

Multi-head Variational Graph Autoencoder Constrained by Sum-product Networks

Riting Xia
College of Artificial Intelligence, Jilin University
Changchun, China
xiart19@mails.jlu.edu.cn

Yan Zhang
College of Communication Engineering, Jilin University
Changchun, China
yanz19@mails.jlu.edu.cn

Chunxu Zhang
College of Computer Science and Technology, Jilin University
Changchun, China
cxzhang@mails.jlu.edu.cn

Xueyan Liu*
College of Computer Science and Technology, Jilin University
Changchun, China
xueyanliu@jlu.edu.cn

Bo Yang*
College of Computer Science and Technology, Jilin University
Changchun, China
ybo@jlu.edu.cn

ABSTRACT

Variational graph autoencoder (VGAE) is a promising deep probabilistic model in graph representation learning. However, most existing VGAEs adopt the mean-field assumption, and cannot characterize the graphs with noise well. In this paper, we propose a novel deep probabilistic model for graph analysis, termed Multi-head Variational Graph Autoencoder Constrained by Sum-product Networks (named SPN-MVGAE), which helps to relax the mean-field assumption and learns better latent representation with fault tolerance. Our proposed model SPN-MVGAE uses conditional sum-product networks as constraints to learn the dependencies between latent factors in an end-to-end manner. Furthermore, we introduce the superposition of the latent representations learned by multiple variational networks to represent the final latent representations of nodes. Our model is the first use sum-product networks for graph representation learning, extending the scope of sum-product networks applications. Experimental results show that compared with other baseline methods, our model has competitive advantages in link prediction, fault tolerance, node classification, and graph visualization on real datasets.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; Artificial intelligence.

KEYWORDS

Variational Graph Autoencoders, Sum-product Networks, Graph Representation Learning.

*Corresponding Authors.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583517

ACM Reference Format:
Riting Xia, Yan Zhang, Chunxu Zhang, Xueyan Liu, and Bo Yang. 2023. Multi-head Variational Graph Autoencoder Constrained by Sum-product Networks. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583517

1 INTRODUCTION

Graphs (or complex networks) are ubiquitous in the real world, such as user-item networks [43], neural networks [27], negation normal form circuits [8, 42], and social networks [16, 39, 40], etc. A graph contains nodes and edges, in which a node denotes an individual and an edge refers to relationship between two nodes. As a general method for graph analysis, graph representation learning (GRL) [2, 35, 38] transforms the graph into low-dimensional dense vectors while preserving their attributes, which is convenient for downstream tasks, such as link prediction, node classification, and node clustering, etc [10, 34, 37].

Recently, variational graph autoencoder (VGAE) [15], as one of the deep probabilistic models, has become a promising research hotspot in GRL studying. Generally, VGAE contains two parts: an encoder and a decoder. The encoder transfers the graph into latent representations by obtaining the parameters of their posterior distributions (Gaussian distributions in common) and sampling them. The decoder reconstructs the topological structures of the graph using the latent representations. VGAE has the following advantages. (1) It is unsupervised, which is significant for those graphs that are difficult to collect labels [4]. (2) It has uncertainty as it models the probability distributions of variables. (3) It has the expressive power of deep neural networks by exploiting non-linearities. Due to the above advantages, researchers proposed lots of variants of VGAEs [1, 3, 7, 11, 12, 17, 18, 29] for learning better latent representations and used them in many downstream tasks. For example, generative adversarial networks (GANs) are incorporated into VGAE [23, 46] for learning robust and general latent representations. To adapt to various scenarios and downstream tasks, the Gaussian distribution is replaced with other distributions, such as mixture Gaussian distribution[3], Gamma distribution [29] and Student's t-distribution [1], etc. Additionally, some methods reconstruct both topological structures and node features [7, 11]

for comprehensively using graph information. For details on these methods, see Related Work (Section 2.1).

Although VGAEs have been studied for several years as mentioned above, they ignore some important information that describes the dependencies between the latent factors, i.e., elements of the latent representation of a node. To facilitate modeling and computation, existing VGAEs and their learning methods use the mean-field assumption [14], which assumes that the latent factors are independent of each other. However, this assumption is easily violated in the real-world data [31]. For example, for a social network in the Friends-and-Smokers domain, nodes represent people and edges represent their friend relationships. It is assumed that latent factors represent the attributes of the people, such as smoking, cancer, gender, age, skin color, occupation, etc. Existing research shows that there is dependency between cancer and smoking, as smoking increases the chance of developing cancer [33]. In this case, the mean-field assumption can lead to losing the dependency information. In addition, there is noise in the real-world graphs. For example, due to privacy protection, the users do not provide all attributes or provide wrong attributes in the social network of the Friends-and-Smokers domain. Most VGAEs ignore the presence of noise in the graphs, which leads to that these methods cannot characterize the graphs with noise well.

Sum-product networks (SPNs) [26] are emerging deep probabilistic models, and have attracted extensive attention because of the tractability of inference. As a variant of SPNs, conditional SPNs (CSPNs) [31], rooted directed acyclic graphs containing leaf, gating, and product nodes [1], can model the dependencies between latent variables. Since CSPNs are easy to integrate with deep neural networks, they also have powerful expressiveness that assist CSPNs model distributions of various data better. Existing work suggests that SPNs can deal with data with noise [24], and CSPN, as a variant of SPNs, inherits this advantage. However, CSPNs have not been used in graph data yet, since they are designed for the data follows independent identical distribution (i.i.d), and the nodes in graphs are dependent due to the existence of edges [2]. Thus, we consider using CSPN to help VGAEs better learn the posterior distribution of latent representations firstly.

To overcome the shortcomings of VGAEs, we propose a novel deep probabilistic model, namely Multi-head Variational Graph Autoencoder Constrained by Sum-product Networks (SPN-MVGAE). SPN-MVGAE can learn better and fault-tolerant latent representations by two ways. (1) SPN-MVGAE models the dependencies between latent factors using the CSPNs as the constraint term of the model. Specifically, we learn a conditional joint probability distribution of the latent representation conditional on the given graph through CSPN, and constrain the parameters of the encoder through this distribution so that the variational posterior distribution can learn the dependencies between latent factors learned by CSPN. (2) We also introduce multiple Gaussian distributions to represent the original graph data distribution, and then superpose the graph representations sampled from multiple Gaussian distributions to obtain the final latent representations. This method can form multiple subspaces of graph representations, making the model focus on different aspects of information.

The main contributions of this article are as follows:

- We propose a novel deep probabilistic model for GRL named as SPN-MVGAE, by incorporating CSPNs (one types of SPNs) into VGAEs, to relax the mean-field assumption of latent factors. To our best knowledge, it is the first time that dependencies between latent factors are considered in GRL. And it is also the first time that SPNs are considered in GRL.
- The final latent representations are obtained by introducing multiple inference networks and superimposing the obtained multiple latent representations. Multiple inference networks can learn richer information of nodes and edges, compared with a single inference network.
- We validate SPN-MVGAE on downstream tasks of GRL comprehensively, including link prediction, node classification and visualization. Also, we verify the fault tolerance of SPN-MVGAE. Experimental results show that SPN-MVGAE outperforms many state-of-the-art algorithms.

