-Leibler (KL) divergence penalty. To capture the high-order relation, we learn two non-linear functions (node encoder and hyperedge encoder respectively) to learn the high-order non-linear behavior of the nodes in each hyperedge, by learning the node embedding and hyperedge embedding jointly. Here, the node embedding is used to capture the representative latent representation of each entity, while hyperedge embedding is used to capture the high-order non-linear relation among those entities. Following, a hypergraph decoder as a generative model takes the node embedding and hyperedge embedding as inputs to reconstruct the heterogeneous relation structures by minimizing the reconstruction errors. Moreover, due to the heterogeneity of each hyperedge (e.g., each hyperedge includes different types of nodes), a hyperedge attention module is designed to learn the importance of different types of nodes in each hyperedge for the final hyperedge embedding.

In sum, the main contributions of this paper are as follows:

- We develop a representation mapping between HINs and heterogeneous hypergraphs by introducing two definitions of the node, *identifier node* and *slave node*, to model different levels of relations in HINs. We model each identifier node in the form of a hyperedge, together with its slave nodes as the member nodes of the hyperedge, to capture both the high-order semantics and complex relations among nodes explicitly, while preserving the original low-order pairwise relations between the identifier nodes and the slave nodes in the HINs.
- We propose a general framework for representation learning on a heterogeneous hypergraph, named HeteHG-VAE, in which both the heterogeneity and the high-order relations among nodes are considered. Moreover, a hyperedge attention mechanism is employed for the importance modeling of different types of nodes in each hyperedge.
- By using variational inference, the stochastic distribution of the latent variables is effectively inferred from the latent space rather than the observation space. To the best of our knowledge, HeteHG-VAE is the first Bayesian deep generative model that considers the heterogeneity of the hypergraph.
- We conducted extensive experiments[1] on multiple popular real-world datasets, and the results demonstrate HeteHG-VAE's effectiveness compared to the state-of-the-art baselines for link prediction in HINs.

The rest of the paper is organized as follows. We first discuss some related works in Section 2. Section 3 outlines

1. All source code will be released upon the acceptance of this paper.

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

the notations and problem definition. In Section 4, we present the proposed method in detail. Lastly, we explain and analyze the results from our experiments in Section 5, before concluding the paper in Section 6.

2 Related Work

In this section, we first review the related studies about link prediction and then discuss hypergraph learning methods that are closely related to this work.

2.1 Link Prediction

Link prediction has been studied extensively in the machine learning communities [1, 9] with various applications [2, 3, 4, 5, 6, 8]. The existing traditional link prediction algorithms can be classified into two categories [29]: unsupervised and supervised methods. Unsupervised methods assign prediction scores to potential links based on the topology of the given networks, e.g., Adamic/Adar [30] captures the intuition that links are more likely to exist between pairs of nodes if they have more common neighbors. Moreover, path-based methods sum over the paths between two nodes to measure the likelihood of a link [9, 31]. For supervised methods, link prediction is often treated as a classification problem where the target class label indicates the presence or absence of a link between a pair of nodes, e.g., Supervised random walks [32] learns a function that assigns strengths to edges in the network such that a random walker is more likely to visit the nodes to which new links will be created in the future, and DEDS [33] constructs ensemble classifiers for better prediction accuracy.

Recently, a number of network embedding techniques have been proposed for link prediction tasks such as DeepWalk [10], LINE [11], and node2vec [12], which implicitly factorize some matrices for representation learning on the network. HSRL [34] considers both the local and global topologies of a network by recursively compressing an input network into a series of smaller networks and then applies Deepwalk, node2vec, or LINE on each compressed network. Some emerging graph neural networks (GNN) models that use different aggregation schemes for node embedding are also successfully applied for representation learning on the graph: Graph convolutional networks (GCNs) [35] based methods including GAE/VGAE [36, 37], $\mathcal{S}$-VGAE [38], Deep Generative Network Embedding (DGE) [39], Relational graph convolutional networks (R-GCNs) [40], and Linear-AE/Linear-VAE [41], are proposed for the link prediction task. SEAL [13] trains a classifier by mapping the subgraph patterns to link existence around each target link. GraphSAGE [42] concatenates the nodes feature in addition to mean/max/LSTM pooled neighborhood information. Graph Attention Networks (GAT) [43] aggregate neighborhood information according to trainable attention weights. Position-aware Graph Neural Networks (P-GNNs) [44] aim to compute position-aware node embeddings. There are also some works towards graph learning in non-Euclidean spaces [45, 46], e.g. hyperbolic space [47], to capture the properties of scale-free and hierarchical structures of real-world graph data. Hyperbolic graph neural networks [45] extend GNN to learn graph embeddings in hyperbolic space, while hyperbolic graph convolutional networks [46] employ a trainable curvature learning layer to map Euclidean input features to embeddings in hyperbolic space.

Moreover, some recent works have focused on the heterogeneous information networks [18, 19, 48], which have multiple types of links and complex dependency structures. Some methods such as HIN2Vec [18] and Metapath2vec [19] utilize meta-path [49] based structural information to capture the heterogeneity of the graph for node embedding. HEER [20] studies the problem of comprehensive transcription for a heterogeneous graph by leveraging the edge representations and heterogeneous metrics for the final graph embedding. HGT [50] employs node- and edge-type dependent parameters to characterize the heterogeneous attention over each edge. There are also some knowledge graph based models which are designed under the heterogeneous settings, including TransE [51] and ConvE [52]. TransE treats each edge as a triplet composed of two nodes and an edge type, and then translates embeddings for modeling multi-relational data. ConvE is similar to TransE but goes beyond simple distance or similarity functions by using the deep neural networks to score the triplet. However, it is noted that the heterogeneity of networks or the high-order relation and semantics among nodes existing commonly have not been systematically explored in previous works.

2.2 Hypergraph Learning

A hypergraph is a generalization of a network (graph) in which a hyperedge can link more than two nodes [53]. Hypergraph learning [54] has been studied to model high-order correlations among data and achieves successful applications in different domains recently [55], such as image ranking [56], 3D visual classification [57, 58], social relationship analysis [59, 60], and active learning [61]. More recently, hypergraph neural networks (HGNN) [62] and hypergraph convolution networks (HyperGCN) [63], are proposed as extension works of graph convolution network frameworks. However, those methods are mainly designed for homogeneous settings where the nodes and hyperedges in a hypergraph are of the same type.

Different from the above mentioned works, there are several works conducted in the heterogeneous settings for network clustering [64, 65] and embedding [23, 24, 25]. In [64], an inhomogeneous hypergraph partitioning method is proposed that assumes different hyperedge cuts have different weights and assigns different costs to different cuts during hypergraph learning. In [65], p-Laplacians is proposed for submodular hypergraphs to solve the scalability problem of clique expansion based methods [64]. In [23], object embeddings are learned based on events in heterogeneous information networks where a hyperedge encompasses the objects participating in a specific event. In [24], a Deep Hyper-Network Embedding (DHNE) model is proposed to embed a hypergraph with indecomposable hyperedges, in which any subset of nodes in a hyperedge cannot form another complete hyperedge. In [25], HHNE is proposed based on the neural tensor network (NTN) [3] to learn a tuple-wise similarity score function for hyperedge scoring in a heterogeneous attributed hypergraph. However, those hypergraph embedding methods mainly focus on

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

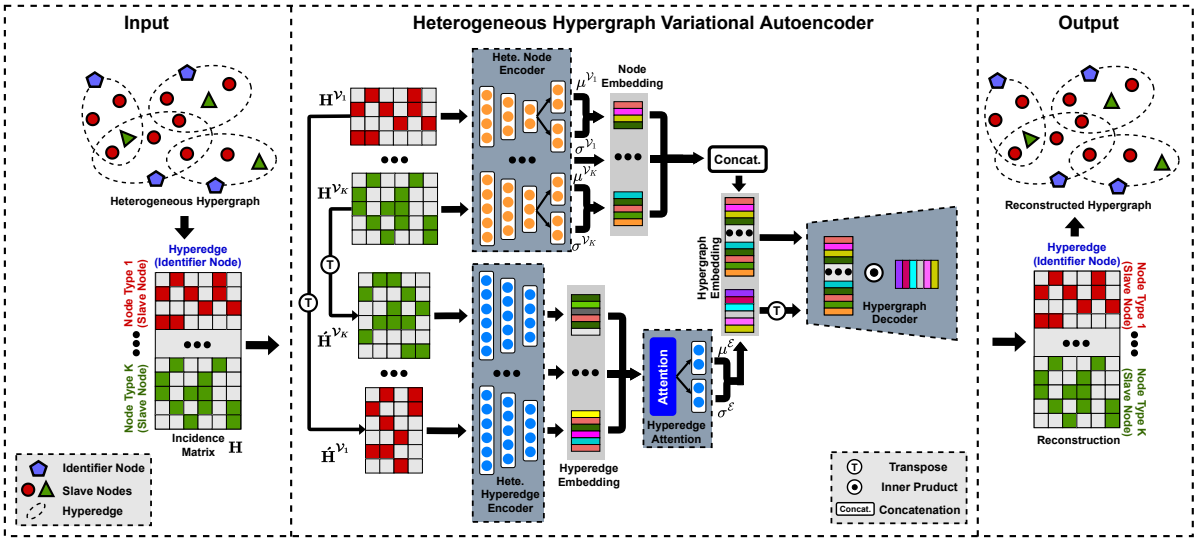

Fig. 2: The framework of the proposed Heterogeneous Hypergraph Variational Autoencoder

TABLE 1: Notations.

Notation	Description
$\mathcal{G}^h/\mathcal{G}$	HIN/heterogeneous hypergraph.
$\mathcal{V}^h/\mathcal{V}$	A set of nodes in HIN/heterogeneous hypergraph.
$\mathcal{E}^h/\mathcal{E}$	A set of edges/hyperedges in HIN/heterogeneous hypergraph.
M/N	The number of nodes/hyperedges.
D	The dimension of latent variables.
$\mathbf{H} \in \mathbb{R}^{M\times N}$	Incidence matrix of a hypergraph.
$\acute{\mathbf{H}} \in \mathbb{R}^{N\times M}$	Incidence matrix of a hypergraph duality.
$\mathbf{Z}^{\mathcal{V}} \in \mathbb{R}^{M\times D}$	Latent embedding of nodes.
$\mathbf{Z}^{\mathcal{E}} \in \mathbb{R}^{N\times D}$	Latent embedding of hyperedges.

hyperedge prediction that takes all objects in a hyperedge as a whole, while ignoring the pairwise relation between different types of objects, which is common in HINs. On the contrary, our method aims at pairwise link prediction in HINs by considering both high-order semantics and relations of objects while preserving the low-order pairwise structure of the network.

