Multi-Scale Variational Graph AutoEncoder for Link Prediction

Zhihao Guo
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
gzh081700@163.com

Feng Wang*
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
sxuwangfeng@126.com

Kaixuan Yao
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
yaokax2@gmail.com

Jiye Liang
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
ljy@sxu.edu.cn

Zhiqiang Wang
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
zhiq.wang@163.com

ABSTRACT

Link prediction has become a significant research problem in deep learning, and the graph-based autoencoder model is one of the most important methods to solve it. The existing graph-based autoencoder models only learn a single set of distributions, which cannot accurately represent the mixed distribution in real graph data. Meanwhile, existing learning models have been greatly restricted when the graph data has insufficient attribute information and inaccurate topology information. In this paper, we propose a novel graph embedding framework, termed *multi-scale variational graph autoencoder* (MSVGAE), which learns multiple sets of low-dimensional vectors of different dimensions through the graph encoder to represent the mixed probability distribution of the original graph data, and performs multiple sampling in each dimension. Furthermore, a self-supervised learning strategy (i.e., graph feature reconstruction auxiliary learning) is introduced to fully use the graph attribute information to help the graph structure learning. Experiment studies on real-world graphs demonstrate that the proposed model achieves state-of-the-art performance compared with other baseline methods in link prediction tasks. Besides, the robustness analysis shows that the proposed MSVGAE method has obvious advantages in the processes of graph data with insufficient attribute information and inaccurate topology information.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Machine learning approaches.*

KEYWORDS

Graph AutoEncoder, Self-Supervised Learning, Graph Neural Networks, Link Prediction

ACM Reference Format:
Zhihao Guo, Feng Wang, Kaixuan Yao, Jiye Liang, and Zhiqiang Wang. 2022. Multi-Scale Variational Graph AutoEncoder for Link Prediction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498531

1 INTRODUCTION

Many fields in the real world can be represented as graph structure data such as community detection [7], biological, and transportation. The real individuals are abstracted as nodes in the graph network, and the interaction between individuals is abstracted as the edges of the graph network. Nowadays, the analysis of graph data attracts more and more attention, especially the research of link prediction [27] has quickly become a very important topic in graph data analysis. Traditional machine learning methods cannot be directly applied to graph data processing tasks because the graph data has a high degree of irregularity and complex structure. Recently, graph representation learning [8, 13, 26] has received careful attention in light of its favorable performance on many graph data modeling tasks, which has the advantage that it can map the graph data node features in the high-dimensional space to the low-dimensional space via the jointly learning between graph attribute information and graph topology information. Indeed, graph embedding learning has been widely used in the research of link prediction tasks.

At present, graph representation learning methods are mainly divided into three research directions: probability models, methods based on matrix factorization, and models based on deep learning [6]. The main learning goal of the probabilistic model is to retain the original topological information in the vectors in the low-dimensional space. Models such as DeepWalk [17], LINE [18] and node2vec [5] use random walks to construct local neighborhood connectivity, and finally learn the network node representation. Methods based on matrix decomposition mainly learn the representation of nodes on the graph by decomposing the adjacency matrix representing the graph data, such as GraRep [1], HOPE [14],

*Feng Wang is the corresponding author.

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498531

M-NMF [25]. Some models based on deep learning are currently receiving the most attention to combine the idea of autoencoders with graph neural networks to construct latent representations of graph data [24]. GAE [10] uses the framework of a self-encoder to apply the graph convolutional network (GCN) [11] to the encoder to achieve the purpose of learning latent node features, and to reconstruct the adjacency matrix through the decoder to achieve the purpose of link prediction. VGAE [10] applies GCN to the encoder of the Variational Autoencoder (VAE) [9], which learns the low-dimensional distribution of node characteristics and then reconstructs the adjacency matrix of the graph data. In addition, models such as MGAE [22] and GALA [16] also learn the representation of node features on the basis of GAE.

It is worth noting that the methods described above only learn a set of low-dimensional vectors to represent the features of nodes when learning low-dimensional vectors representing graph data. In practice, when the attribute information and structure information in the graph data are insufficient, we found that only learning a set of low-dimensional vectors cannot represent the original miscellaneous graph data well. Meanwhile, all existing graph autoencoder models [10] use the GCN [11] as their encoders, in which the adjacency matrix of the encoder is fixed during the model training. Therefore, when the model learns the representation of low-dimensional vectors, each node cannot assign corresponding weights to different neighbor nodes according to the similarity of the features of each neighbor node when aggregating the features of other neighbor nodes. In addition, in the process of model training, when there is little information about valid node links or node features in the graph data, the model′s ability of representing the original graph data is poor, which greatly affects the learning effect and makes the link information still incomplete after the final reconstruction.