2 RELATED WORK

In this section, we introduce variational graph autoencoders and sum-product networks related to our work.

2.1 Variational Graph Autoencoders

VGAE is a representative model for GRL. Most of the current variants of VGAEs are proposed for adapting to various scenarios and downstream tasks by using different variational distribution, compared with the classical VGAE [15]. For example, Choong et al. [3] uses Gaussian mixture distribution to learn the posterior probability distribution of representations, which helps to detect communities in graphs. Aiming at explaining what latent factors are and why they perform well in graph modeling and generation, Li et al. [17] proposed the Dirichlet Graph Variational Audoencoder (DGVAE) method for graph generation and graph clustering tasks. Zheng et al. [46] proposed a distribution-induced bidirectional generative adversarial network named DBGAN to balance sample and distribution consistency through bidirectional adversarial learning, which improves generalization while maintaining representational power. Sarkar et al. [29] proposed a gamma staircase variational autoencoder, which models arbitrary nodes in the data with multiple layers of gamma-distributed latent variables. This method can learn the latent representations through the gamma distribution, which makes the latent representations interpretable and sparse. Besides, in order to learn more graph information, Guo et al. [7] proposed a method named MSVGAE, which learns multi-scale latent representations to represent the complex probability distribution. In the decoding stage, MSVGAE reconstructs the structure and features of the graph to make full use of the attribute information.

In addition to the above methods, Pan et al. [23] introduced generative adversarial learning [6] into VGAEs, where the latent representation matches the prior distribution through an adversarial training scheme to improve the performance of the learned latent representation. Jin et al. [11] proposed a method that considers the orthogonal relationship between node features and topology,

[1] For the specific definition and details of CSPNs, please refer to Section 3.
[2] It is worth noting that CSPNs model the dependencies between output variables, not for input data.

which utilizes different decoders in the decoding stage to reconstruct topology and node attributes. Mehta et al. [20] proposed a deep generative framework (DGLFRM) combining VGAE and stochastic block model (SBM) [9], which integrates graph neural networks (GNN) [41] and SBM into a VGAE framework for improving the interpretability and expression ability. Liu et al. [18] proposed a block-based learning method for disassortative graphs, which uses SBM to reconstruct the network.

Although there are many existing studies, they have some common shortcomings. These methods all exploit the mean-field assumption, which reduces the quality of the variational posterior distribution to approximate the real posterior distribution. In addition, most of the VGAEs cannot characterize the graphs with noise well. In general, these problems will make VGAEs produce inaccurate results on link prediction and node classification, as well as poor fault tolerance tasks.

2.2 Sum-product Networks

SPNs [26] have become a class of new promising probabilistic modeling method due to their expressive efficientcy [3] and tractable inference. SPNs can represent probability density by a computational graph that satisfies the validity property. Compared with the probabilistic graphical models, the inference of SPNs is tractable. Compared with deep neural networks, SPNs have a certain degree of interpretability, because each node in the computational graph of SPNs has probabilistic semantics.

SPNs have received extensive attention since they were proposed [26]. In recent years, SPNs have made significant progress in representation, inference, learning, and applications [28]. As a variant of SPNs, Shao et al. [31] proposed CSPNs, which can model dependencies between variables. CSPNs combine SPNs with a general function approximator, which makes CSPNs not only maintain tractability but also exploit the expressive power of general function approximators. For SPN structure learning, it is mainly divided into constructing SPN structure manually [26], learning SPN structure from data [5], constructing SPN structure randomly [24]. The random structure learning methods have the advantages of simple construction, strong scalability, and easy integration with deep learning, which expands the application scope of SPNs. There are currently two literatures that apply SPNs to graphs. Nath et al. [22] proposed a relational SPN named RSPN, which can be used to solve the link prediction task on graphs. Zheng et al. [45] proposed a graph-structured SPN named GraphSPN for modeling complex and noisy graph-structured data in robotics. However, the structure learning methods of RSPN and GraphSPN are cumbersome, difficult to tune, and have poor generalization ability. In general, SPNs are rarely used for tasks on graphs at present, and they have not been used for GRL yet.

In this paper, we consider using CSPN, a variant of SPN, to learn the dependencies between latent factors.

3 PRELIMINARIES

In order to provide important background information, we will introduce CSPN in this section.

[3] http://starai.cs.ucla.edu/slides/AAAI20.pdf.

Definition 3.1. (CSPNs). A CSPN $C = (\mathbf{N}, \mathbf{E}_c, \boldsymbol{\theta})$ over random variables (RVs) $\mathcal{X}$ and $\mathbf{Y}$ is a rooted directed acyclic graph, which charaterizes conditional distributions of RVs (i.e., $p(\mathbf{Y}|\mathcal{X})$), where $\mathbf{N}$ denotes node set, $\mathbf{E}_c$ denotes edge set, and $\boldsymbol{\theta}$ denotes the parameters of CSPN. There are four types of nodes in $\mathbf{N}$, which are a root node n_r, many leaf nodes $\mathbf{L}$, and many internal nodes consisting of gating nodes (also called sum nodes) $\mathbf{S}$, and product nodes $\mathbf{P}$, and $\mathbf{N} = n_r \cup \mathbf{L} \cup \mathbf{S} \cup \mathbf{P}$. The types of nodes are defined as follows:

(1) A leaf node n_l in $\mathbf{L}$ encodes a univariate conditional distribution $p(Y|\mathcal{X})$, where $Y \in \mathbf{Y}$, $\mathbf{Y}$ represents the target RVs sets, $\mathcal{X}$ represents the feature RVs sets, and $\mathcal{X} \cap \mathbf{Y} = \emptyset$.
(2) A product node n_p in $\mathbf{P}$ represents the product of the conditional probability distributions over its child nodes $ch(n_p)$, i.e. $p_{n_p} = \prod_{k=1}^{\mathcal{K}} p_k(\mathbf{Y}_k|\mathcal{X})$, where node k is the k^{th} child of n_p, $\mathbf{Y}_k \in \mathbf{Y}$, $\mathcal{K} = |ch(n_p)|$ is the number of child nodes.
(3) A gating node n_s in $\mathbf{S}$ represents the weighted sum of its child nodes $ch(n_s)$, i.e., $p_{n_s} = \sum_{j=1}^{J} g_{n_s,j} p_j(\mathbf{Y}|\mathcal{X})$, where node j is the j^{th} child of n_s, $g_{n_s,j}$ is the j^{th} output of a non-negative gating function $g(\mathcal{X}) = \{g_1(\mathcal{X}), g_2(\mathcal{X}), \ldots, g_J(\mathcal{X})\}$, and $\sum_i g_i(\mathcal{X}) = 1$; $J = |ch(n_s)|$ is the number of child nodes.
(4) The value of the root node n_r is calculated by evaluating its probability distribution P_{n_r}. The conditional probability distribution value of CSPN is denoted as $p_c = \frac{p_{n_r}}{Z}$, where Z is the normalization constant.