3 Notations and Problem Statement

The notations used throughout the paper are formally summarized in Table 1. The heterogeneous information network (HIN) [66] $\mathcal{G}_h = \{\mathcal{V}_h, \mathcal{E}_h\}$ is defined as a network with multiple types of nodes or edges, where $\mathcal{V}_h$ denotes the set of nodes and $\mathcal{E}_h$ is the set of edges. Moreover, two types of mapping functions $f_{\mathcal{V}_h} : \mathcal{V}_h \rightarrow \mathcal{O}$ and $f_{\mathcal{E}_h} : \mathcal{E}_h \rightarrow \mathcal{R}$ are defined in HIN to map nodes and edges to predefined types respectively. For example, in a bibliographic network, $\mathcal{O}$={Paper, Author, Venue}, $\mathcal{R}$={Writing/Written, Publishing/Published}.

Similarly, a heterogeneous hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is defined as a hypergraph with multiple types of nodes or hyperedges, where $\mathcal{V} = \{v_1, ..., v_M\}$ denotes the set of nodes and $\mathcal{E} = \{e_1, ..., e_N\}$ is the set of hyperedges. Moreover, two types of mapping functions $f_{\mathcal{V}} : \mathcal{V} \rightarrow \mathcal{O}$ and $f_{\mathcal{E}} : \mathcal{E} \rightarrow \mathcal{R}$ are defined to map nodes and hyperedges to the predefined types respectively. For example, in a bibliographic hypergraph, $\mathcal{O}$={Paper, Author, Venue}, $\mathcal{R}$={Co-paper, Co-author, Co-venue, Paper publication event.}. Given a hypergraph, its incidence matrix is denoted as $\mathbf{H} \in \mathbb{R}^{M\times N}$ with $\mathbf{H}_{i,j} = 1$ if the j-th hyperedge is incident with the i-th node; otherwise, $\mathbf{H}_{i,j} = 0$. Given a heterogeneous hypergraph with K types of nodes, there are K sub-incidence matrices $\{\mathbf{H}^{\mathcal{V}_k}\}_{k=1}^{K}$, each of which represents the incidence relation of k-th type of nodes. For each hypergraph, its duality [67] is defined as $\acute{\mathcal{G}} = \{\acute{\mathcal{V}}, \acute{\mathcal{E}}\}$, where $\acute{\mathcal{V}} = \mathcal{E}$ and $\acute{\mathcal{E}} = \{\acute{e}_1, ...\acute{e}_M\}$ such that $\acute{e}_i = \{e \in \mathcal{E} | v_i \in e\}$. Therefore, the incidence matrix $\acute{\mathbf{H}} \in \mathbb{R}^{N\times M}$ of duality $\acute{\mathcal{G}}$ is the transpose of incidence matrix $\mathbf{H} \in \mathbb{R}^{M\times N}$ of $\mathcal{G}$.

We focus on the problem of representation learning on a heterogeneous hypergraph for the link prediction task in HINs. The definition of this problem is as follows:

Problem 3.1. *Given a heterogeneous information network $\mathcal{G}_h = \{\mathcal{V}_h, \mathcal{E}_h\}$, we firstly construct a heterogeneous hypergraph $\mathcal{G} = \{\mathcal{V} = \{\mathcal{V}_1, ..., \mathcal{V}_K\}, \mathcal{E}\}$ based on a certain kind of high-order semantics reflected in the interactions among nodes (objects) in $\mathcal{G}_h$. In the constructed $\mathcal{G}$, each identifier node is modeled in the form of a hyperedge, together with its K types of salve nodes as the member nodes of this hyperedge, to represent a complete high-order semantics. Then, we aim to represent nodes (slave nodes) and hyperedges (identifier nodes) in the low-dimensional vector space by learning a mapping function f_{emb}:*

$$f_{emb}(\mathcal{G}) \rightarrow \mathbf{Z}^{\mathcal{V}}, \mathbf{Z}^{\mathcal{E}} \tag{1}$$

where $\mathbf{Z}^{\mathcal{V}} \in \mathbb{R}^{M\times D}$ and $\mathbf{Z}^{\mathcal{E}} \in \mathbb{R}^{N\times D}$ are the learned low-dimensional embedding of M nodes and N hyperedges respectively, $D \ll min(|\mathcal{V}|, |\mathcal{E}|)$ is the dimension of embedding, and a scoring function f_{sco}:

$$f_{sco}(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{E}}) \tag{2}$$

where $\mathbf{Z}_i^{\mathcal{V}}$ and $\mathbf{Z}_j^{\mathcal{E}}$ are the embeddings of node i and hyperedge j respectively, and the likelihood of the link between slave node i and identifier node j in the original $\mathcal{G}_h$ can be measured by f_{sco}.

4 Proposed Method

In this section, we introduce the proposed heterogeneous hypergraph variational autoencoder in detail.

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

4.1 Overall Framework

The proposed HeteHG-VAE is a Bayesian deep generative model that learns the heterogeneous hypergraph embedding in low-dimensional vector space, where the heterogeneity and high-order relations among nodes are preserved. As shown in Figure 2, HeteHG-VAE consists of a heterogeneous hypergraph encoder as an inference model to infer the desired embedding, and a hypergraph decoder as a generative model to reconstruct the original hypergraph topologies. Specifically, a heterogeneous hypergraph encoder consists of a heterogeneous node encoder to encode the observed node representation $\{\mathbf{H}^{\mathcal{V}_k}\}_{k=1}^{K}$ of different types of nodes for node embedding, and a heterogeneous hyperedge encoder to encode hyperedge representation $\{\acute{\mathbf{H}}^{\mathcal{V}_k}\}_{k=1}^{K}$ in forms of hypergraph duality for high-order relation embedding. Following a heterogeneous hyperedge encoder, a hyperedge attention module is employed to embed the identifier node induced high-order semantics by learning the importance of different types of nodes in each hyperedge, and then fusing different types of hyperedge embedding based on the learned importance weights. Finally, given the learnt hypergraph embedding including node embedding and weighted hyperedge embedding, a hypergraph decoder is used to reconstruct the original topology of the hypergraph.

4.2 Variational Evidence Lower Bound

Traditional autoencoders encourage a perfect reconstruction at the cost of overfitting to the training data by only penalizing the reconstruction error [68]. However, networks in the real-world are often incomplete and noisy, therefore, a more robust embedding is needed for the downstream link prediction task. In the proposed model, by using variational inference, the stochastic distribution of the latent variables can be effectively inferred from the latent space instead of the observation space for high-quality embeddings.

Given the observed data points $\{\mathbf{H}^{\mathcal{V}_k}\}_{k=1}^{K}$, HeteHG-VAE assumes that each data point is generated i.i.d from the following process: Firstly, generate the latent node embeddings $\{\mathbf{Z}^{\mathcal{V}_k}\}_{k=1}^{K}$, and high-order relation embedding $\mathbf{Z}^{\mathcal{E}}$ by drawing $\mathbf{Z}^{\mathcal{V}_k} \sim p_0(\mathbf{Z}^{\mathcal{V}_k}), \mathbf{Z}^{\mathcal{E}} \sim p_0(\mathbf{Z}^{\mathcal{E}})$ from a parameter-free prior distribution $p_0(\bullet)$, e.g., Gaussian distribution in this paper. Then, the data points $\{\mathbf{H}^{\mathcal{V}_k}\}_{k=1}^{K}$ are generated from a conditional distribution $\mathbf{H}^{\mathcal{V}_k} \sim p(\mathbf{H}^{\mathcal{V}_k}|\mathbf{Z}^{\mathcal{V}_k}, \mathbf{Z}^{\mathcal{E}}; \boldsymbol{\lambda}^{\mathcal{V}_k})$, where $\boldsymbol{\lambda}^{\mathcal{V}_k}$ is the parameters of the distribution. Based on this, the generation of each data point is performed on its latent node embedding and its associated high-order relation embedding conditionally.

HeteHG-VAE aims to optimize $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^{\mathcal{V}_k}\}_{k=1}^{K}$ to maximize the log-likelihood of the observed data $\{\mathbf{H}^{\mathcal{V}_k}\}_{k=1}^{K}$, which is derived according to Jensen's Inequality as follows:

$$\begin{aligned}
&\log p(\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\lambda}) \\
=&\log \int_{\mathbf{Z}^{\mathcal{V}_1}} ... \int_{\mathbf{Z}^{\mathcal{V}_K}} \int_{\mathbf{Z}^{\mathcal{E}}} p(\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}, \\
&\mathbf{Z}^{\mathcal{V}_1}, ..., \mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}}; \boldsymbol{\lambda}) \mathrm{d}\mathbf{Z}^{\mathcal{V}_1} ... \mathrm{d}\mathbf{Z}^{\mathcal{V}_K} \mathrm{d}\mathbf{Z}^{\mathcal{E}} \\
\geqslant&\mathbb{E}_q[\log \frac{p(\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{V}_1}, ..., \mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}}; \boldsymbol{\lambda})}{q(\mathbf{Z}^{\mathcal{V}_1}, ..., \mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}}|\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\theta})}] \\
:=&\mathcal{L}(\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\theta}, \boldsymbol{\lambda})
\end{aligned} \tag{3}$$

where $q(\mathbf{Z}^{\mathcal{V}_1}, ..., \mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}}|\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\theta})$ is the variational posterior used to approximate the true posterior $p(\mathbf{Z}^{\mathcal{V}_1}, ..., \mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}}|\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K})$, $\boldsymbol{\theta}$ indicates the parameters needed to be estimated. So, the evidence lower bound $\mathcal{L}(\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\theta}, \boldsymbol{\lambda})$ on the marginal likelihood of $\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}$ is expressed as follows by applying logarithmic product rule to Eq. (3):

$$\begin{aligned}
&\mathcal{L}(\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\theta}, \boldsymbol{\lambda}) \\
=&\mathbb{E}_q[\log (p(\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}|\mathbf{Z}^{\mathcal{V}_1}, ..., \mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}}; \boldsymbol{\lambda}) \\
&\cdot \frac{p(\mathbf{Z}^{\mathcal{V}_1}, ..., \mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}})}{q(\mathbf{Z}^{\mathcal{V}_1}, ..., \mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}}|\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\theta})})] \\
=&\mathbb{E}_q[\log p(\mathbf{H}^{\mathcal{V}_1}|\mathbf{Z}^{\mathcal{V}_1}, \mathbf{Z}^{\mathcal{E}}; \boldsymbol{\lambda}^{\mathcal{V}_1})] + ... \\
&+ \mathbb{E}_q[\log p(\mathbf{H}^{\mathcal{V}_K}|\mathbf{Z}^{\mathcal{V}_K}, \mathbf{Z}^{\mathcal{E}}; \boldsymbol{\lambda}^{\mathcal{V}_K})] \\
&- \mathrm{KL}(q(\mathbf{Z}^{\mathcal{V}_1}|\mathbf{H}^{\mathcal{V}_1}; \boldsymbol{\theta}^{\mathcal{V}_1})||p(\mathbf{Z}^{\mathcal{V}_1})) - ... \\
&- \mathrm{KL}(q(\mathbf{Z}^{\mathcal{V}_K}|\mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\theta}^{\mathcal{V}_K})||p(\mathbf{Z}^{\mathcal{V}_K})) \\
&- \mathrm{KL}(q(\mathbf{Z}^{\mathcal{E}}|\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K}; \boldsymbol{\theta}^{\mathcal{E}})||p(\mathbf{Z}^{\mathcal{E}}))
\end{aligned} \tag{4}$$

where the term $\mathrm{KL}(\bullet||\bullet)$ is the Kullback-Leibler (KL) divergence. This can be viewed as a regularization that pulls the variational distribution $q(\bullet|\bullet;\boldsymbol{\theta})$ towards the prior distribution $p(\bullet)$, $\mathbb{E}_q[\sum \log p(\bullet|\boldsymbol{\lambda})]$ is the likelihood of reconstruction for each data point given the latent embedding constrained on its distribution learned by HeteHG-VAE, and $\boldsymbol{\lambda}$ is the parameter that needs to be estimated.