In this paper, we propose a novel graph representation framework with self-supervised learning, termed *multi-scale variational graph autoencoder* (MSVGAE). Firstly, our model learns multiple groups of distributions represented by different low-dimensional vectors through the encoder, carries out sampling on each distribution represented by low-dimensional vectors to represent the original complex graph data. Secondly, the graph attention network (GAT) [20] is chosen as the encoder; in this way, when the model learns the distribution of low dimensional vector representation, each node can adaptively aggregate all neighbor nodes, i.e., assigning corresponding weight coefficients to different neighbor nodes according to the similarity of node features. Finally, the model reconstructs the adjacency matrix to generate a more complete adjacency matrix than the previous mutilated adjacency matrix. Meanwhile, in order to construct the new adjacency matrix, the idea of self-supervised learning is introduced in the model, i.e., the feature matrix of the nodes is re-represented using the graph autoencoder model in combination with the new adjacency matrix, and the differences between the re-represented node feature matrix and the original feature matrix are added to the final loss function. In brief, by considering synthetically of graph attribute information, the graph features reconstruction based self-supervised learning can get a better graph structure. The advantage of the self-supervised learning task allows all the predicted topological information to be involved in the computation of the optimization objective function, and the added self-supervised loss can help the model reconstruct a more accurate topological structure matrix.

- We proposed a novel multi-scale variation graph autoencoder (MSVGAE) for link prediction, which can learn a more accurate mixed probability distribution to represent the original graph data.
- A multi-scale graph embedding learning approach is designed to improve the robustness of the graph autoencoder models, which shows good potential to deal with uncertainty issues in several graph modeling tasks.
- A graph feature reconstruction based self-supervised learning is introduced to help the graph structure learning.

2 RELATED WORK

In the initial stage of link prediction, the main calculation is the similarity between nodes. The greater the similarity, the greater the possibility of link generation. The problem of link prediction on the graph was first proposed in social networks. The literature [12] summarized the evaluation index of the similarity of graph topology and pointed out the most accurate (Adamic-Adar Index, AA) index. Later, some better indicators such as resource allocation indicators and partial path indicators were proposed [30]. Considering that the connectivity between nodes depends on the degree of similarity between nodes, a model [3] for link prediction based on the hierarchical structure of the network is also proposed. This model performs well in graph networks with obvious hierarchical structures. The DeepWalk [17] model combines random walk with the word embedding model in natural language processing, and maps the graph network of non-euclidean spatial data to euclidean space to use the embedding vector of the node to solve the link prediction problem. On this basis, many graph network embedding learning models have been proposed, for example, Node2vec [5] and LINE [18].

With the rapid development of graph neural networks in recent years, the link prediction problem has also been further developed. Combining the idea of graph convolutional neural network and variational autoencoder, kipf et al. proposed a variational graph autoencoder (VGAE) [10] and achieved good results on the link prediction problem. Subsequently, a model [23] combined generative adversarial network (GAN) [4] and graph convolutional neural network was proposed for the first time. On this basis, Adversarial Regularized Variational Graph Auto-Encoder (ARVGA) and the Distribution-Induced Bidirectional Generative Adversarial Network (DBGCN) are derived and used for graph representation learning to solve the link prediction problem.

On the other hand, self-supervised learning (SSL) [28] has also been deeply studied by many researchers. Since SSL has the advantage of not requiring any tagged data, and in real life, it often requires a large amount of expensive tagged data, SSL has been widely applied to graph embedded learning. Compared with the rapid development of graph neural networks, the application of SSL in graph data is a new field. Inspired by the image field, some self-supervised learning methods based on graph neural networks, such as DGI [21], began to rise. Chen et al.[2] proposed a self-supervised learning strategy called context recovery, which can better use unmarked data.

In our model MSVGAE, we have learned multiple sets of node embedding representations of different scales so that the model has a stronger representation ability and can also have good robustness in the situation of insufficient graph data information. At the same time, self-supervised learning is introduced into the model to improve learning ability.

3 PROBLEM DEFINITION

$G = (V, \mathcal{E})$ is an undirected graph, $V = \{V_1, \ldots, V_N\}$ is the set of nodes in the graph, N is the number of nodes, $\mathcal{E}$ is the set of edges in the graph, $X = \{X_1, \ldots, X_N\} \in \mathbb{R}^{N \times F}$ is the feature matrix of all nodes in the graph, and F is the dimension of features each node. The adjacency matrix $A \in \mathbb{R}^{N \times N}$ represents the topological structure of the graph.