Neural CSPNs are an instantiation of CSPNs, specifically using deep neural networks as gating function and using random structure learning methods [24] to learn the structure of CSPNs. Based on the theoretical basis of CSPNs and neural CSPNs, we will use neural CSPNs to to design our method SPN-MVGAE.

4 METHODOLOGY

In this section, we first state the problem definition of GRL. Then, we elaborate on the SPN-MVGAE. Finally, the inference and learning methods of SPN-MVGAE are introduced.

4.1 Problem Definition of GRL

An undirected graph is defined as $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{F})$, which consists of node set $\mathbf{V} = \{v_1, \ldots, v_n\}$, feature set $\mathbf{F} = \{f_1, \ldots, f_m\}$, and edge set $\mathbf{E} = \{e_{ij} = (v_i, v_j) | v_i, v_j \in \mathbf{V}\}$, where n and m are the numbers of nodes and feature types, respectively. The edges are modeled by an adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$, where $a_{ij} = a_{ji} = 1$ if $e_{ij} \in \mathbf{E}$, otherwise $a_{ij} = 0$. The node features are characterized by a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, each row $\mathbf{x}_i$ of $\mathbf{X}$ is the feature of node v_i.

Given a graph $\mathbf{G}$, we aim to learn a latent representation matrix $\mathbf{Z} \in \mathbb{R}^{n \times d}$ that preserves graph information, where d represents the dimension of the latent representations, and i^{th} row vector $\mathbf{z}_i$ represents the latent representation of node v_i. It is worth noting that we assume that the latent factors z_{ij} and z_{ik} are dependent on each other for $\forall 1 \leq j, k \leq d$. The key notations that appear in the model are shown in Appendix A.1.

4.2 Overall Framework

Figure 1 shows the overall framework of SPN-MVGAE, which includes three parts: encoder, decoder, and constraint term CSPN. (1)

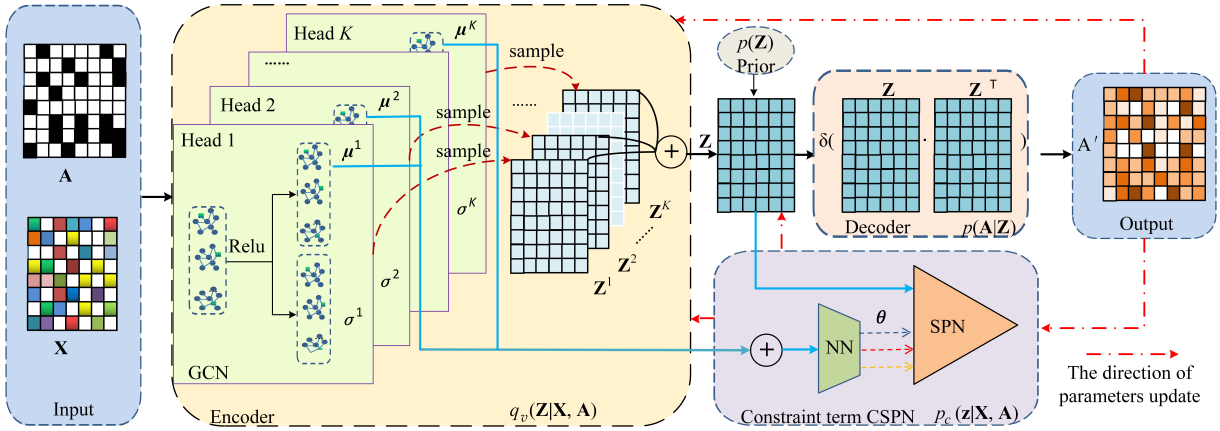


Figure 1: The architecture of SPN-MVGAE. A, X and A′ denote the adjacency, feature and reconstructed adjacency matrix, respectively. Z denotes the latent representation matrix. z denotes the set of target RVs, i.e., latent factors of representations. And $p_c(\mathbf{z}|\mathbf{X},\mathbf{A})$ denotes the conditional probability distribution.

The encoder transfers the topological structure and attributed information denoted by **A** and **X** to the latent representation **Z** using multiple GCNs. Each GCN produces the parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ of the variational posterior distributions of **Z**. Then, $\mathbf{Z}^j$ sampled from different variational posterior distributions are then superimposed to obtain the final **Z**. (2) The decoder reconstructs the graph **G** using the inner-product of **Z**. (3) Constraint term CSPN makes the latent factors dependent by introducing a conditional joint probability distribution to modify the learned **Z** by encoder.

4.3 The Proposed Model

The three parts (encoder, decoder, and constraint term CSPN) of SPN-MVGAE will be introduced in detail.

4.3.1 Encoder. In most existing VGAEs, the variational distribution learned by a single variational network cannot accurately represent the complex distribution of the data in the real world, which limits the inference network to approximate the true posterior distribution of the latent representations. Therefore, we design an encoder that uses multiple variational networks (i.e., multi-head) to learn better posterior variational distributions of the latent representation **Z**. This method can form multiple subspaces of **Z**, making the model focus on different aspects of information. Specifically, we assume **Z** follows Gaussian distribution with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Then, we take **A** and **X** as input to the encoder, and learn multiple parameters through multiple GCNs. Here, we use K GCNs. For j^{th} GCN, it produces $\boldsymbol{\mu}^j$ and $\boldsymbol{\sigma}^j$:

$$\boldsymbol{\mu}^j = \mathrm{GCN}_{\boldsymbol{\mu}^j}(\mathbf{X},\mathbf{A}), \tag{1}$$

$$log\boldsymbol{\sigma}^j = \mathrm{GCN}_{\boldsymbol{\sigma}^j}(\mathbf{X},\mathbf{A}). \tag{2}$$

The two-layer GCN is defined as:

$$\mathrm{GCN}(\mathbf{X},\mathbf{A}) = \bar{\mathbf{A}}\mathrm{ReLU}(\bar{\mathbf{A}}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1, \tag{3}$$

where $\mathbf{W}_0$ and $\mathbf{W}_1$ represent the weight matrices, $\mathrm{GCN}_{\boldsymbol{\mu}^j}(\mathbf{X},\mathbf{A})$ and $\mathrm{GCN}_{\boldsymbol{\sigma}^j}(\mathbf{X},\mathbf{A})$ share the first-layer parameters $\mathbf{W}_0$. $\mathrm{ReLU}(\cdot) = \max(0,\cdot)$ and $\bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix. **D** represents the degree matrix, where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. The variational posterior distribution inferred by j^{th} variational network can be expressed as:

$$q_v(\mathbf{Z}^j|\mathbf{X},\mathbf{A}) = \prod_{i=1}^n q_v(\mathbf{z}_i^j|\mathbf{X},\mathbf{A}), \tag{4}$$

$$q_v(\mathbf{z}_i^j|\mathbf{X},\mathbf{A}) = N(\mathbf{z}_i^j|\boldsymbol{\mu}_i^j, diag((\boldsymbol{\sigma}_i^j)^2)), \tag{5}$$

The latent representation **Z** is expressed as:

$$\mathbf{Z} = \frac{1}{K}\sum_{j=1}^K \mathbf{Z}^j, \tag{6}$$

where $\mathbf{Z}^j$ is sampled from $q_v(\mathbf{Z}^j|\mathbf{X},\mathbf{A})$.