To estimate the parameters of the objective in Eq. (4), HeteHG-VAE consists of a heterogeneous hypergraph encoder as an inference model to parametrize the approximate posterior $q(\mathbf{Z}^{\bullet}|\mathbf{H}^{\bullet};\boldsymbol{\theta}^{\bullet})$, and a hypergraph decoder as a generative model to parametrize the distribution $p(\mathbf{H}^{\bullet}|\mathbf{Z}^{\bullet}, \mathbf{Z}^{\mathcal{E}}; \boldsymbol{\lambda}^{\bullet})$, respectively.

4.3 Inference Model

As an inference model, a heterogeneous hypergraph encoder consists of a heterogeneous node encoder and a heterogeneous hyperedge encoder to project different types of nodes and their associated high-order relations into a common feature space.

4.3.1 Heterogeneous Node Encoder

The heterogeneous node encoder firstly conducts a non-linear mapping from the observed heterogeneous space $\mathbf{H}^{\mathcal{V}_k}$ to a common latent space $\widetilde{\mathbf{Z}}^{\mathcal{V}_k}$, $k = 1, ..., K$, which is as follows:

$$\widetilde{\mathbf{Z}}^{\mathcal{V}_k} = f_{\mathcal{V}_k}(\mathbf{H}^{\mathcal{V}_k}\mathbf{X}^{\mathcal{E}}\mathbf{W}^{\mathcal{V}_k} + \mathbf{b}^{\mathcal{V}_k}) \tag{5}$$

where $\mathbf{X}^{\mathcal{E}} \in \mathbb{R}^{N \times F}$ can be the initial F-dimension hyperedge features if available, otherwise, $\mathbf{X}^{\mathcal{E}} = \mathrm{I}$ is simply an identity matrix. $f_{\mathcal{V}_k}(\bullet)$ is a non-linear activation function such as $ReLU(x) = max(0, x)$ or $\mathrm{Tanh}(x) = \frac{2}{1+e^{-2x}} - 1$, here, we empirically select Tanh in the experiments. $\mathbf{W}^{\mathcal{V}_k} \in \mathbb{R}^{D_{in}^{\mathcal{V}_k} \times D_{out}^{\mathcal{V}_k}}$ and $\mathbf{b}^{\mathcal{V}_k} \in \mathbb{R}^{D_{out}^{\mathcal{V}_k}}$ are the weight and bias learned by the encoder, $D_{in}^{\mathcal{V}_k}$ and $D_{out}^{\mathcal{V}_k}$ are the dimensionalities of $\mathbf{X}^{\mathcal{E}}$ and $\widetilde{\mathbf{Z}}^{\mathcal{V}_k}$, respectively.

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

Given the projected node embedding $\widetilde{\mathbf{Z}}^{\mathcal{V}_k}$, two individual fully connected layers are then employed to estimate the means $\boldsymbol{\mu}^{\mathcal{V}_k}$ and variances $\boldsymbol{\sigma}^{\mathcal{V}_k}$ of $q(\mathbf{Z}^{\mathcal{V}_k}|\mathbf{H}^{\mathcal{V}_k};\boldsymbol{\theta}^{\mathcal{V}_k})$:

$$\boldsymbol{\mu}^{\mathcal{V}_k} = \widetilde{\mathbf{Z}}^{\mathcal{V}_k}\mathbf{W}_{\boldsymbol{\mu}}^{\mathcal{V}_k} + \mathbf{b}_{\boldsymbol{\mu}}^{\mathcal{V}_k} \tag{6}$$

$$\boldsymbol{\sigma}^{\mathcal{V}_k} = \widetilde{\mathbf{Z}}^{\mathcal{V}_k}\mathbf{W}_{\boldsymbol{\sigma}}^{\mathcal{V}_k} + \mathbf{b}_{\boldsymbol{\sigma}}^{\mathcal{V}_k} \tag{7}$$

where $\mathbf{W}_{\boldsymbol{\mu}}^{\mathcal{V}_k}, \mathbf{W}_{\boldsymbol{\sigma}}^{\mathcal{V}_k} \in \mathbb{R}^{D_{out}^{\mathcal{V}_k} \times D}$ are the two learnable weight matrices, and $\mathbf{b}_{\boldsymbol{\mu}}^{\mathcal{V}_k}, \mathbf{b}_{\boldsymbol{\sigma}}^{\mathcal{V}_k} \in \mathbb{R}^D$ are biases, respectively. D is the dimensionality of the final node embedding $\mathbf{Z}^{\mathcal{V}_k}$, which is sampled by the following process:

$$\mathbf{Z}^{\mathcal{V}_k} = \boldsymbol{\mu}^{\mathcal{V}_k} + \boldsymbol{\sigma}^{\mathcal{V}_k} \odot \boldsymbol{\epsilon} \tag{8}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\odot$ indicates the element-wise product. All the learned parameters $\{\mathbf{W}^{\mathcal{V}_k}, \mathbf{W}_{\boldsymbol{\mu}}^{\mathcal{V}_k}, \mathbf{W}_{\boldsymbol{\sigma}}^{\mathcal{V}_k}, \mathbf{b}^{\mathcal{V}_k}, \mathbf{b}_{\boldsymbol{\mu}}^{\mathcal{V}_k}, \mathbf{b}_{\boldsymbol{\sigma}}^{\mathcal{V}_k}\}$ constitute the parameters $\boldsymbol{\theta}^{\mathcal{V}_k}$ of the posterior $q(\mathbf{Z}^{\mathcal{V}_k}|\mathbf{H}^{\mathcal{V}_k};\boldsymbol{\theta}^{\mathcal{V}_k})$.

4.3.2 Heterogeneous Hyperedge Encoder

To encode the high-order semantics and complex relations represented in the form of hyperedges, a heterogeneous hyperedge encoder firstly conducts a non-linear feature transformation from the observed heterogeneous space $\acute{\mathbf{H}}^{\mathcal{V}_k}$ into a common latent space $\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}}$, $k = 1, ..., K$, which is as follows:

$$\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}} = f_{\mathcal{E}^{\mathcal{V}_k}}(\acute{\mathbf{H}}^{\mathcal{V}_k}\mathbf{X}^{\mathcal{V}_k}\mathbf{W}^{\mathcal{E}^{\mathcal{V}_k}} + \mathbf{b}^{\mathcal{E}^{\mathcal{V}_k}}) \tag{9}$$

where $\mathbf{X}^{\mathcal{V}_k} \in \mathbb{R}^{M^{\mathcal{V}_k} \times F}$ can be the initial F-dimension node features if available, otherwise, $\mathbf{X}^{\mathcal{V}_k} = \mathrm{I}$ is simply an identity matrix. $f_{\mathcal{E}^{\mathcal{V}_k}}(\bullet)$ is a non-linear activation function such as Tanh used here, $\mathbf{W}^{\mathcal{E}^{\mathcal{V}_k}} \in \mathbb{R}^{D_{in}^{\mathcal{E}^{\mathcal{V}_k}} \times D_{out}^{\mathcal{E}^{\mathcal{V}_k}}}$ and $\mathbf{b}^{\mathcal{E}^{\mathcal{V}_k}} \in \mathbb{R}^{D_{out}^{\mathcal{E}^{\mathcal{V}_k}}}$ are the learnable weight and bias, and $D_{in}^{\mathcal{E}^{\mathcal{V}_k}}$ and $D_{out}^{\mathcal{E}^{\mathcal{V}_k}}$ are the dimensionalities of $\mathbf{X}^{\mathcal{V}_k}$ and $\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}}$, respectively.

Given the projected type-specific embedding $\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}}$, the hyperedge attention module is proposed to learn the importance of different types of nodes in each hyperedge. As shown in Figure 3, hyperedge attention module aims at learning a non-linear attention scheme over different types of nodes in each hyperedge. Firstly, the type-specific embeddings $\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}}$ is transformed to a common transformation space through a non-linear transformation layer:

$$\widetilde{\mathbf{Z}}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}} = \mathrm{Tanh}(\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}}\mathbf{W}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}} + \mathbf{b}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}}) \tag{10}$$

where $\mathbf{W}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}} \in \mathbb{R}^{D_{out}^{\mathcal{E}^{\mathcal{V}_k}} \times D_{trans}^{\mathcal{E}}}$ and $\mathbf{b}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}} \in \mathbb{R}^{D_{trans}^{\mathcal{E}}}$ are the learnable weight and bias of the transformation, respectively, and $D_{trans}^{\mathcal{E}}$ is the dimensionality of the transformed embedding. Next, the importance weights $\boldsymbol{\alpha}^{\mathcal{E}^{\mathcal{V}_k}} \in \mathbb{R}^{N \times 1}$ of the k-th type of node for all N hyperedges are calculated based on the similarity between the transformed embedding $\widetilde{\mathbf{Z}}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}}$ and the type preference vector $\mathbf{P} \in \mathbb{R}^{D_{trans}^{\mathcal{E}} \times 1}$, which is randomly initialized and jointly updated during training:

$$\widetilde{\boldsymbol{\alpha}}^{\mathcal{E}^{\mathcal{V}_k}} = \widetilde{\mathbf{Z}}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}} \cdot \mathbf{P} \tag{11}$$

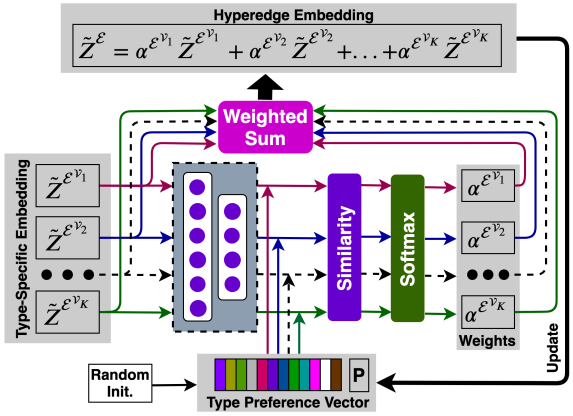

Fig. 3: Hyperedge Attention Module.

and then the importance weights are normalized through a softmax function:

$$\boldsymbol{\alpha}^{\mathcal{E}^{\mathcal{V}_k}} = \frac{exp(\widetilde{\boldsymbol{\alpha}}^{\mathcal{E}^{\mathcal{V}_k}})}{\sum_{k=1}^{K} exp(\widetilde{\boldsymbol{\alpha}}^{\mathcal{E}^{\mathcal{V}_k}})} \tag{12}$$