The main purpose of the graph autoencoder for the link prediction is to generate a more complete graph topology with the graph attribute information and less topology information. Specificially, according to the given graph feature matrix $X \in \mathbb{R}^{N \times F}$ and incomplete edge set $\mathcal{E}$, the model learns the low-dimensional distribution of node vectors through the encoder (GCN in most models). Then the new graph embedding $Z \in \mathbb{R}^{N \times F'}$ is obtained, which is sampled via the potential distribution of the graph data. Finally, the topology of the graph is reconstructed by the decoder (link prediction: predict the possibility of a connection between node V_p and node V_q), and a new adjacency matrix $A' \in \mathbb{R}^{N \times N}$ is obtained.

4 MODEL ARCHITECTURE

4.1 Overall Structure

Fig.1 shows the overall structure of our multi-scale variational graph autoencoder (MSVGAE), which consists of three parts: graph encoder, decoder, and a self-supervised learning module. The graph encoder uses a graph attention mechanism to implement a variational inference model and learns the probability distributions of multiple sets of low-dimensional vector representations of different dimensions to estimate the true posterior probability distribution of the graph data. Technically, it learns multiple sets of latent representations $Z \in \mathbb{R}^{N \times F'}$. The decoder implements a generative model to reconstruct a new adjacency matrix $A' \in \mathbb{R}^{N \times N}$(learning from Z to get A'). Finally, a self-supervised learning task is introduced to reconstruct the graph attribute information via the potential representation Z learned by the encoder and the reconstructed adjacent matrix A' learned by the decoder. This kind of graph feature reconstruction based auxiliary learning can make full use of the graph attribute information to help generate the complete graph structure.

4.2 Multi-Scale Variational Graph Autoencoder

4.2.1 Inference Model. In the variational model of the graph encoder, the node feature matrix $X \in \mathbb{R}^{N \times F}$ of the graph data and the given links set $\mathcal{E}$ are taken as inputs. The variational part is responsible for the graph embedding learning of the whole model and the dimensionality reduction representation of complex and high-dimensional graphs. Unlike most models that use the vanilla GCN [11] as the encoders, we use the graph attention network (GAT) [20] to implement the graph encoder, i.e., a multi-head attention network is used to aggregate the features of all neighbor nodes that distance is 1 for each node. The parametric inference model of variational probability distribution can be expressed as:

$$q(Z|X, \mathcal{E}) = \prod_{i=1}^{N} q(Z_i|X, \mathcal{E}), \tag{1}$$

$$q(Z_i|X, \mathcal{E}) = \mathcal{N}(Z_i|\mu_i, diag(\sigma_i^2)), \tag{2}$$

where μ_i and σ_i^2 represent the mean vector and the variance vector corresponding to the node V_i, respectively, which are learned by the encoder.

$$\mu = GAT_\mu(X, \mathcal{E}), \quad log_\sigma = GAT_\sigma(X, \mathcal{E}). \tag{3}$$

The following describes our graph attention mechanism, we have made a little change to GAT. $W \in \mathbb{R}^{F \times F'}$ is a shared weight parameterized matrix for linear transformation learning for each node. When the features of the node V_i and its neighbor V_j are aggregated, the corresponding positions of the vectors are multiplied, which can better calculate the similarity between nodes. The attention mechanism $a \in \mathbb{R}^{F'}$ is applied to LeakyRelu through a single-layer feedforward neural network parameterized by a weight vector.

$$e_{i,j} = a(W\vec{h}_i \times W\vec{h}_j). \tag{4}$$

The feature vector of the input node can be represented by a set $\vec{h} = \left\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\right\}, \vec{h}_i \in \mathbb{R}^F$, and the neighbor features are aggregated by the attention layer and then represented by the set $\vec{h'} = \left\{\vec{h'}_1, \vec{h'}_2, \ldots, \vec{h'}_N\right\}, \vec{h}'_i \in \mathbb{R}^{F'}$ as the output. Where $\boldsymbol{\alpha}_{i,j}$ is the weight of node V_i relative to node V_j, g is the *sigmoid* activation function.

$$h'_i = g(\sum_{j \in \mathcal{N}_i} \alpha_{i,j} W\vec{h}_j), \tag{5}$$

$$\boldsymbol{\alpha}_{i,j} = \frac{exp(LeakyRelu(e_{i,j}))}{\sum_{k \in \mathcal{N}_i} exp(LeakyRelu(e_{i,k}))}. \tag{6}$$