4.3.2 Decoder. In the decoder stage, we reconstruct the adjacency matrix denoted by $\mathbf{A}'$:

$$p(\mathbf{A}'|\mathbf{Z}) = \prod_{i=1}^n \prod_{j=1}^n p(A'_{ij}|\mathbf{z}_i,\mathbf{z}_j), \tag{7}$$

$$p(a'_{ij} = 1|\mathbf{z}_i,\mathbf{z}_j) = \delta(\mathbf{z}_i^\top \mathbf{z}_j), \tag{8}$$

where a'_{ij} is the elements of $\mathbf{A}'$, and $\delta(\cdot)$ is the sigmoid function.

4.3.3 Constraint Term CSPN. SPN-MVGAE uses neural CSPN as a constraint term to relax the mean-field assumption. The parameters of the encoder are modified by constraint term, so that SPN-MVGAE can learn the dependencies between latent factors end-to-end. To overcome the shortcoming that original neural CSPNs can only deal with i.i.d. data, we introduce the GCN module (i.e., the encoder in SPN-MVGAE) in the gating function, which is used to model the dependencies between the nodes and learn the information of the nodes and edges on the graph. Specifically, we take **X**, **A** and $\mathbf{z}_i$ as input, and model the conditional joint probability distribution of $\mathbf{z}_i$ conditional on **A** and **X** through neural CSPN. The conditional probability distribution of the neural CSPN is obtained as

$$p_c(\mathbf{z}_i|\mathbf{X},\mathbf{A}) = p_c(\mathbf{z}_i;\boldsymbol{\theta} = g(\mathbf{X},\mathbf{A})). \tag{9}$$

where $\mathbf{z}_i = (z_{i1},\ldots,z_{id})^\top$ is the latent representation of node v_i.

As shown in Figure 2, the conditional probability is calculated from the bottom up of the triangular part. **A** and **X** are used as the input of the gating function to learn the parameters of the neural

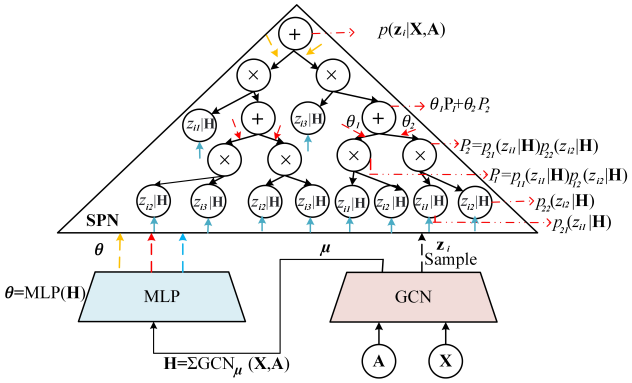


Figure 2: An instantiation of the constraint term CSPN for node i. + and × represent gating and product node, respectively. The dashed arrows of different colors represent the different data flows. z_{i1}, z_{i2}, and z_{i3} are the latent factors.

CSPN, and $\mathbf{z}_i$ is used as the input of leaf nodes. The leaf nodes of neural CSPN adopt Gaussian distribution. $\mathbf{H} = \sum_{j=1}^{K} \boldsymbol{\mu}^j$ is input into the multilayer perceptron (MLP) to obtain the parameters $\boldsymbol{\theta}$ of gating node weights and leaf node distributions:

$$\boldsymbol{\theta} = g(\mathbf{X}, \mathbf{A}) = \mathrm{MLP}(\mathbf{H}) = \mathrm{MLP}(\sum_{j=1}^{\mathrm{K}} \mathrm{GCN}_{\boldsymbol{\mu}^j}(\mathbf{X}, \mathbf{A})). \quad (10)$$

The superiority of using constraint term CSPN for VGAEs can be illustrated from five aspects. (1) Neural CSPN can capture the conditional dependencies between the output variables (i.e., the latent factors in GRL) through the product and gating nodes. (2) The inference of neural CSPN is tractable. (3) Neural CSPN can deal with the data with noise. (4) CSPN is easy to integrate with the deep learning frameworks (i.e., VGAEs in our paper), because it uses a random structure. However, many other SPNs are hard to integration into VGAEs, because they use data-based structures, which are opposed to the homogeneous structures employed in deep learning. (5) Our SPN-MVGAE needs to model a conditional distribution as a constraint term. CSPN can directly model conditional distributions, thus it is a natural fit for this purpose. However, other SPNs can only model joint distributions, thus they are unable to update the parameters of the encoder conditioned on $\mathbf{X}$ and $\mathbf{A}$.

4.4 Inference

Inference in our model consists of two parts, variational inference in the encoder part and tractable inference in the constraint term.

Similar to the classical VGAE, since the inference of posterior distribution is intractable, for the inference of the encoder part, we use variational inference to find the variational approximation distribution $q_v(\mathbf{Z}|\mathbf{X}, \mathbf{A})$ (as shown in Eq.(4 - 5)) of the true complex posterior distribution $p_v(\mathbf{Z}|\mathbf{X}, \mathbf{A})$.

The inference of constraint term CSPN refers to the calculation of $p_c(\mathbf{z}_i|\mathbf{X}, \mathbf{A})$, as shown in Eq.(9). Since the inference of neural CSPN is tractable, the probability distribution value can be directly calculated on the graph structure of neural CSPN, that is, multiplication is used when encountering a product node, and addition is used when encountering a gating node, from the bottom to top until the value of the root node is obtained.

4.5 Learning

The learning of our proposed model includes structure learning for neural CSPN and parameter learning for the whole model. We adopt random structure [24] as mentioned in Section 2.2 for structure learning. For parameter learning, we minimize the objective function in an end-to-end manner by stochastic gradient descent.