Then, the fused hyperedge embedding can be obtained by the weighted sum of the projected type-specific embedding $\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}}$ according to the learned importance weights $\boldsymbol{\alpha}^{\mathcal{E}^{\mathcal{V}_k}}$:

$$\widetilde{\mathbf{Z}}^{\mathcal{E}} = \sum_{k=1}^{K} \boldsymbol{\alpha}^{\mathcal{E}^{\mathcal{V}_k}} \widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}} \tag{13}$$

Given the fused hyperedge embedding $\widetilde{\mathbf{Z}}^{\mathcal{E}}$, two individual fully connected layers are then employed to estimate the means $\boldsymbol{\mu}^{\mathcal{E}}$ and variances $\boldsymbol{\sigma}^{\mathcal{E}}$ of $q(\mathbf{Z}^{\mathcal{E}}|\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K};\boldsymbol{\theta}^{\mathcal{E}})$:

$$\boldsymbol{\mu}^{\mathcal{E}} = \widetilde{\mathbf{Z}}^{\mathcal{E}}\mathbf{W}_{\boldsymbol{\mu}}^{\mathcal{E}} + \mathbf{b}_{\boldsymbol{\mu}}^{\mathcal{E}} \tag{14}$$

$$\boldsymbol{\sigma}^{\mathcal{E}} = \widetilde{\mathbf{Z}}^{\mathcal{E}}\mathbf{W}_{\boldsymbol{\sigma}}^{\mathcal{E}} + \mathbf{b}_{\boldsymbol{\sigma}}^{\mathcal{E}} \tag{15}$$

where $\mathbf{W}_{\boldsymbol{\mu}}^{\mathcal{E}}, \mathbf{W}_{\boldsymbol{\sigma}}^{\mathcal{E}} \in \mathbb{R}^{D_{trans}^{\mathcal{E}} \times D}$ are the two learnable weight matrices, and $\mathbf{b}_{\boldsymbol{\mu}}^{\mathcal{E}}, \mathbf{b}_{\boldsymbol{\sigma}}^{\mathcal{E}} \in \mathbb{R}^D$ are biases, respectively. D is the dimensionality of the final hyperedge embedding $\mathbf{Z}^{\mathcal{E}}$, which is sampled by the following process:

$$\mathbf{Z}^{\mathcal{E}} = \boldsymbol{\mu}^{\mathcal{E}} + \boldsymbol{\sigma}^{\mathcal{E}} \odot \boldsymbol{\epsilon} \tag{16}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\odot$ indicates the element-wise product. Similarly, $\boldsymbol{\theta}^{\mathcal{E}} = \{\mathbf{W}^{\mathcal{E}^{\mathcal{V}_k}}, \mathbf{W}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}}, \mathbf{W}_{\boldsymbol{\mu}}^{\mathcal{E}}, \mathbf{W}_{\boldsymbol{\sigma}}^{\mathcal{E}}, \mathbf{b}^{\mathcal{E}^{\mathcal{V}_k}}, \mathbf{b}_{\boldsymbol{trans}}^{\mathcal{E}^{\mathcal{V}_k}}, \mathbf{b}_{\boldsymbol{\mu}}^{\mathcal{E}}, \mathbf{b}_{\boldsymbol{\sigma}}^{\mathcal{E}}\}$ is the set of the estimated posterior parameters of $q(\mathbf{Z}^{\mathcal{E}}|\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_K};\boldsymbol{\theta}^{\mathcal{E}})$.

4.4 Generative Model

As a generative model, the hypergraph decoder aims to decode from the node embedding $\mathbf{Z}^{\mathcal{V}_k}$ and hyperedge embedding $\mathbf{Z}^{\mathcal{E}}$ to the generative random variables, from which the original hypergraph topologies can be reconstructed.

As the observed space $\mathbf{H}^{\mathcal{V}_k}$ is of the binary-valued data points, Bernoulli distribution parametrized by $\boldsymbol{\mathcal{H}}_{ij}^{\mathcal{V}_k}$ is utilized to estimate the distributions $p(\mathbf{H}_{i,j}^{\mathcal{V}_k}|\mathbf{Z}_i^{\mathcal{V}_k}, \mathbf{Z}_j^{\mathcal{E}};\boldsymbol{\lambda}^{\mathcal{V}_k})$:

$$\boldsymbol{\mathcal{H}}_{ij}^{\mathcal{V}_k} = Sigmoid(\mathbf{Z}_i^{\mathcal{V}_k}(\mathbf{Z}_j^{\mathcal{E}})^{\mathrm{T}}) \tag{17}$$

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

$$p(\mathbf{H}_{i,j}^{\mathcal{V}_k}|\mathbf{Z}_i^{\mathcal{V}_k}, \mathbf{Z}_j^{\mathcal{E}}; \boldsymbol{\lambda}^{\mathcal{V}_k}) = Ber(\boldsymbol{\mathcal{H}}_{ij}^{\mathcal{V}_k}) \tag{18}$$

where $Sigmoid(\bullet)$ is the sigmoid activation function.

Finally, the overall process of HeteHG-VAE is described in Algorithm 1.

4.5 Link Prediction

Based on the inferred embedding $\mathbf{Z}^{\mathcal{V}_k}$ and $\mathbf{Z}^{\mathcal{E}}$, the likelihood of the link between slave node i and identifier node j is measured by the similarity between the two embeddings, which is as follows:

$$f_{sco}(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{E}}) = Sim(\mathbf{Z}_i^{\mathcal{V}}, \mathbf{Z}_j^{\mathcal{E}}) \tag{19}$$

where $Sim(\bullet)$ is a similarity measure function, which can be either simple similarity measuring such as Euclidean distance and cosine similarity, or a more complex edge classifier such as logistic regression.

4.6 Connection with Previous Work

We choose variational graph autoencoder (VGAE) [36], a recent VAE [28] based generative model for graph representation learning as the base model to discuss the connection between the proposed model and previous work.

VGAE extends the variational autoencoder on graph structure data by inferring the latent variables z_i for each node v_i as the node representations in the latent space:

$$q(\mathbf{Z}) = \prod_{i=1}^{M+N} q(z_i|\mathbf{A}) \tag{20}$$

where $q(z_i|\mathbf{A}) = \mathcal{N}(z_i|\mu_i, \text{diag}(\sigma_i^2))$, $\mathbf{A} \in \mathbb{R}^{(M+N)\times(M+N)}$ is the adjacency matrix of the graph, and μ_i and $\text{diag}(\sigma_i^2)$ are the Gaussian parameters learned by two GCN branches as follows:

$$\mu = \text{GCN}_\mu(\mathbf{A}, \mathbf{X}),\ \sigma = \text{GCN}_\sigma(\mathbf{A}, \mathbf{X}) \tag{21}$$

Here, $\text{GCN}(\bullet) = \tilde{\mathbf{A}}\text{ReLU}(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W_0})\mathbf{W_1}$ is a two-layer GCN. $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix, with $\mathbf{I}$ being the identity matrix and $\mathbf{D}$ being the diagonal degree matrix of $\mathbf{A}+\mathbf{I}$. $\mathbf{X}$ is the node feature matrix. $\mathbf{W}_0$ and $\mathbf{W}_1$ are the trainable weights and $\text{ReLU}(\bullet)$ is activation function $\text{ReLU}(x) = max(0, x)$. Different from VGAE, in HeteHG-VAE, for the estimation of node distribution, the Gaussian parameters, i.e. μ, is inferred by $\mu = \text{Tanh}(\mathbf{H}\mathbf{X}\mathbf{W_0} + \mathbf{b_0})\mathbf{W_1} + \mathbf{b_1}$.

The similarity between HeteHG-VAE and VGAE is that a message passing mechanism including feature transformation and aggregation from neighbors is employed in both models. For example, the graph convolution operation of $\tilde{\mathbf{A}}\mathbf{X}\mathbf{W_0}$ in VGAE aims to aggregate features of neighbors to the target node following a linear transform, and then another following GCN layer is used to estimate the parameter of Gaussian distribution. Similarly, the operation of $\mathbf{H}\mathbf{X}\mathbf{W}^{\mathcal{V}}$ in HeteHG-VAE also aims to generate the node embedding by aggregating the features of its incident hyperedges. Then, a following dense layer is used for distribution estimation, which is slightly different from VGAE. The main differences between HeteHG-VAE and VGAE lie in the model inputs and the manner of network embedding. VGAE takes the normalized adjacency matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{(M+N)\times(M+N)}$ as the inputs to learn the embeddings of both identifier nodes and slave nodes directly by aggregating features from neighbors in a homogeneous setting. However, in HeteHG-VAE, the incidence matrix $\mathbf{H} \in \mathbb{R}^{M\times N}$ and its duality $\acute{\mathbf{H}} \in \mathbb{R}^{N\times M}$, transformed from the adjacency matrix $\mathbf{A}$ are taken as inputs to infer the embeddings of slave nodes and identifier nodes respectively. In the node encoder, the operation of $\mathbf{H}\mathbf{X}\mathbf{W}^{\mathcal{V}}$ generates the embeddings of the nodes by aggregating the features of its incident hyperedges. While in the hyperedge encoder, the operation of $\acute{\mathbf{H}}\acute{\mathbf{X}}\mathbf{W}^{\mathcal{E}}$ generates the embeddings of the hyperedges by aggregating the features of its incident nodes. Moreover, considering the heterogeneity of the network, we first extract different sub-hypergraphs for different types of nodes in form of the sub-incidence matrices $\mathbf{H}^{\mathcal{V}_k}$, and conduct a type-aware node embedding in a heterogeneous setting. We also introduce the attention mechanism in the hyperedge for a better embedding quality by learning the importance of different types of nodes in each hyperedge.

Algorithm 1 HeteHG-VAE.