In order to make the learning process more stable, multi-head attention is used, and each node learns K sets of weights. The final node vector $\vec{h}'_i$ can be expressed as:

$$\vec{h}'_i = g(\frac{1}{K}\sum_{k=1}^{K}\sum_{k \in \mathcal{N}_i} \alpha_{i,j}^k W^k \vec{h}_j). \tag{7}$$

4.2.2 Multi-Scale Learning. At present, most link prediction models based on variational graph autoencoders only learn a set of hidden variable representations in the encoder part to approximate the complex distribution of the original data. When the feature information or structure information of the graph data is insufficient, only a set of distributions cannot well represent the complex distribution of the original data. Therefore, our model learns hidden vector matrix Z of different dimensions (16, 32, 64, 128) through the encoder, samples two sets of probability distributions on each dimension, and learns several sets of distributions of different dimensions to approximate the real distribution (hidden variable matrix Z).The expression of Z sampling is: $Z_i = exp(\sigma_i) \times e_i + \mu_i$.

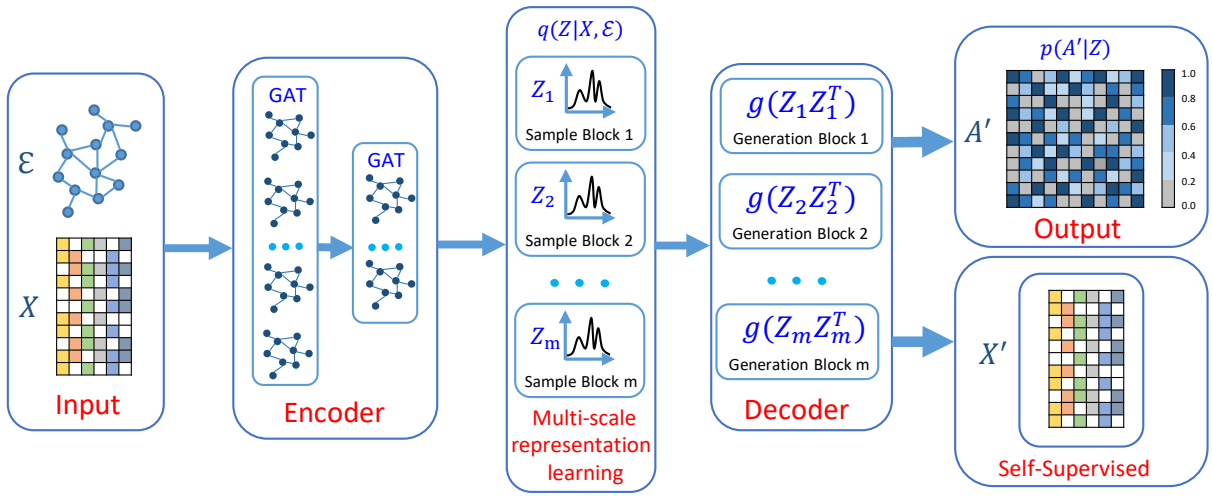


Figure 1: The overall architecture of the proposed MSVGAE. Using the graph feature matrix X and the edge set $\mathcal{E}$, multiple sets of embedding matrix Z with different scales are generated by the encoder, which are used to reconstruct the adjacency matrix A' via the generator. Finally, the graph convolutional neural network (GCN) is used to learn the new eigenmatrix X', and X' participates in the self-supervised learning task.

4.2.3 Generative Model. In the decoder part, we multiply all the low-dimensional hidden variable matrix Z sampled by the encoder and its transposed matrix Z^T to generate the topology structure.

$$p(A'|Z) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A'_{i,j}|Z_i, Z_j), \tag{8}$$

$$p(A'_{i,j} = 1|Z_i, Z_j) = g(Z_i^T Z_j), \tag{9}$$

where $A'_{i,j}$ is an element of the reconstructed adjacency matrix.

4.3 Self-Supervised Learning Task

In real life, often there is only a very small amount of labeled information in a large amount of data. Therefore, when there is too little information about the known node links, the model cannot accurately represent the original data distribution, which seriously affects the learning effect of the model. To address the problem, a self-supervised learning task is introduced in our model, it aims to add auxiliary tasks to improve the accuracy of the main learning task and enhance the performance of the model.