The overall objective function of SPN-MVGAE is divided into three parts: reconstruction term, KL divergence term, and constrained term, which can be expressed as:

$$\mathcal{L} = E_{q_v(\mathbf{Z}|\mathbf{X},\mathbf{A})}[logp_v(\mathbf{A}'|\mathbf{Z})] - \alpha * \mathrm{KL}[q_v(\mathbf{Z}|\mathbf{X}, \mathbf{A})||p(\mathbf{Z})] - \beta * \frac{1}{n}\sum_{i=1}^{n} logp_c(\mathbf{z}_i; \boldsymbol{\theta} = g(\mathbf{X}, \mathbf{A})). \quad (11)$$

where $E_{q_v(\mathbf{Z}|\mathbf{X},\mathbf{A})}[logp_v(\mathbf{A}'|\mathbf{Z})]$ represents the distance measurement between the generated topology graph and the original topology graph, $KL[q_v(\mathbf{Z}|\mathbf{X}, \mathbf{A})||p(\mathbf{Z})]$ represents a measure of the difference between $p(\mathbf{Z})$ and $q_v(\mathbf{Z}|\mathbf{X}, \mathbf{A})$. We use Gaussian prior $p(\mathbf{Z}) = \prod_{i=1} N(\mathbf{z}_i|0, \mathbf{I})$ as prior knowledge, where $\mathbf{I}$ represents the identity matrix. $\frac{1}{n}\sum_{i=1}^{n} logp_c(\mathbf{z}_i; \boldsymbol{\theta} = g(\mathbf{X}, \mathbf{A}))$ represents constraints on model parameters. α and β represent hyper-parameters.

We update $\boldsymbol{\theta}$ by maximizing the conditional probability distribution $p_c(\mathbf{z}_i|\mathbf{X}, \mathbf{A})$, and update the parameters of the MLP and encoder (i.e., GCNs) by the chain rule. In addition, the parameters of the neural CSPN (including the parameters of encoder and MLP) are constrained by the reconstruction term to ensure that the neural CSPN learns a better conditional joint probability distribution.

5 EXPERIMENTAL RESULTS

To verify the effectiveness of our proposed model SPN-MVGAE[4], we validate SPN-MVGAE on four tasks: link prediction, fault tolerance, node classification, and graph visualization. Furthermore, to test the functions of the different modules of our proposed model, we also set up ablation experiments.

5.1 Evaluation Setup and Metrics

5.1.1 Datasets. We evaluate our proposed model using the following three datasets widely used for graph analysis, where the nodes of the datasets all represent papers and the edges all represent citation relationships.

- **Pubmed** [21] consists of 44,338 links and 19,717 diabetes-related papers. These publications contain 3 categories. Each publication consists of 500 features.
- **Cora** [19] consists of 5429 links and 2708 scientific publications. These publications contain 7 categories. Each publication consists of 1433 features.
- **Citeseer** [30] consists of 4732 links and 3327 scientific publications. These publications contain 6 categories. Each publication consists of 3703 features.

[4]https://github.com/godlovexiari/MVGAE-SPN

Table 1: Results for link prediction task.

Methods	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
Spectral Clustering	84.6±0.01	88.5±0.00	80.5±0.01	85.0±0.01	84.2±0.02	87.8±0.01
DeepWalk	83.1±0.01	85.0±0.00	80.5±0.01	83.6±0.01	84.4±0.00	84.1±0.00
GAE	91.0±0.01	92.0±0.03	89.5±0.04	89.9±0.05	96.4±0.00	96.5±0.00
VGAE	91.4±0.00	92.6±0.01	90.8±0.02	92.0±0.02	94.4±0.02	94.7±0.02
ARGA	92.4±0.03	93.2±0.03	91.9±0.03	93.0±0.03	96.8±0.01	97.1±0.01
ARVGA	92.4±0.04	92.6±0.04	92.4±0.03	93.0±0.03	96.5±0.01	96.8±0.01
EVGAE	93.0±0.02	93.6±0.03	91.6±0.03	93.2±0.02	96.8±0.01	96.9±0.02
DBGAN	94.5±0.01	95.1±0.05	94.5±0.04	95.8±0.01	96.8±0.01	97.3±0.02
MSVGAE	**95.3±0.05**	95.4±0.04	95.4±0.03	96.1±0.04	-	-
SPN-VGAE	94.1±0.03	94.7±0.01	94.4±0.04	95.4±0.02	96.8±0.02	97.2±0.01
SPN-MVGAE	94.6±0.02	**95.6±0.02**	**95.5±0.02**	**96.3±0.01**	**97.0±0.00**	**97.5±0.01**

5.1.2 Evaluation Metrics. We mainly adopt link prediction and semi-supervised node classification tasks to verify the latent representations learned by SPN-MVGAE. For the link prediction task, AUC and AP value [15] are used as evaluation metrics. Higher values of AUC and AP denote better performance of the methods. For the node classification task, we use accuracy as evaluation metrics. We obtain the class of the nodes by connecting the latent representation $\mathbf{Z}$ to a fully connected layer, and calculate the difference between the predicted label and the true label through the cross-entropy loss function. The detailed definitions of evaluation metrics are described in Appendix A.2

5.1.3 Baselines. We select two types baselines related to GRL algorithms, especially those that have achieved the state-of-the-art results recently. (1) The models only utilize the graph structure. (2) The models utilize both graph structure and node attribute, especially VGAE and its variants, because our method focuses on overcoming the shortcomings of VGAEs.

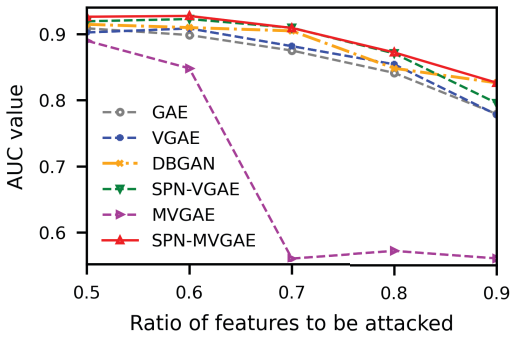
- **DeepWalk** [25]: It is a method of encoding a graph into a latent representation through a random walk method, which only utilizes the graph structure.
- **Spectral Clustering** [36]: It is also an efficient way to learn social latent representations using only graph structures.
- **GAE** [15]: It is an autoencoder based on graph, the method utilizes both graph structure and node attribute information.
- **VGAE** [15]: It is a variational autoencoder based on graph, which belongs to the category of deep probabilistic models.
- **ARGA** [23]: It is a variant of the GAE, which introduces an adversarial regularization term based on GAE.
- **ARVGA** [23]: Similar to ARGA, this method introduces an adversarial regularization term based on VGAE.
- **EVGAE** [13]: It is also a variant of VGAE, which is mainly used to solve the over-pruning problem.
- **DBGAN** [46]: It improves VGAE through distribution-induced bidirectional generative adversarial networks.
- **MSVGAE** [7]: It is a multi-scale VGAE that uses multiple sets of low-dimensional vectors of different dimensions to represent mixed probability distributions.

In addition, to verify the validity of the CSPN constraint term of SPN-MVGAE, we design a variant of SPN-MVGAE, the VGAE based on CSPN constraints (SPN-VGAE), as an ablation experiment.