Require:
$\mathbf{H} = \{\mathbf{H}^{\mathcal{V}_k}\}_{k=1}^K$: Incidence matrix of hypergraph $\mathcal{G}$ with K types of nodes.
Ensure:
$\mathbf{Z}^{\mathcal{V}} = \{\mathbf{Z}^{\mathcal{V}_k}\}_{k=1}^K$: Node embedding;
$\mathbf{Z}^{\mathcal{E}}$: Hyperedge embedding.

for $epoch$ = 1 to T **do**
  Project the observed space $\mathbf{H}^{\mathcal{V}_k}$ of k-th type of node $\boldsymbol{\mathcal{V}^k}$ to a common latent space $\widetilde{\mathbf{Z}}^{\mathcal{V}_k}$ via Eq. (5);
  Sample the node embedding $\mathbf{Z}^{\mathcal{V}_k}$ from the estimated posterior distribution $q(\mathbf{Z}^{\mathcal{V}_k}|\mathbf{H}^{\mathcal{V}_k}; \boldsymbol{\theta}^{\mathcal{V}_k})$ via Eq. (8);
  Project the observed space $\acute{\mathbf{H}}^{\mathcal{V}_k}$ of high-order relation of k-th type of node $\boldsymbol{\mathcal{V}^k}$, to a common latent space $\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}}$ via Eq. (9);
  Learn the importance of each kind of node in each hyperedge via Eq. (12);
  Fuse the type-specific hyperedge embedding $\widetilde{\mathbf{Z}}^{\mathcal{E}^{\mathcal{V}_k}}$ via Eq. (13);
  Sample the hyperedge embedding $\mathbf{Z}^{\mathcal{E}}$ from the estimated posterior distribution $q(\mathbf{Z}^{\mathcal{E}}|\mathbf{H}^{\mathcal{V}_1}, ..., \mathbf{H}^{\mathcal{V}_k}; \boldsymbol{\theta}^{\mathcal{E}})$ via Eq. (16);
  Reconstruct $\mathbf{H}^{\mathcal{V}_k}$ from the estimated distribution $p(\mathbf{H}_{i,j}^{\mathcal{V}_k}|\mathbf{Z}_i^{\mathcal{V}_k}, \mathbf{Z}_j^{\mathcal{E}}; \boldsymbol{\lambda}^{\mathcal{V}_k})$ via Eq. (18);
  Update the HeteHG-VAE with its stochastic gradient by Eq. (4).
end for
return $\mathbf{Z}^{\mathcal{V}}, \mathbf{Z}^{\mathcal{E}}$;

5 Experiments

In this section, we firstly describe the experimental setup including datasets, baseline methods, parameter settings and evaluation metrics. Then, we present the experimental results in comparison with the state-of-the-art methods on the link prediction task.

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 2: Statistics of Datasets. The density of graph is $\frac{2|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}|-1)}$. Identifier node is marked in **red** and slave node in **blue**.

Dataset	Node					Edge				Density	Note
	$\lvert\mathcal{V}_1\rvert$	$\lvert\mathcal{V}_2\rvert$	$\lvert\mathcal{V}_3\rvert$	$\lvert\mathcal{V}_4\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}_1\rvert$	$\lvert\mathcal{E}_2\rvert$	$\lvert\mathcal{E}_3\rvert$	$\lvert\mathcal{E}\rvert$		
DBLP	14,376 P	14,475 A	20 V	8,920 T	37,791	41,794 P-A	14,376 P-V	114,624 P-T	170,794	2.39E-4	P: Paper, A: Author V: Venue, T: Term
Douban	2,783 M	2,388 A	746 D	3,022 U	8,939	8,456 M-A	3,059 M-D	11,637 M-U	23,152	5.79E-4	M: Movie, A: Actor D: Director, U: User
IMDB	1,320 M	41,523 A	891 D	943 U	44,677	61,213 M-A	1,409 M-D	72,071 M-U	134,693	1.35E-4	M: Movie, A: Actor D: Director, U: User
Yelp	13,938 B	62 L	575 C	14,079 U	28,654	13,938 B-L	39,406 B-C	184,587 B-U	237,931	5.8E-4	B: Business, L: Location C: Category, U: User

5.1 Experimental Setup

5.1.1 Datasets

Four real-world plain networks [34] including DBLP, Douban, IMDB and Yelp are used in this paper to evaluate the proposed method. During the experiments, we use the identity matrix as the feature matrix of the node and hyperedge as in the previous works [36, 41, 43]. The statistics of datasets are shown in Table 2. Details of the datasets are as follows:

- **DBLP** is a bibliographic network in computer science collected from four research areas: Machine Learning, Database, Data Mining, and Information Retrieval. Four types of objects are included in the network: paper, author, venue, and term.
- **Douban** is a user-movie interest network from a user review website Douban in China[2], which contains four types of nodes including movies, directors, actors, and users.
- **IMDB** is also a user-movie interest network collected from the Internet Movie Data[3]. Four types of nodes including movies, directors, actors, and users, are contained in the network.
- **Yelp** is a user-business network collected from a website Yelp in America[4]. It contains four types of nodes including users, businesses, locations, and business categories.

5.1.2 Baseline Methods

We compare our proposed HeteHG-VAE with four kinds of graph representation learning models: homogeneous graph embedding models including DeepWalk [10], LINE [11], node2vec [12], and HSRL [34]; heterogeneous graph embedding models including HIN2Vec [18], metapath2vec [19], HEER [20], HGT [50], TransE [51], and ConvE [52]; graph neural network based models including SEAL [13] and HGCN [46], and graph autoencoders including GAE [36], VGAE [36], $\mathcal{S}$-VGAE [38], GAT-AE [43], Linear-AE [41], and Linear-VAE [41]. Details of these methods are as follows:

- **DeepWalk** [10] is a random walk based graph embedding method which conducts random walks on each node to sample node sequences and uses the Skip-Gram model to learn node embeddings by treating node sequences as sentences and nodes as words.
- **LINE** [11] defines the first-order and second-order proximities to measure the similarity between nodes, and learns node embeddings by preserving the proximities of nodes in the embedding space.
- **node2vec** [12] is a generalized version of DeepWalk that provides a trade-off between depth first and breadth first search for random walks on nodes.
- **HSRL** [34] recursively compresses an input network into a series of smaller networks using a community-awareness compressing strategy, and then learns node embeddings based on the compressed networks by using DeepWalk, LINE, or node2vec.
- **HIN2Vec** [18] exploits different types of relationships in forms of meta-paths in a heterogeneous graph by carrying out multiple prediction training tasks jointly based on a target set of relationships to learn latent vectors of nodes and meta-paths.
- **metapath2vec** [19] employs metapath based random walks on skip-gram model for heterogeneous graph embedding.
- **HEER** [20] studies the problem of comprehensive transcription for a heterogeneous graph by leveraging the edge representations and heterogeneous metrics for the final graph embedding.
- **TransE** [51] is a knowledge graph embedding model designed under the heterogeneous settings by treating each edge as a triplet composed of two nodes and an edge type, and then translating embeddings for modeling multi-relational data.
- **ConvE** [52] is similar to TransE but goes beyond simple distance or similarity functions by using the deep neural networks to score the triplet.
- **GAE/VGAE** [36] GAE is an autoencoder-based unsupervised model to learn both topologies and node contents of graph data, while VGAE is a variant of GAE, in which, variational autoencoder is utilized for network embedding.
- **$\mathcal{S}$-VGAE** [38] replaces Gaussian distribution of VGAE with von Mises-Fisher distribution to learning the embedding in the hyperspherical latent space.
- **GAT-AE** [43] GAT explores attention mechanisms among node neighbors for node classification. We replace the classifier of GAT with an inner product decoder to construct an autoencoder based model, termed as GAT-AE, for link prediction task.
- **Linear-AE/Linear-VAE** [41] are the linear version of GAE/VGAE by replacing the GCN encoder with a simple linear model w.r.t. the adjacency matrix of the graph.

2. https://movie.douban.com/
3. https://www.imdb.com/
4. https://www.yelp.com

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 3: Link Prediction Performance on DBLP and Douban. All values are percentages. The best results are marked in **bold**.

DBLP								
Training	20%		40%		60%		80%	
Metrics	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DeepWalk [10]	66.94±0.5	60.57±0.57	67.92±0.16	62.47±1.78	67.13±0.21	63.86±0.41	68.34±0.18	64.88±1.79
LINE [11]	70.95±1.26	72.27±0.66	72.73±1.24	74.03±1.49	78.46±0.31	79.79±1.65	79.69±0.9	80.47±1.01
node2vec [12]	69.78±1.08	69.12±0.61	71.21±0.3	70.57±1.22	71.4±0.71	72.01±1.25	72.36±1.15	73.11±1.42
HSRL [34]	71.0±0.29	69.72±0.41	73.34±0.59	71.9±1.78	73.53±0.48	72.28±1.48	74.78±0.12	73.04±1.51
HIN2Vec [18]	67.6±0.99	72.23±0.98	69.48±1.2	73.32±1.74	70.46±0.5	73.74±1.46	71.71±1.08	74.72±1.39
metapath2vec [19]	61.23±1.23	62.76±1.0	63.03±0.24	66.82±0.48	65.69±0.73	68.58±1.32	71.19±0.32	74.9±0.52
HEER [20]	77.81±1.81	79.23±1.55	78.86±2.97	78.69±2.81	80.42±1.3	83.22±2.35	82.91±1.27	85.38±1.82
TransE [51]	58.69±0.54	58.55±1.39	67.83±0.73	67.34±1.09	74.6±1.05	74.33±1.07	78.32±1.17	78.7±0.64
ConvE [52]	64.89±0.99	70.39±1.49	68.26±0.2	70.85±1.21	73.67±1.11	75.79±1.39	83.28±1.19	87.01±1.42
GAE [36]	69.65±0.14	73.06±1.42	76.73±0.82	77.94±1.55	79.91±1.27	81.68±0.96	83.35±1.31	85.8±1.24
VGAE [36]	76.04±1.34	77.55±0.94	77.82±0.88	78.81±1.66	79.53±1.04	81.16±1.4	84.41±0.79	86.16±1.74
$\mathcal{S}$-VGAE [38]	77.5±1.33	77.95±0.79	78.48±0.38	78.98±0.76	78.97±0.82	81.91±0.94	85.84±0.53	86.91±0.97
GAT-AE [43]	69.75±0.99	70.6±1.48	73.32±0.4	75.01±1.27	78.17±1.13	79.73±1.02	80.25±0.3	80.41±1.37
Linear-AE [41]	69.92±0.65	74.52±0.46	76.41±0.51	78.59±1.55	81.12±0.44	83.33±1.48	83.95±0.48	85.04±1.49
Linear-VAE [41]	70.27±0.59	73.77±1.63	76.32±0.6	77.34±1.67	82.16±1.28	83.11±0.77	85.16±0.74	84.89±1.07
SEAL [13]	76.71±0.41	76.2±0.31	79.18±0.35	78.51±0.62	83.6±0.42	82.15±1.01	87.11±0.31	86.04±0.69
HGCN [46]	75.61±0.64	76.76±1.33	80.12±0.26	81.8±1.05	83.92±1.7	84.74±0.6	85.79±1.43	86.5±1.32
HGT [50]	78.62±0.43	78.51±0.44	80.7±0.66	80.63±1.53	84.53±1.23	84.37±1.52	86.21±0.27	86.54±1.13
HeteHG-VAE	**84.04±0.39**	**86.33±0.77**	**85.20±1.22**	**87.23±0.73**	**87.92±0.14**	**89.34±0.74**	**89.82±0.26**	**90.82±0.42**
Douban								
Training	20%		40%		60%		80%	
Metrics	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DeepWalk [10]	62.59±0.29	57.4±0.99	63.32±0.73	59.69±1.38	64.15±0.47	60.26±1.52	65.04±1.13	61.36±0.84
LINE [11]	66.87±0.94	66.6±1.36	71.89±1.35	70.57±1.64	73.7±0.62	71.7±0.61	74.78±1.25	72.63±1.11
node2vec [12]	62.63±0.21	59.24±1.03	63.79±0.37	60.25±1.31	64.75±1.11	61.13±1.7	65.06±1.37	61.76±1.1
HSRL [34]	68.19±1.11	65.79±1.18	70.64±0.63	66.74±0.99	74.9±0.91	70.61±1.52	75.97±0.82	73.15±1.28
HIN2Vec [18]	66.61±0.29	61.36±0.65	67.38±0.95	62.36±1.54	68.49±0.47	63.86±0.91	69.39±0.36	65.31±0.67
metapath2vec [19]	64.31±0.2	63.95±1.31	66.73±0.44	64.77±1.15	67.18±1.37	65.6±0.9	68.17±0.79	66.22±1.5
HEER [20]	73.84±0.72	71.85±1.29	75.45±1.02	75.44±1.49	77.12±1.37	77.07±1.38	78.24±0.99	77.51±1.65
TransE [51]	63.04±0.3	61.1±0.85	69.1±0.88	66.72±0.96	69.67±0.21	66.94±1.38	69.93±0.78	67.03±1.49
ConvE [52]	68.06±0.7	66.6±1.58	69.06±0.52	68.6±0.81	71.06±0.43	70.6±0.44	72.25±0.31	70.77±0.58
GAE [36]	75.11±1.08	75.66±0.77	78.29±0.81	77.99±1.52	81.37±0.24	79.9±0.82	83.24±1.13	81.53±0.72
VGAE [36]	76.01±0.94	76.22±1.28	77.3±0.31	77.09±1.0	80.99±0.27	79.33±1.36	83.03±1.18	81.57±1.48
$\mathcal{S}$-VGAE [38]	75.51±0.39	75.42±0.98	76.24±0.43	76.4±1.12	79.45±0.52	78.53±1.03	81.3±1.41	80.55±1.23
GAT-AE [43]	66.47±1.31	63.84±0.66	69.34±0.75	66.15±1.01	71.24±0.98	67.54±0.93	73.87±1.27	70.76±0.76
Linear-AE [41]	76.1±0.44	76.03±1.22	77.84±0.82	77.58±1.01	80.19±0.2	80.12±1.75	82.21±0.84	81.06±0.63
Linear-VAE [41]	76.08±0.88	76.06±1.27	77.43±0.46	77.59±0.74	80.18±1.29	79.25±0.82	82.25±0.51	81.15±1.45
SEAL [13]	75.41±0.31	74.21±0.31	76.52±0.61	76.22±0.35	77.18±0.45	76.91±0.61	78.53±0.51	77.22±0.55
HGCN [46]	75.45±0.87	75.39±0.67	78.88±0.75	78.27±1.03	80.79±0.47	79.61±0.89	82.1±1.02	80.65±0.73
HGT [50]	76.62±0.46	75.31±0.25	77.62±0.52	76.63±0.44	78.75±0.55	78.48±0.82	79.83±0.74	79.36±0.46
HeteHG-VAE	**77.11±0.62**	**76.92±0.31**	**80.50±0.31**	**78.51±0.62**	**82.09±0.69**	**81.16±0.94**	**84.18±0.48**	**82.36±0.13**