In our model, after learning the low-dimensional representation of the original data through the encoder, the auxiliary task of relearning the nodal feature representation is added along with the main learning task of reconstructing the adjacency matrix (as shown in figure 2), the model is also capable of learning well when faced with insufficient feature and topological information of the data. The learning problem can be represented as:

$$f^*, h^* = arg\min_{(f,h)} \mathcal{L}_{sup}(f, h, Q) + \lambda \mathcal{L}_{ssl}(f, P), \tag{10}$$

where Q represents the link prediction of the model as the main learning task, P represents the feature distribution of the original graph data, the autoencoder of the model is represented as f, and the prediction head is represented as h. In the formula, f is trained under the self-supervision, and h is trained under the supervision of Q. λ is a positive scalar weight that balances the two terms in the loss.

The self-supervised learning task uses graph convolution network (GCN) [11] propagation, and its process can be represented as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}'\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \tag{11}$$

where $\tilde{A}'$=A'+I_N is the new adjacency matrix generated by the decoder and adds self-connection, and I is the identity matrix. D is the degree matrix, and it is used to normalize A'. $H^{(l)}$ represents the characteristics of the lth layer, and $W^{(l)}$ represents the weight of the lth layer.

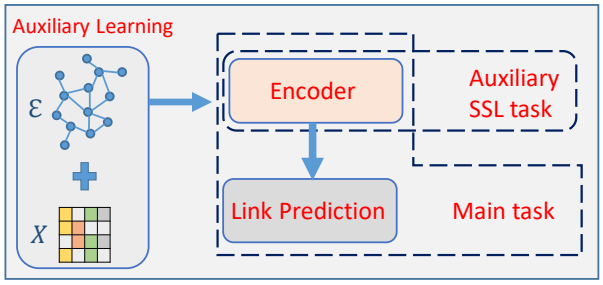


Figure 2: An auxiliary task of self-supervised learning is included in the learning process of the model to help the learning of the model's main task. The autocoder is trained jointly by both the auxiliary task and the main task.

Algorithm 1 The learning process of MSVGAE

Input: Graph $G = (X, \mathcal{E})$

Parameter: T is the number of iterations, M is the number of sampled hidden matrices Z, N is the number of nodes in the graph

Output: Reconstructed topological matrix $A' \in \mathbb{R}^{N \times N}$

1: **for** iterator=1, 2, 3,$\cdots\cdots$, T **do**
2: Learn multiple sets of latent matrices by Eq.(1): $\{Z_1, Z_2, \cdots, Z_m\}$
3: Reconstruct topological matrixs by Eq.(9): $\{A_1, A_2, \cdots, A_m\}$
4: Calculate the final topology matrix : $A' = \frac{1}{M}\sum_{i=1}^{M} A_i$
5: Self-Supervised learning tasks by Eq.(10)
6: Update the parameters of the MSVGAE by Eq.(12)
7: **end for**
8: **return** $A' \in \mathbb{R}^{N \times N}$

4.4 Learning

The optimization function for model training is:

$$\mathcal{L} = \frac{1}{M}\sum_{m=1}^{M}\Big(E_{(q,Z_m|X,\mathcal{E})}[logp(A'_m|Z_m)] - KL[q(Z_m|X,\mathcal{E})||p(Z)]\Big) - \frac{1}{N}\sum_{t=1}^{N}(X - X')^2, \quad (12)$$

where $E_{(q,Z|X,\mathcal{E})}[logp(A'|Z)]$ represents the distance measurement between the generated topology graph and the original topology graph, $KL[q(Z|X,\mathcal{E})||p(Z)]$ represents a measure of the difference between two probability distributions p and q. We use Gaussian prior $p(Z) = \prod_{i=1} \mathcal{N}(Z_i|0, I)$. M represents the total number of dimensions of embeddings Z.

In formula (12), the loss function in the last part of the self-supervised learning task denotes the mean square error (MSE), it represents the sum of squares of the differences between $X \in \mathbb{R}^{N \times F}$ (the original feature matrix) and $X' \in \mathbb{R}^{N \times F}$ (the reconstructed feature matrix). It is worth that in the task of reconstructing $X' \in \mathbb{R}^{N \times F}$, $X \in \mathbb{R}^{N \times F}$ is re-represented using a GCN graph convolutional neural network combined with the reconstructed adjacency matrix $A' \in \mathbb{R}^{N \times N}$. One advantage is that the assisted training of the SSL task leads to more accurate link information generated by the reconstructed adjacency matrix $A' \in \mathbb{R}^{N \times N}$ compared to the unsupervised task.