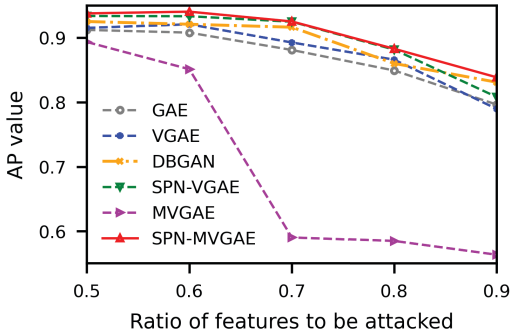
5.1.4 Parameter Settings. The hyperparameters are set by gird search. We optimize the parameters with the Adam optimizer and set the learning rate to 0.01. α and β are set to 0.003 and 0.004, respectively. For link prediction, the dimensions of the latent representations are 32. The dataset segmentation is the same as in reference [46]. For the Cora, Citeseer, and Pubmed, the number of iterations is set to 50, 50, and 600, respectively. In order to avoid the influence of randomness, we take the mean and standard deviation after 5 training sessions as the final score. For semi-supervised node classification, like [44], we randomly sample 5, 10, and 20 labels in each class for training. The dimensions of the latent representations are set to 16. The hidden layer dimensions of other contrasting methods are set to 32 and 16. The number of iterations is set to 50. We take the average of 10 runs as the final result. The parameter settings of all baselines are the same as the original paper.

5.2 Link Prediction

The experimental results of different models on the link prediction task are shown in Table 1. The results of MSVGAE are taken from the original paper. The values in bold are the best results among the SPN-MVGAE and the baselines. We can conclude the following observations. (1) SPN-MVGAE achieves the best performance on all three datasets, except that the AUC value on the Cora achieves comparable results to current methods (AUC is 0.007 lower than MSVGAE, but higher than DBGAN). This is because the scale of Cora are relatively small, which leads to learning non-optimal structure of CSPN. (2) For the SPN-VGAE, which only introduces CSPN constraints, all AP and AUC scores are higher than VGAE on the three datasets, second only to DBGAN. This illustrates the effectiveness of our neural CSPN module. (3) Compared with SPN-VGAE, the link prediction accuracy of SPN-MVGAE is improved after introducing multiple variational networks. This demonstrates the effectiveness of multiple variational networks. (4) Compared with Spectral Clustering and DeepWalk that only use graph structure, the model that uses both graph structure and node attribute can effectively improve the performance of link prediction task.

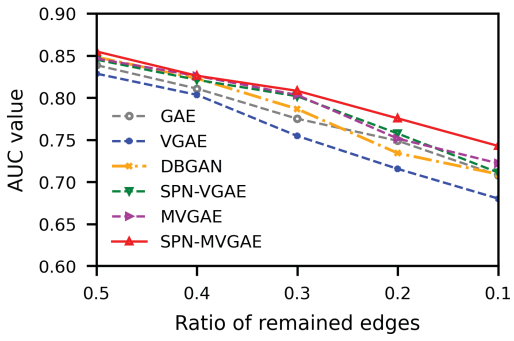


(a) AUC

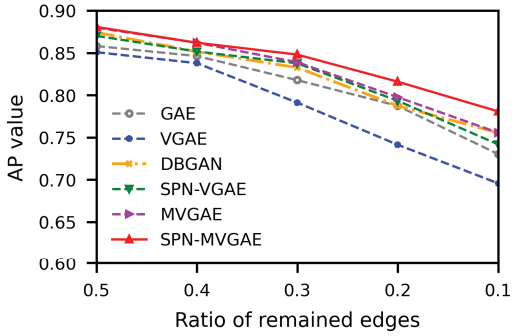


(b) AP

Figure 3: AUC and AP values for feature attack.



(a) AUC



(b) AP

Figure 4: AUC and AP values for edge attack.

5.3 Fault Tolerance

To verify the fault tolerance of SPN-MVGAE, we randomly attack the node features and edges, respectively. Specifically, we add noise with different proportions to node features and reducing the number of edges by a certain proportion. Then, we compare SPN-MVGAE with GAE, VGAE, DBGAN algorithms and the variants SPN-VGAE and MVGAE (i.e., VGAE based on multiple variational networks) of our proposed model. To ensure the timeliness of the algorithms, the Cora dataset is selected for testing in the following experiments.

5.3.1 Attack Features. We attack node features by randomly change original features to noisy features of all nodes in training set. Specifically, the noisy features are generated from a uniform distribution ranging from 0 to 0.2. The ratio of noisy features to all features ranges from 0.5 to 0.9 by step 0.1. Figure 3 shows the experimental results of link prediction under feature attack.

In Figure 3, as the ratio of noisy features increases, the performance of link prediction for all models tends to decline, but SPN-MVGAE consistently outperforms the other comparison methods. In addition, SPN-VGAE outperforms GAE and VGAE, and is competitive with DBGAN. MVGAE performs poor when the feature ratio of attacks reaches 0.7, both AUC and AP values are lower than 0.6. Although MVGAE learns more information by multiple variational networks, it also integrates more noise, leading to worse performance as the noisy features increase. The comparison between the performance of SPN-VGAE and MVGAE demonstrates that neural CSPN module can tolerate more noisy features than multiple variational networks. However, SPN-MVGAE containing both two modules outperforms SPN-VGAE and MVGAE, it indicates that with the help of neural CSPN module, the multiple variational networks will not integrates so much noise when learning more information that assist link prediction.

5.3.2 Attack Edges. In order to verify the performance of SPN-MVGAE when the edges are noisy, we reduce the ratio of the number of edges from 0.5 to 0.1 by step 0.1 in training set. As we can see in Figure 4, SPN-MVGAE performs best compared with baselines. The performance of the VGAE is poor as the edges reduce. Compared VGAE, SPN-VGAE, MVGAE, and SPN-MVGAE all perform better. It indicates that both multiple variational networks and neural CSPN modules are both useful for link prediction when the edges are reduced, because neural CSPN makes SPN-VGAE robust and the multiple variational networks can learn more information abundantly through multiple variational networks.

5.4 Ablation Study

Multiple variational networks (named MNs) in SPN-MVGAE can be used as a model-agnostic plug-in model. To verify the effectiveness of the MNs module and the effect of each module on VGAE promotion, we performed ablation experiments with GAE, VGAE, and SPN-MVGAE on Cora and Citeseer, where GAE+MNs and



(a) VGAE (b) ARVGE (c) DBGAN (d) SPN-VGAE (e) SPN-MVGAE

Figure 5: Visualization of the Citeseer dataset.

Table 2: Ablation experiments on Cora and Citeseer in terms of link prediction.

Methods	Cora		Citeseer	
	AUC	AP	AUC	AP
GAE	91.0±0.01	92.0±0.03	89.5±0.04	89.9±0.05
GAE+MNs	**93.7±0.01**	**94.0±0.01**	**94.5±0.02**	**95.1±0.03**
VGAE	91.4±0.00	92.6±0.01	90.8±0.02	92.0±0.02
VGAE+MNs	**92.8±0.01**	**93.8±0.01**	**92.1±0.02**	**93.6±0.02**
SPN-VGAE	94.1±0.03	94.7±0.01	94.4±0.04	95.4±0.02
SPN-MVGAE	**94.6±0.02**	**95.6±0.02**	**95.5±0.02**	**96.3±0.01**

Table 3: Results for classification task.