- **SEAL** [13] is a link prediction framework, which extracts a local enclosing subgraph around each target link, and uses a graph classifier based on DGCNN [69] for the final link prediction.
- **HGCN** [46] is a hyperbolic graph neural network that employs GCNs with curvature learning layers to learn node embedding in the hyperbolic space.
- **HGT** [50] is a transformer based self-attention model for heterogeneous network embedding by designing node- and edge-type dependent parameters to characterize the heterogeneous attention over each edge.

5.1.3 Experimental Design and Metrics

In the experiment, we implement HeteHG-VAE based on Tensorflow[5] and train it with 200 iterations for DBLP, Yelp datasets, and 300 iterations for Douban and IMDB dataset respectively. Adam algorithm [70] is utilized for optimization with the learning rate as 0.001. The embedding size of all algorithms is set to 50 by default. Following the standard unsupervised network embedding settings, in the experiments, we randomly hide 20%-80% of existing edges as a test edge set, and train all models with the remaining edges, a held-out validation set (5% edges) is used to tune the hyper-parameters across all of the models. For all the models except SEAL and HGCN, we follow the strategy recommended by previous literature [12, 71] to train a logistic regression classifier for link prediction. We compute the feature representation of each edge by using the Hadamard operator to compute the element-wise product for embeddings of the linked target nodes, and use a classifier to predict link existence based on the computed edge representation. For SEAL, the learned subgraph classifier is directly applied on the same sampled test edges for evaluation. For HGCN, we use the Fermi-Dirac predictor [72, 73] as used in the original paper [46], to predict the edge probability. We repeat the process five times and the average performance and the mean variance are reported as the final results. Similar to the previous studies [13, 18, 34], in this paper, we use the widely used **AUC** score (the **A**rea **U**nder a receiver operating characteristic **C**urve) and **AP** score (**A**verage **P**recision) to measure the link prediction accuracy

5. https://github.com/tensorflow/tensorflow

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 4: Link Prediction Performance on IMDB and Yelp. All values are percentages. The best results are marked in **bold**.

IMDB								
Training	20%		40%		60%		80%	
Metrics	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DeepWalk [10]	60.42±0.7	56.07±1.43	61.95±0.94	57.05±1.77	64.23±0.69	58.5±0.73	66.3±1.19	60.41±0.9
LINE [11]	67.3±0.11	68.95±1.09	72.17±1.27	72.51±1.01	73.14±1.03	73.77±0.71	74.48±0.26	74.28±0.85
node2vec [12]	60.07±0.78	57.33±1.39	61.74±0.67	59.23±0.59	62.83±0.72	58.05±1.26	64.98±0.5	60.22±0.76
HSRL [34]	61.13±0.52	56.72±0.62	65.41±0.88	60.26±1.6	69.48±0.48	63.67±1.75	74.84±0.4	70.15±1.77
HIN2Vec [18]	60.9±0.99	59.75±0.77	61.68±1.21	62.51±0.52	64.49±0.26	63.97±1.61	65.07±1.29	64.83±1.02
metapath2vec [19]	60.5±1.04	63.25±1.64	61.88±1.15	64.06±0.68	62.65±0.3	65.23±1.65	66.19±1.19	67.69±1.37
HEER [20]	71.19±1.01	71.91±1.54	71.08±1.33	73.13±1.42	72.72±1.18	73.39±1.23	74.14±1.45	74.01±0.94
TransE [51]	61.92±0.46	64.7±0.92	63.12±0.87	65.29±1.64	63.43±1.29	66.15±1.4	64.28±1.22	67.62±1.28
ConvE [52]	66.38±0.33	65.33±0.51	67.1±0.67	66.26±1.38	69.25±0.12	68.34±1.77	70.08±0.41	69.98±1.57
GAE [36]	64.52±0.74	69.52±1.1	68.84±0.34	74.16±1.39	69.61±0.41	72.01±0.95	73.22±1.35	75.26±0.46
VGAE [36]	64.04±0.23	69.53±1.11	67.25±1.38	71.22±1.67	68.28±1.12	74.18±0.51	71.67±1.14	75.1±0.94
$\mathcal{S}$-VGAE [38]	66.1±0.92	69.45±1.21	69.22±0.71	72.33±0.95	70.32±0.83	73.53±1.02	72.56±0.94	77.01±0.8
GAT-AE [43]	64.6±0.56	63.96±0.67	68.73±1.11	67.09±0.83	67.59±1.35	65.92±1.65	72.52±0.1	68.98±0.61
Linear-AE [41]	64.93±0.94	71.22±1.52	66.67±0.66	72.44±0.96	67.74±0.59	73.16±0.4	69.88±1.18	73.14±1.2
Linear-VAE [41]	63.12±0.51	69.17±0.63	64.01±1.03	70.08±0.44	65.02±0.98	71.78±1.39	68.35±0.34	72.16±1.6
SEAL [13]	67.32±0.44	71.43±0.57	68.58±0.66	72.81±0.47	68.78±0.72	73.59±0.43	70.92±0.57	74.41±0.64
HGCN [46]	66.41±0.88	70.66±1.11	70.53±0.72	74.33±1.03	72.35±0.66	75.01±0.83	75.21±1.01	75.06±0.69
HGT [50]	70.53±0.72	72.56±0.56	72.41±0.63	73.52±0.66	74.59±0.57	74.53±0.39	76.44±0.49	75.42±1.01
HeteHG-VAE	**77.07±0.71**	**79.25±0.49**	**79.58±0.36**	**80.13±0.64**	**81.40±0.88**	**81.67±0.45**	**82.36±0.35**	**82.86±0.28**
Yelp								
Training	20%		40%		60%		80%	
Metrics	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DeepWalk [10]	75.03±0.29	67.49±0.96	76.63±1.4	68.78±1.56	77.61±0.65	68.98±1.03	78.04±0.46	69.67±0.99
LINE [11]	71.36±0.26	72.43±0.73	79.12±0.21	81.23±1.28	79.96±0.81	82.33±1.08	88.59±0.21	89.26±1.33
node2vec [12]	77.14±0.89	73.85±1.27	77.09±0.25	73.43±0.9	77.82±1.16	75.44±0.62	79.43±0.37	76.4±0.64
HSRL [34]	79.24±0.71	75.64±1.08	80.66±1.07	76.67±1.6	81.28±0.72	76.35±0.66	82.08±0.42	77.52±1.54
HIN2Vec [18]	77.17±0.38	74.0±1.58	78.17±1.32	75.77±1.62	78.97±1.33	76.79±1.7	79.51±0.79	77.17±1.45
metapath2vec [19]	66.53±0.46	69.98±0.55	69.77±1.02	72.22±1.74	70.67±0.82	73.89±1.28	71.04±0.18	74.76±0.56
HEER [20]	84.26±0.15	86.65±1.49	88.64±1.32	90.21±1.49	89.68±1.01	90.82±1.36	90.48±1.01	91.44±1.04
TransE [51]	68.04±0.65	70.47±0.79	78.22±0.33	79.68±1.67	80.71±0.12	81.32±1.42	83.5±0.8	83.72±1.27
ConvE [52]	77.82±0.62	82.2±1.52	78.82±1.19	83.2±0.67	82.39±1.31	84.44±0.7	84.08±0.98	85.92±1.34
GAE [36]	87.64±1.36	88.2±1.33	87.69±1.01	89.6±1.05	87.82±1.34	90.75±1.02	90.14±0.82	92.44±1.69
VGAE [36]	86.95±0.3	87.97±1.72	89.65±0.53	90.69±1.53	90.39±0.87	90.49±1.04	89.94±0.83	92.52±1.03
$\mathcal{S}$-VGAE [38]	85.49±1.13	86.39±1.37	88.36±1.05	88.46±1.35	89.22±0.84	89.54±0.82	89.04±0.68	91.25±0.93
GAT-AE [43]	77.99±0.88	77.84±0.71	82.07±0.72	83.51±1.56	84.53±0.44	85.71±1.17	85.56±0.74	86.09±0.84
Linear-AE [41]	78.87±0.67	84.18±1.77	85.33±1.13	88.06±1.44	87.94±0.48	91.33±1.13	90.14±1.37	92.0±1.1
Linear-VAE [41]	77.53±0.33	82.99±0.45	82.6±1.03	86.32±0.87	88.77±0.33	91.52±1.41	91.37±0.79	92.63±1.69
SEAL [13]	87.47±0.45	87.57±0.61	89.52±0.41	90.07±0.41	91.03±0.53	91.22±0.52	92.02±0.53	91.12±0.75
HGCN [46]	85.23±1.15	87.22±1.11	89.62±0.92	91.23±0.95	91.71±1.12	92.78±1.04	92.78±0.91	93.62±1.33
HGT [50]	89.11±0.25	88.69±0.54	90.84±0.34	90.51±0.66	91.89±0.75	92.14±0.47	92.85±0.41	91.88±0.46
HeteHG-VAE	**90.76±0.46**	**91.16±0.51**	**92.42±0.90**	**92.59±0.84**	**93.98±0.43**	**94.14±0.22**	**95.07±0.32**	**95.12±0.58**

for different models. All the experiments are conducted on the Ubuntu 16.04 Server with 189GB memory, and Intel(R) Xeon(R) CPU E5-2640 (2.4 GHz).