4.5 Algorithm Explanation

The specific operation of our proposed MSVGAE method is shown in Algorithm 1. The input part is a graph G, and step 2 learns the distribution represented by multiple sets of the latent variable matrix Z from the encoder (GAT). In Step 3, the corresponding topology matrix is reconstructed for each Z in Step 2. In Step 4, the average value of all topology structures is calculated, and the self-supervised learning task assist model is used for training in Step 5. In step 6, formula 8 is used to update all the parameters of the model (MSVGAE) using gradient descent. Finally, in step 8, the topological structure $A' \in \mathbb{R}^{N \times N}$ obtained by the final training is returned.

5 EXPERIMENTS

To verify the effectiveness of the proposed method, this paper conducted experiments on five real-world graphs datasets.

5.1 Evaluation Setup and Metrics

5.1.1 Datasets. For our link prediction model MSVGAE, we choose to evaluate on five widely used graph data sets. They are Citation Network (Cora, Citeseer, Pubmed), Co-Buy Charts (Computers, Photo). The total number of nodes, edge sets, total number of features of nodes, and detailed information of node categories in each data set are shown in Table 1.

Table 1: Real-world graph datasets used in the paper

Datasets	Nodes	Edges	Attributes	Classes
Cora	2708	5278	1433	7
CiteSeer	3327	4552	3703	6
Dblp	17716	52867	1639	4
Photo	7650	119081	745	8
Computers	13752	245861	767	10

5.1.2 Baselines. We compare our model with some other link prediction models: **GAE** [10]: The node embedding model that combines the graph domain with the auto-encoder uses the topological information and feature information of the graph. **VGAE** [10]: The graph embedding model combines the graph domain and the variational autoencoder. **ARGA** [15]: Adversarially regularized auto-encoder graph learning embedding algorithm. **ARVGA** [15]: A variational graph autoencoder based on ARGA learns embedding. **DBGAN** [29]: The distribution-induced bidirectional generative confrontation network is used for graph representation learning. **MSVGAE**: Our proposed a graph embedding multi-scale representation learning based on a variational autoencoder.

5.1.3 Metrics. We use AUC value (the area under the receiver operating characteristic curve) and AP value (the area under the precision-recall rate curve) as the evaluation indicators of the model. The data set is divided into training set (85% of the edge set), validation set (5% of the edge set), and test set (10% of the edge set). The average and standard deviation of each model after five trainings are used as the final score of the model.

5.1.4 Parameter Settings. Because the total number of nodes and edges in the Cora and Citeseer data sets are small, for the proposed MSVGAE, the learning rate is set to 0.01, and the model training only needs to learn 20 iterations. For other data sets (Dblp, Photo, Computers, CS), due to the large number of edges and node features in the data set, the learning rate of the model MSVGAE is uniformly set to 0.001 during training, and the number of training iterations is set to 600. The dimensions of the four hidden layers learned by the encoder are set to 16, 32, 64, and 128, respectively. All other baseline parameters are consistent with the original paper.

5.2 The Results of Link Prediction

The detailed information of the experimental results of different models on the link prediction task is shown in Table 2. It can be

Table 2: Results for link prediction

Methods	Cora		Citeseer		Dblp		Photo		Computers	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
GAE	91.5±0.02	92.6±0.03	91.0±0.04	91.7±0.03	92.5±0.01	92.4±0.01	96.5±0.03	96.2±0.02	92.5±0.03	92.8±0.02
VGAE	91.5±0.04	92.3±0.05	91.2±0.03	92.4±0.05	94.6±0.03	94.6±0.02	95.2±0.04	94.9±0.04	92.5±0.04	92.8±0.05
ARGA	91.5±0.03	93.0±0.01	92.8±0.03	93.9±0.03	95.5±0.01	95.8±0.01	94.3±0.02	93.7±0.02	94.2±0.02	94.3±0.01
ARVGA	93.1±0.03	93.4±0.04	92.5±0.01	93.4±0.04	95.6±0.01	95.8±0.01	93.7±0.04	92.5±0.05	93.7±0.01	93.1±0.01
DBGAN	94.5±0.01	95.1±0.05	94.5±0.04	95.8±0.01	96.0±0.01	96.4±0.01	96.3±0.01	95.8±0.01	94.6±0.01	94.2±0.02
MSVGAE	**95.3±0.05**	**95.4±0.04**	**95.4±0.03**	**96.1±0.04**	95.7±0.01	95.3±0.02	**96.7±0.01**	**96.3±0.01**	**95.1±0.02**	**94.6±0.01**

found from the table that under different data sets, the proposed MSVGAE model is superior to other comparison methods in link prediction tasks, and shows better performance. Besides, except for individual data sets, the results of both AUC and AP are more than the 95.0. The reasons are as follows. On the one hand, the proposed method adopts multi-scale representation learning, which has better representation ability in graph embedding learning. On the other hand, different from the traditional graph autoencoder methods, the label information of nodes is used as the characteristics of nodes to assist model training in this paper.