Methods	Cora			Citeseer		
	5 L	10 L	20 L	5 L	10 L	20 L
GAE	81.1	82.5	82.9	68.8	69.5	68.3
VGAE	81.8	82.2	82.3	68.1	69.1	70.2
SPN-VGAE	**82.3**	82.4	82.4	**69.8**	**69.9**	**70.5**
SPN-MVGAE	82.2	**82.5**	**83.2**	69.5	69.6	70.2

VGAE+MNs (i.e., MVGAE) represent the GAE and VGAE based on the MNs module.

The results of the ablation experiments are shown in Table 2. The link prediction performance of VAE+MNs, VGAE+MNs, and SPN-MVGAE is better than that without MNs module. It demonstrates that the MNs module can assist the model to achieve better performance. In addition, the experimental results show that compared with VGAE based on MNs module (MVGAE), SPN-VGAE has more obvious improvement effect on VGAE.

5.5 Node Classification

We compare GAE, VGAE, SPN-VGAE, and SPN-MVGAE on the Cora and Citeseer. The experimental results are shown in Table 3. L represents the number of labels that can be obtained for each class. The results show that SPN-MVGAE and SPN-VGAE outperform the other two baselines on the node classification task at different scales of labels. Compared with GAE and VGAE, the classification accuracy of the proposed methods is competitive. Compared with GAE, SPN-VGAE achieves about 1% improvement in classification accuracy on both datasets with only 5 labels, which shows our methods improves the classification accuracy more obviously in the case of fewer labels.

5.6 Graph Visualization

To better show the latent representations obtained by SPN-MVGAE and baselines, we apply the t-SNE algorithm [32] on the latent representations to visualize the Citeseer. Specifically, we visualize the latent representations for dimensionality reduction to a two-dimensional space. The perplexity and the number of iterations of t-SNE are set to 30 and 3000, respectively. Figure 5 shows the visualization results. From the visualization results, SPN-MVGAE has advantages. The classic VGAE and its variants ARVGA and DBGAN cannot distinguish different types of nodes in the latent space very well. The common feature of these methods is that the center part overlaps more, and the nodes within the class are loose. Because VGAE, ARVGA, and DBGAN all use the standard normal distribution as a prior, the centers of all categories are moved closer to the origin, resulting in the inability to make full use of the latent space. Our proposed method alleviates the center clustering problem by relaxing this strong prior assumption. Moreover, experimental results show that the intra-class nodes are denser of our proposed model.

6 CONCLUSION

In this paper, we propose a novel deep probabilistic generative model for GRL, i.e., Multi-head Variational Graph Autoencoder Constrained by Sum-product Networks (named SPN-MVGAE). It can learn the dependencies of latent factors through the constraint CSPN. In addition, multiple latent representations are obtained through multiple variational networks. And these latent representations are superimposed to represent the final latent representations, which results in a better graph representations to represent the original data. We conduct extensive experiments on four tasks, and the experimental results show that our model SPN-MVGAE learns better latent representations than other baselines.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China under Grant Nos. 2021ZD0112501 and 2021ZD0112502; the National Natural Science Foundation of China under Grant Nos. U22A2098, 62172185, 62202200 and 61876069. We would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] Najmeh Abiri and Mattias Ohlsson. 2019. Variational auto-encoders with Student's t-prior. In *ESANN*. 415–420.
[2] Jinheon Baek, Minki Kang, and Sung Ju Hwang. 2021. Accurate Learning of Graph Representations with Graph Multiset Pooling. In *ICLR*. 1–22.

[3] Jun Jin Choong, Xin Liu, and Tsuyoshi Murata. 2018. Learning Community Structure with Variational Autoencoder. In *ICDM*. 69–78.

[4] Thilini Cooray and Ngai-Man Cheung. 2022. Graph-Wise Common Latent Factor Extraction for Unsupervised Graph Representation Learning. In *AAAI*. 6420–6428.

[5] Robert Gens and Pedro Domingos. 2013. Learning the Structure of Sum-Product Networks. In *ICML*. 873–880.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. 2672–2680.

[7] Zhihao Guo, Feng Wang, Kaixuan Yao, Jiye Liang, and Zhiqiang Wang. 2022. Multi-Scale Variational Graph AutoEncoder for Link Prediction. In *WSDM*. 334–342.

[8] Pascal Hitzler and Md. Kamruzzaman Sarker (Eds.). 2021. *Neuro-Symbolic Artificial Intelligence: The State of the Art*. Frontiers in Artificial Intelligence and Applications, Vol. 342. IOS Press, Amsterdam, Berlin, Washington, DC.

[9] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social networks* 5, 2 (1983), 109–137.

[10] Hong Huang, Yu Song, Yao Wu, Jia Shi, Xia Xie, and Hai Jin. 2022. Multitask Representation Learning With Multiview Graph Convolutional Networks. *IEEE Transactions on Neural Networks and Learning Systems* 33, 3 (2022), 983–995.

[11] Di Jin, Bingyi Li, Pengfei Jiao, Dongxiao He, and Weixiong Zhang. 2019. Network-Specific Variational Auto-Encoder for Embedding in Attribute Networks. In *IJCAI*. 2663–2669.

[12] Rayyan Ahmad Khan, Muhammad Umer Anwaar, and Martin Kleinsteuber. 2020. Epitomic Variational Graph Autoencoder. In *ICPR*. 7203–7210. https://doi.org/10.1109/ICPR48806.2021.9412531

[13] Rayyan Ahmad Khan, Muhammad Umer Anwaar, and Martin Kleinsteuber. 2020. Epitomic Variational Graph Autoencoder. In *ICPR*. 7203–7210.

[14] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*. 14–16.

[15] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS*, 1–3.

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. 1–14.

[17] Jia Li, Jianwei Yu, Jiajin Li, Honglei Zhang, Kangfei Zhao, Yu Rong, Hong Cheng, and Junzhou Huang. 2020. Dirichlet Graph Variational Autoencoder. In *NeurIPS*. 1–12.

[18] Xueyan Liu, Bo Yang, Wenzhuo Song, Katarzyna Musial, Wanli Zuo, Hongxu Chen, and Hongzhi Yin. 2021. A block-based generative model for attributed network embedding. *World Wide Web* 24, 5 (2021), 1439–1464.

[19] Qing Lu and Lise Getoor. 2003. Link-based Classification. In *ICML*. 496–503.

[20] Nikhil Mehta, Lawrence Carin, and Piyush Rai. 2019. Stochastic Blockmodels meet Graph Neural Networks. In *ICML*, Vol. 97. 4466–4474.