5.2 Experimental Results

5.2.1 Performance Analysis

In this section, we demonstrate the effectiveness of the proposed method by presenting the results of our model on the link prediction task, which is important in the real world to predict missing links or links that are likely to occur in the future in an information network, and provide a comparison with the state-of-the-art methods. The results of all compared methods on DBLP, Douban, IMDB, and Yelp datasets are provided in Table 3 and Table 4.

The experimental results show that the proposed HeteHG-VAE significantly outperforms all baselines across all the datasets. For example, with 20% training ratio, we observe that HeteHG-VAE improves AUC score and AP score over the best baseline by 6.89% (HGT) and 8.96% (HEER) on the DBLP dataset, and 8.25% (HEER) and 9.22% (HGT) on the IMDB dataset, respectively. With 80% training ratio, HeteHG-VAE improves AUC score and AP score over the best baseline by 3.11% (SEAL) and 4.37% (ConvE) on the DBLP dataset, and 7.74% (HGT) and 7.59% ($\mathcal{S}$-VGAE) on the IMDB dataset, respectively, which demonstrates that high-order relations and semantics are useful for pairwise link prediction in HINs. Among those baselines, DeepWalk, LINE, node2vec, and HSRL learn embedding by capturing various order proximity patterns of the network, while neglecting the heterogeneity of HINs. Heterogeneous based methods such as HEER, ConvE, and HGT instead make use of different types of relationships between nodes for more reasonable node embedding. Moreover, we find that SEAL, HGCN, and HGT achieve much better link prediction performance than other baselines, where SEAL trains a supervised link classifier based on the extracted local subgraph pattern around each target link, HGCN benefits from hyperbolic geometry, and HGT models heterogeneous edges according to the meta relation schema for better representation learning. Compared with the Euclidean GNN models such as GAE/VGAE, HGCN achieves better performance in most cases of all of the datasets except Douban. Since the performance gain of HGCN is correlated with graph hyperbolicity [46], it indicates that the underlying graph

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 5: Time efficiency performance (CPU time of model learning in seconds for one training epoch).

Dataset	DBLP	Douban	IMDB	Yelp
GAE [36]	17.7±0.2	1.01±0.1	24.8±0.3	10.1±0.2
VGAE [36]	16.2±0.3	0.93±0.1	27.1±1.3	9.3±0.2
GAT-AE [43]	45.7±1.4	2.02±0.2	151.4±10.1	18.3±1.3
Linear-AE [41]	16.1±0.9	0.98±0.1	21.1±0.2	9.57±0.3
Linear-VAE [41]	15.3±0.3	0.91±0.1	23.08±0.7	8.31±0.3
HeteHG-VAE	**3.7±0.08**	**0.21±0.01**	**0.69±0.02**	**2.4±0.05**

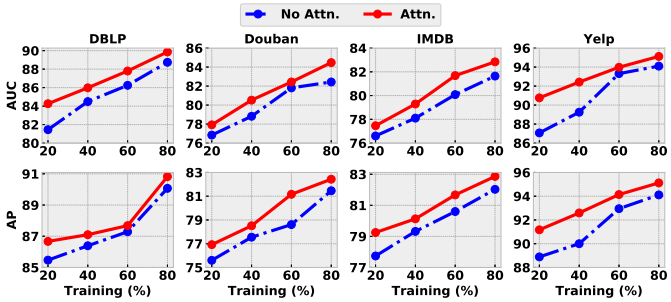

Fig. 4: Link prediction performance with/without hyperedge attention module.

structure of Douban is more Euclidean than hyperbolic.

Different from the baselines neglecting the high-order semantics and complex relations among different types of nodes, HeteHG-VAE considers both the different levels of relations among nodes and the heterogeneity of the network, as well as uses attention mechanisms to obtain more representative node embeddings. This explains its better performance on the link prediction task. Compared with five autoencoder based baselines, namely GAE, VGAE, GAT-AE, Linear-AE, and Linear-VAE, HeteHG-VAE achieves not only higher prediction accuracy, but also better model training efficiency. As shown in Table 5, compared with GAE and VGAE, HeteHG-VAE achieves a significant reduction of CPU time during training on all the four datasets. This is because of the simplicity of linear layer and the small size of input data used in HeteHG-VAE compared with those conventional graph autoencoders.

5.2.2 Effectiveness of Hyperedge Attention

As shown in Figure 4, we conduct an analysis on the effect of the hyperedge attention module. The link prediction performances of HeteHG-VAE with/without the hyperedge attention module on different datasets are reported, and the results demonstrate the effectiveness of hyperedge attention mechanism by considering the importance of different types of nodes. Moreover, we report the learned attention weights of different types of nodes (slave nodes) during the generation of the final hyperedge (identifier node) embedding in Figure 5. We find that those slave nodes with high degrees such as venue (V), director (D), and localization (L) tend to be associated with low attention weights because they are likely shared by an amount of identifier nodes, which plays a less representative role in representing an identifier node (hyperedge). Nodes with types such as term (T) and user (U) are less shared among hyperedges and are more discriminative to represent a specific identifier node, therefore, they are assigned with higher attention weights to represent a specific hyperedge.

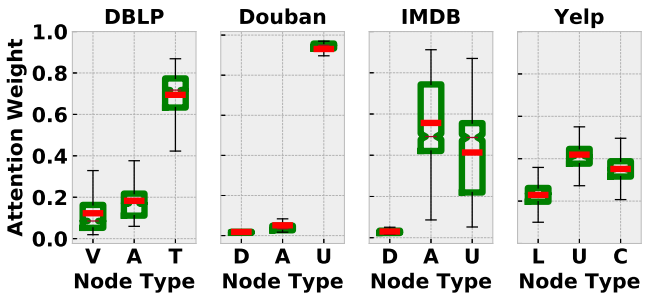

Fig. 5: Box plots of attention weights learned by HeteHG-VAE for different types of node.

5.2.3 Visualization

Different from the conventional heterogeneous graph embedding methods, the proposed HeteHG-VAE aims at learning a high-order relation aware node embedding for the link prediction task. To gain a better insight into the difference between HeteHG-VAE and conventional graph based methods, we visualize the learned node embedding via the t-SNE tool [74], which embeds the inferred node embedding into a two-dimensional space.

Specifically, we take the low-dimensional embeddings of nodes learned by three different methods, named DeepWalk, VGAE, and HeteHG-VAE, as the inputs to the t-SNE tool. Here, we randomly choose four target nodes, namely b1137, b5024, b5069, and b8602, and two sets of their neighbor nodes and non-neighbor nodes, respectively, to generate the node embeddings for visualization. As shown in Figure 6, the red triangle indicates the target node, the orange filled circles are the neighbors linked to it and the blue filled circles are randomly selected non-neighbors without links to it. As we can see, comparing with DeepWalk and VGAE, the neighbor nodes with links to the target node are clustered and separated better from those non-neighbor nodes that have no links to the target node by HeteHG-VAE, which demonstrates that high-order relation facilitates better node embedding to capture the structure and semantic proximity among nodes. The visualized results also explain why our approach is capable of achieving better performance on link prediction.

5.2.4 Embedding Dimension Sensitivity

We investigate the sensitivity of different numbers of the embedding dimension D for link prediction. The experiment results are shown in Figure 7. We can see that the performance is proportional to the size of embedding dimension until 64 or 128 because higher dimensional embeddings are capable of encoding more information. However, when the number of dimensions keeps on increasing, the performance tends to drop because the model might suffer from overfitting on the observed edges and thus performs poorly on new edges. Overall, the performance of HeteHG-VAE is relatively stable within a large range of embedding dimensions.

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

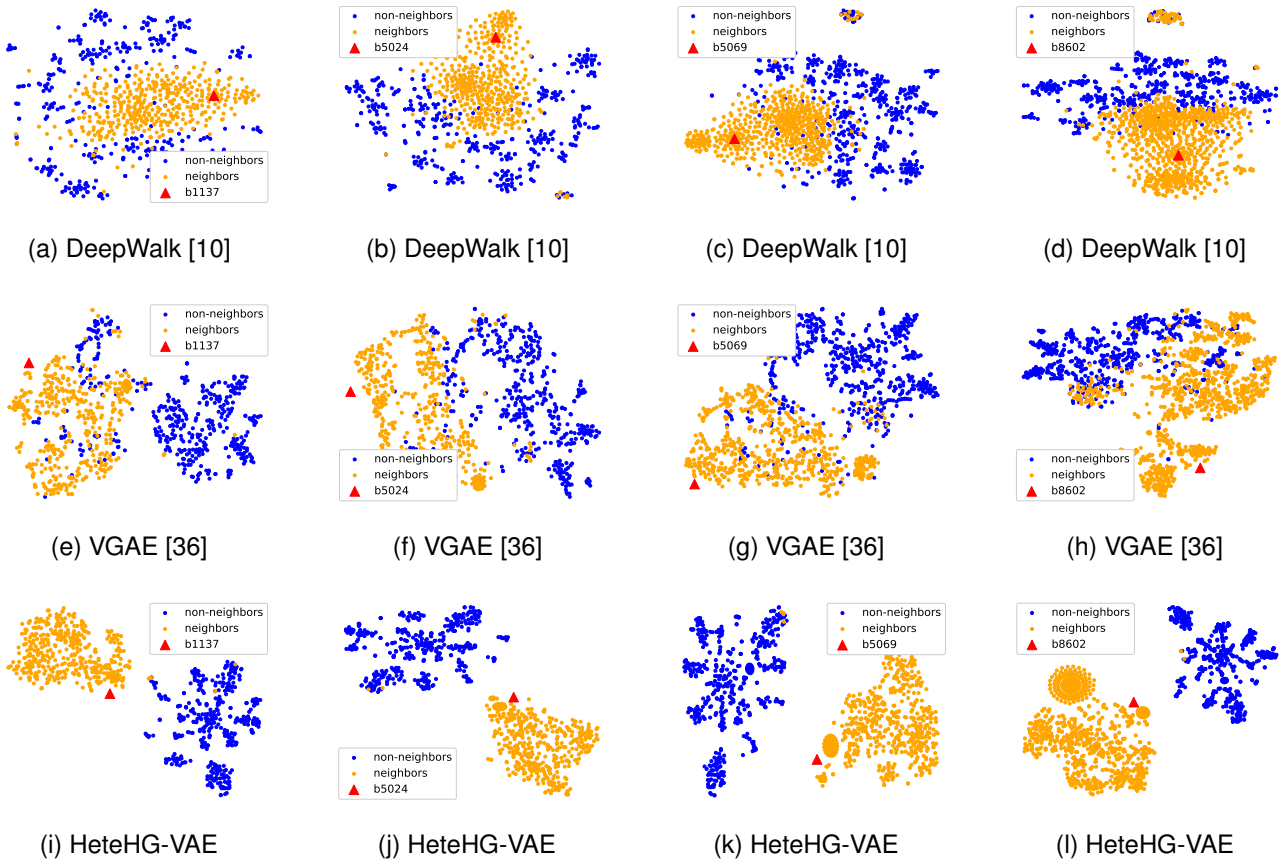

Fig. 6: 2D t-SNE visualization of the latent embeddings on the Yelp dataset. (**Orange** points and **blue** points indicate the neighbor nodes and the randomly sampled non-neighbor nodes of the **red** triangle shaped target node, respectively.)