5.3 Ablation Study

The above experimental results show that the advantages of the proposed MSVGAE algorithm are verified by the experimental results and theoretical analysis of each algorithm. In order to further verify the effectiveness of the self-supervised learning task used in the model, we conducted VAE, VGAE and MSVGAE experiments on Cora and Citeseer data sets. Finally, the learning ability of each algorithm was compared, and the ablation study of each algorithm was as follows:

- GAE: Non-Self-Supervised Learning;
- GAE+SSL: With Self-Supervised Learning;
- VGAE: Non-Self-Supervised Learning;
- VGAE+SSL: With Self-Supervised Learning;
- MSVGAE: Non-Self-Supervised Learning;
- MSVGAE+SSL: With Self-Supervised Learning

The ablation experimental results of each algorithm are listed in Table 3. It can be seen that the performance of each algorithm with the assisted training of self-supervised learning task is better than that without self-supervised learning task. That is, the proposed self-supervised learning task can assist the model to get better performance, which indicates the superiority of self-supervised learning. This is consistent with the theoretical and empirical studies we discussed earlier.

5.4 Robustness Analysis

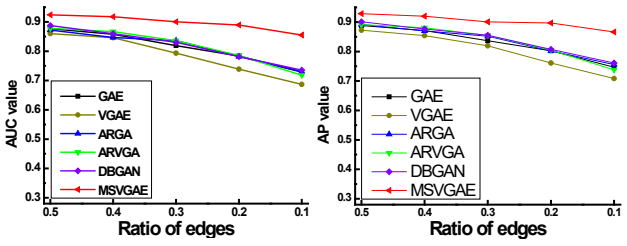
To verify the robustness of the proposed MSVGAE models, we attacked the total number of edge sets, the number of node features, and the number of simultaneous attack edge sets and node features in the training set, respectively. In different ways, we can reduce the ratio of edges or nodes in the training set, which comprehensively evaluates the robustness of the model and compares it with other

Table 3: Ablation studies on two datasets.

Methods	Cora		Citeseer	
	AUC	AP	AUC	AP
GAE	91.5±0.02	92.6±0.03	91.0±0.04	91.7±0.03
GAE+SSL	92.3±0.02	93.5±0.02	91.6±0.03	92.3±0.04
VGAE	91.5±0.04	92.3±0.05	92.8±0.03	93.9±0.03
VGAE+SSL	92.2±0.03	93.6±0.03	93.4±0.02	94.3±0.02
MSVGAE	94.8±0.05	94.2±0.04	94.6±0.04	95.4±0.05
MSVGAE+SSL	95.3±0.05	95.4±0.04	95.4±0.03	96.1±0.04

baselines. To ensure the timeliness of the algorithms, the Cora data set is selected for testing in the following experiments.

5.4.1 Random Attack Edges. We continuously reduce the number of edge sets in training set to 50%, 40%, 30%, 20%, and 10%, respectively. The detailed information of the experimental effect of our model and other baselines is shown in Figure 3. In the figure, subgraph (a) is the AUC value of each model corresponding to different ratio of edges, and subgraph (b) is the AP value of each model corresponding to different edge ratios.



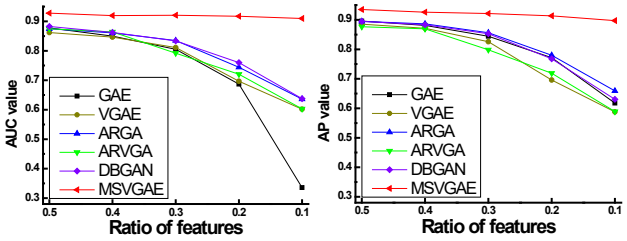
(a) AUC value of Random attack (b) AP value of Random attack

Figure 3: The results of edge attack.

As can be seen from the figure, as a whole, as the edge ratio in the training set continues to decrease, the overall trend of each model shows a downward trend. However, the proposed model has a smaller decline and is always better than other comparison methods. For the comparison methods, when the edge ratio is 0.1, the results of AUC and AP are below 0.8. This is because when learning the embedding representation of nodes, these methods

only learn a set of low-dimensional vector representations. In other words, they cannot fully fit the original graph data. The proposed model is around 0.9. In addition, the VGAE method performs poorly.