[21] Galileo Namata, Ben London, Lise Getoor, and Bert Huang. 2012. Query-driven Active Surveying for Collective Classification. In *MLG*. 1–8.

[22] Aniruddh Nath and Pedro Domingos. 2015. Learning Relational Sum-Product Networks. In *AAAI*. 2878–2886.

[23] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *IJCAI*. 2609–2615.

[24] Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Xiaoting Shao, Kristian Kersting, and Zoubin Ghahramani. 2019. Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning. In *UAI*. 334–344.

[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD*. 701–710.

[26] Hoifung Poon and Pedro Domingos. 2011. Sum-Product Networks: A New Deep Architecture. In *UAI*. 337–346.

[27] Angelika Romanou, Panayiotis Smeros, and Karl Aberer. 2021. On Representation Learning for Scientific News Articles Using Heterogeneous Knowledge Graphs. In *WWW*. 422–425.

[28] Raquel Sánchez-Cauce, Iago París, and Francisco Javier Díez. 2022. Sum-Product Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 7 (2022), 3821–3839.

[29] Arindam Sarkar, Nikhil Mehta, and Piyush Rai. 2020. Graph Representation Learning via Ladder Gamma Variational Autoencoders. In *AAAI*. 5604–5611.

[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Mag.* 29, 3 (2008), 93–106.

[31] Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. 2022. Conditional sum-product networks: Modular probabilistic circuits via gate functions. *Int. J. Approx. Reason.* 140 (2022), 298–313.

[32] Xianjun Shen, Xianchao Zhu, Xingpeng Jiang, Li Gao, Tingting He, and Xiaohua Hu. 2017. Visualization of non-metric relationships by adaptive learning multiple maps t-SNE regularization. In *IEEE BigData*. 3882–3887.

[33] Parag Singla and Pedro Domingos. 2008. Lifted First-Order Belief Propagation. In *AAAI*. 1094–1099.

[34] Arjun Subramonian. 2021. MOTIF-Driven Contrastive Learning of Graph Representations. In *AAAI*. 15980–15981.

[35] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*. 1–16.

[36] Lei Tang and Huan Liu. 2011. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery* 23, 3 (2011), 447–478.

[37] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L. Dyer, Rémi Munos, Petar Velickovic, and Michal Valko. 2022. Large-Scale Representation Learning on Graphs via Bootstrapping. In *ICLR*. 1–21.

[38] Yijun Tian, Chuxu Zhang, Zhichun Guo, Yihong Ma, Ronald A. Metoyer, and Nitesh V. Chawla. 2022. Recipe2Vec: Multi-modal Recipe Representation Learning with Graph Neural Networks. In *IJCAI*. 3473–3479.

[39] Zhen Wang, Chunyu Wang, Chao Gao, Xuelong Li, and Xianghua Li. 2020. An evolutionary autoencoder for dynamic community detection. *Sci. China Inf. Sci.* 63, 11 (2020), 1–16. https://doi.org/10.1007/s11432-020-2827-9

[40] Zhen Wang, Chunyu Wang, Xianghua Li, Chao Gao, Xuelong Li, and Junyou Zhu. 2022. Evolutionary Markov Dynamics for Network Community Detection. *IEEE Trans. Knowl. Data Eng.* 34, 3 (2022), 1206–1220. https://doi.org/10.1109/TKDE.2020.2997043

[41] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

[42] Yaqi Xie, Ziwei Xu, Kuldeep S. Meel, Mohan S. Kankanhalli, and Harold Soh. 2019. Embedding Symbolic Knowledge into Deep Networks. In *NeurIPS*. 4235–4245.

[43] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems. In *IJCAI*. 4264–4270.

[44] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. 2019. Bayesian Graph Convolutional Neural Networks for Semi-Supervised Classification. In *AAAI*. 5829–5836.

[45] Kaiyu Zheng, Andrzej Pronobis, and Rajesh P. N. Rao. 2018. Learning Graph-Structured Sum-Product Networks for Probabilistic Semantic Maps. In *AAAI*. 4547–4555.

[46] Shuai Zheng, Zhenfeng Zhu, Xingxing Zhang, Zhizhe Liu, Jian Cheng, and Yao Zhao. 2020. Distribution-Induced Bidirectional Generative Adversarial Network for Graph Representation Learning. In *CVPR*. 7222–7231.

A APPENDIX

A.1 Notation Table

In this section, we present a notation table. Table 4 shows the notations that appear in the model.

Table 4: Key notations.

Symbols	Definitions and descriptions
$\mathbf{G}$	Graph
$\mathbf{E}$	Set of edges
$\mathbf{V}$	Set of nodes
$\mathbf{F}$	Set of features
$\mathbf{A}$	Adjacency matrix
$\mathbf{X}$	Feature matrix
$\mathbf{Z}$	Latent representation matrix
$\mathbf{z}_i$	Latent representation of node v_i
$\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$	The parameters of Gaussian distribution
$\mathbf{A}'$	Reconstructed adjacency matrix
$\mathbf{W}_0$ and $\mathbf{W}_1$	The weight matrices of GCN
$\mathbf{D}$	Degree matrix
$\mathbf{I}$	Identity matrix
$\boldsymbol{\theta}$	Parameters of gating and leaf nodes of CSPN
α, β	Hyper-parameters of loss function
K	Number of variational networks

A.2 Evaluation Metrics

In this section, we present the definitions of the evaluation metrics. For link prediction task, AUC (the area under the ROC curve) and AP (average precision) value are used as evaluation metrics. The description of AUC is as follows:

$$\text{AUC} = \frac{\sum_{C}^{i=1} I(P_{pos}, P_{neg})}{M * N}, \tag{12}$$

$$I(P_{pos}, P_{neg}) = \begin{cases} 1, & \text{if } P_{pos} > P_{neg} \\ 0.5, & \text{if } P_{pos} = P_{neg} \\ 0, & \text{if } P_{pos} < P_{neg} \end{cases}, \tag{13}$$

where M, N, C represent the number of positive examples, negative examples, and all pairs of positive and negative examples, respectively. P_{pos} represents the predicted score of a positive example, and P_{neg} represents the predicted score of a negative example. When the number of sample pairs with positive probability greater than negative probability increases, the AUC tends to 1. This indicates that the larger the value of AUC, the better the performance of the algorithm.

The description of AP is as follows:

$$\text{AP} = \frac{1}{M} \sum P(recall), \tag{14}$$

$$Precison = \frac{TP}{(TP + FP)}, \tag{15}$$

$$Recall = \frac{TP}{(TP + FN)}, \tag{16}$$

where, TP represents the set of data in which positive examples are predicted as positive examples, FP represents the set of data in which negative examples are predicted as positive examples, and FNrepresents the set of data in which positive examples are predicted as negative examples. $p(recall)$ represents the maximum $Precision$ calculated at M number of $Recall$.

For the node classification task, we use accuracy as evaluation metrics. The accuracy is the proportion of correctly classified nodes.