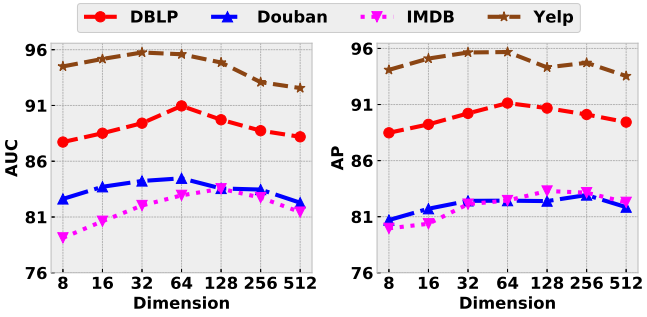

Fig. 7: Impact of different embedding dimensions.

6 Conclusion

In this paper, we study the problem of representation learning on the heterogeneous hypergraph for link prediction in heterogeneous information networks (HINs) and present a method named Heterogeneous Hypergraph Variational Autoencoder (HeteHG-VAE) to capture the high-order semantics and complex relations among different types of nodes in HINS. HeteHG-VAE first maps a conventional HIN to a heterogeneous hypergraph based on an event-induced semantics, which captures the high-order complex relation among nodes while preserving the low-order pairwise topology of the original HIN. Then, by using variational inference, it infers the stochastic distribution of the latent variables from the latent space rather than the observation space. HeteHG-VAE is also empowered by a hyperedge attention module that learns the importance of different types of nodes in each hyperedge for high-order semantic embedding. We show that HeteHG-VAE consistently outperforms previous state-of-the-art methods on link prediction on multiple challenging datasets in terms of link prediction accuracy and model training efficiency. In future works, we would like to explore a more complex mapping manner to construct multi-type hyperedges, which is beyond event-induced high-order semantics, and extend the proposed method to address the problem of multi-type hyperedge representation learning. We would also like to explore hypergraph learning on other non-Euclidean spaces, e.g., hyperbolic space, to investigate the structural properties of different hypergraphs and the geometric representation learning on the hypergraph.

References

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[2] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.

[3] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems (NIPS'13)*, 2013, pp. 926–934.

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

[4] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[5] A. Clauset, C. Moore, and M. E. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, no. 7191, pp. 98–101, 2008.

[6] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. 457–466, 2018.

[7] W. Feng and J. Wang, "Incorporating heterogeneous information for personalized tag recommendation in social tagging systems," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 1276–1284.

[8] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 357–370, 2018.

[9] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[11] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[12] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[13] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems (NIPS'18)*, 2018, pp. 5165–5175.

[14] S. Scellato, A. Noulas, and C. Mascolo, "Exploiting place features in link prediction on location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1046–1054.

[15] M. Nickel, X. Jiang, and V. Tresp, "Reducing the rank in relational factorization models by including observable patterns," in *Advances in Neural Information Processing Systems (NIPS'14)*, 2014, pp. 1179–1187.

[16] H. Zhao, L. Du, and W. Buntine, "Leveraging node attributes for incomplete relational data," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 4072–4081.

[17] H. Wang, X. Shi, and D.-Y. Yeung, "Relational deep learning: A deep latent variable model for link prediction," in *Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17)*, 2017.

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

[19] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 135–144.

[20] Y. Shi, Q. Zhu, F. Guo, C. Zhang, and J. Han, "Easing embedding learning by comprehensive transcription of heterogeneous information networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2190–2199.

[21] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 793–803.

[22] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, 2019, pp. 2022–2032.

[23] H. Gui, J. Liu, F. Tao, M. Jiang, B. Norick, L. Kaplan, and J. Han, "Embedding learning with events in heterogeneous information networks," *IEEE transactions on knowledge and data engineering*, vol. 29, no. 11, pp. 2428–2441, 2017.

[24] K. Tu, P. Cui, X. Wang, F. Wang, and W. Zhu, "Structural deep embedding for hyper-networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[25] I. M. Baytas, C. Xiao, F. Wang, A. K. Jain, and J. Zhou, "Heterogeneous hyper-network embedding," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 875–880.

[26] T. Chen and Y. Sun, "Task-guided and path-augmented heterogeneous network embedding for author identification," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 295–304.

[27] C. Zhang, C. Huang, L. Yu, X. Zhang, and N. V. Chawla, "Camel: Content-aware and meta-path augmented metric learning for author identification," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 709–718.

[28] D. Kingma and M. Welling, "Auto-encoding variational bayes," *2nd International Conference on Learning Representations (ICLR'14)*, 2014.

[29] L. Duan, S. Ma, C. Aggarwal, T. Ma, and J. Huai, "An ensemble approach to link prediction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2402–2416, 2017.

[30] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[31] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[32] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 635–644.

[33] Y.-L. Chen, M.-S. Chen, and S. Y. Philip, "Ensemble of diverse sparsifications for link prediction in large-scale networks," in *2015 IEEE International Conference on Data Mining*. IEEE, 2015, pp. 51–60.

[34] G. Fu, C. Hou, and X. Yao, "Learning topological representation for networks via hierarchical sampling," in *2019 International Joint Conference on Neural Networks (IJCNN'19)*. IEEE, 2019, pp. 1–8.

[35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[36] Kipf, Thomas N and M. Welling, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[37] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak, "Hyperspherical variational auto-encoders," 2018.

[38] Davidson, Tim R, Falorsi, Luca, De Cao, Nicola, T. Kipf, and J. M. Tomczak, "Hyperspherical variational auto-encoders," in *34th Conference on Uncertainty in Artificial Intelligence (UAI'18)*, 2018.

[39] S. Zhou, X. Wang, J. Bu, M. Ester, P. Yu, J. Chen, Q. Shi, and C. Wang, "Dge: Deep generative network embedding based on commonality and individuality." in *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI'20)*, 2020, pp. 6949–6956.

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

[41] G. Salha, R. Hennequin, and M. Vazirgiannis, "Keep it simple: Graph autoencoders without graph convolutional networks," in *Workshop on Graph Representation Learning, 33rd Conference on Neural Information Processing Systems (NeurIPS'19)*, 2019.

[42] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems (NIPS'17)*, 2017, pp. 1024–1034.

[43] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR'18)*, 2018.

[44] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML'19)*, 2019.

[45] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," in *Advances in Neural Information Processing Systems (NeurIPS'19)*, 2019, pp. 8230–8241.

[46] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *Advances in neural information processing systems (NeurIPS'19)*, 2019, pp. 4868–4879.

[47] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *Advances in neural information processing systems (NIPS'18)*, 2018, pp. 5345–5355.

[48] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings*

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3059313, IEEE Transactions on Pattern Analysis and Machine Intelligence

of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1165–1174.

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

[50] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020*, 2020, pp. 2704–2710.

[51] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems (NIPS'13)*, 2013, pp. 2787–2795.

[52] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI'18)*, 2018.

[53] C. Berge, *Graphs and Hypergraphs*. Oxford, UK, UK: Elsevier Science Ltd., 1985.

[54] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Advances in neural information processing systems (NIPS'07)*, 2007, pp. 1601–1608.

[55] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou, "Hypergraph learning: Methods and practices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[56] Y. Huang, Q. Liu, S. Zhang, and D. N. Metaxas, "Image retrieval via probabilistic hypergraph ranking," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3376–3383.

[57] H. Shi, Y. Zhang, Z. Zhang, N. Ma, X. Zhao, Y. Gao, and J. Sun, "Hypergraph-induced convolutional networks for visual classification," *IEEE transactions on neural networks and learning systems*, 2018.

[58] Z. Zhang, H. Lin, X. Zhao, R. Ji, and Y. Gao, "Inductive multi-hypergraph learning and its application on view-based 3d object classification," *IEEE Transactions on Image Processing*, vol. 27, no. 12, pp. 5957–5968, 2018.

[59] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux, "Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach," in *The World Wide Web Conference*. ACM, 2019, pp. 2147–2157.

[60] P. Li, G. J. Puleo, and O. Milenkovic, "Motif and hypergraph correlation clustering," *IEEE Transactions on Information Theory*, vol. 66, no. 5, pp. 3065–3078, 2019.

[61] I. E. Chien, H. Zhou, and P. Li, "hs^2: Active learning over hypergraphs with pointwise and pairwise queries," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 2466–2475.

[62] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI'19)*, 2019.

[63] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "Hypergcn: A new method for training graph convolutional networks on hypergraphs," in *33rd Conference on Neural Information Processing Systems (NeurIPS'19)*, 2019, pp. 1509–1520.

[64] P. Li and O. Milenkovic, "Inhomogeneous hypergraph clustering with applications," in *Advances in neural information processing systems (NIPS'17)*, 2017, pp. 2308–2318.

[65] P. Li and O. Milenkovic, "Submodular hypergraphs: p-laplacians, cheeger inequalities and spectral clustering," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 3014–3023.

[66] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2016.

[67] E. R. Scheinerman and D. H. Ullman, *Fractional graph theory: a rational approach to the theory of graphs*. Courier Corporation, 2011.

[68] M. Probst and F. Rothlauf, "Harmless overfitting: Using denoising autoencoders in estimation of distribution algorithms," *Journal of Machine Learning Research*, vol. 21, no. 78, pp. 1–31, 2020.

[69] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI'18)*, 2018.

[70] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR'15)*, 2015.

[71] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dynamic graph representation learning via self-attention networks," in *Workshop on Representation Learning on Graphs and Manifolds (ICLR'19)*, 2019.

[72] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *Physical Review E*, vol. 82, no. 3, p. 036106, 2010.

[73] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Advances in neural information processing systems (NIPS'17)*, 2017, pp. 6338–6347.

[74] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. 86, pp. 2579–2605, 2008.

0162-8828 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Canberra. Downloaded on May 22,2021 at 09:58:25 UTC from IEEE Xplore. Restrictions apply.