5.4.2 Random Attack Features. We continuously reduce the number of features of each node by attacking the node features in the training set, and the attack method is consistent with the edge of the node. The detailed information of the experimental effect of our model and other baselines is shown in Figure 4. In the figure, subgraph (a) is the AUC value of each model corresponding to different ratio of features, and subgraph (b) is the AP value of each model corresponding to different ratio of features.

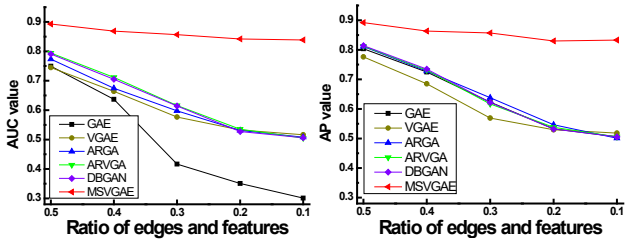


(a) AUC value of Random attack (b) AP value of Random attack

Figure 4: The results of feature attack.

As can be seen from the figure, as a whole, as the ratio of features in the training set continues to decrease, the proposed method is almost maintained at around 0.92, and shows the best results, which indicates that the method has strong robustness. For the comparison methods, when the ratio of features is in the range of 0.5-0.4, it shows better model performance, but after it is 0.4, these models show a clear downward trend. Especially for the VGAE method, when the ratio of features is 0.1, the value of AUC is 0.3. This is because when the features of the neighbor nodes are aggregated, the neighbor nodes cannot be assigned weight coefficients.

5.4.3 Random Attack Edges and Features. By attacking the edges and features of each node in the training set, we continuously reduce the number of them. The detailed information of the experimental effect of our model and other baselines is shown in Figure 5. In the figure, subgraph (a) is the AUC value of each model corresponding to different ratio of the edges and features, and subgraph (b) is the AP value of each model corresponding to different ratio of the edges and features. It can be seen from the figure that in the most extreme case, i.e., when attacking the set of edges and the set of node features in training set at the same time, the proposed method is still superior to other comparison methods and remains above 0.85. For the comparison methods, when the set of edges and node features is 0.5, the result of ACU is below 0.8, and the result of AP is between 0.8. Besides, for the overall trend, the methods have a relatively large decline. In summary, from different perspectives, the proposed method shows good robustness. This is mainly because that the multi-scale variational inference can learn the data distribution better and the introduced self-supervised learning task can make full use of the attribute feature information.



(a) AUC value of Random attack (b) AP value of Random attack

Figure 5: The results of edge and feature attack.

5.5 Evaluation on Node Clustering

We evaluated the node clustering task, and the parameter settings of each model in the clustering task were the same as those of the link prediction task. After learning the embedded representation of the graph, the model uses three indicators: normalized mutual information (NMI), adjusted mutual information (AMI) and average Rand index (ARI) to verify the clustering results.

On Cora and Citeseer data sets, the clustering results of GAE, VGAE and MSVGAE models are shown in Table 4. The results showed that MSVGAE significantly improves all three indicators compared to the other two baselines. For example, on the Cora data set, the normalized mutual information (NMI) of MSVGAE increases from 0.344 to 0.452 compared with GAE, and the adjusted mutual information (AMI) increases from 0.268 to 0.450 compared with VGAE. The average Rand index (ARI) increased from 0.192 to 0.377. Overall, the proposed MCVGAE method has better experimental results, which indicates that the proposed model is superior to the comparison methods, i.e., it shows the superiority of the model in terms of clustering levels.

Table 4: Clustering results.

Methods	Cora			Citeseer		
	NMI	AMI	ARI	NMI	AMI	ARI
GAE	0.344	0.341	0.276	0.126	0.124	0.108
VGAE	0.270	0.268	0.192	0.108	0.107	0.101
MSVGAE	**0.452**	**0.450**	**0.377**	**0.257**	**0.245**	**0.243**

5.6 Graph Visualization

A good unsupervised graph embedding algorithm can usually represent the original graph structure well in low dimensional space. To illustrate graphically the representativeness of graphs embedded in low dimensional space. We use t-SNE [19] algorithm to visualize the low-dimensional features learned by GAE, VGAE, MSVGAE and other models in two-dimensional space. Figures 6 and 7 show the visualization results of the three models on the Cora and Citeseer datasets, respectively. From the visualization results, it can be clearly seen that, overall, the proposed MSVGAE method has the advantage, followed by the VGAE method, and the GAE method has a poor performance. Besides, even MSVGAE takes an unsupervised approach to graph representation learning. However, in the final

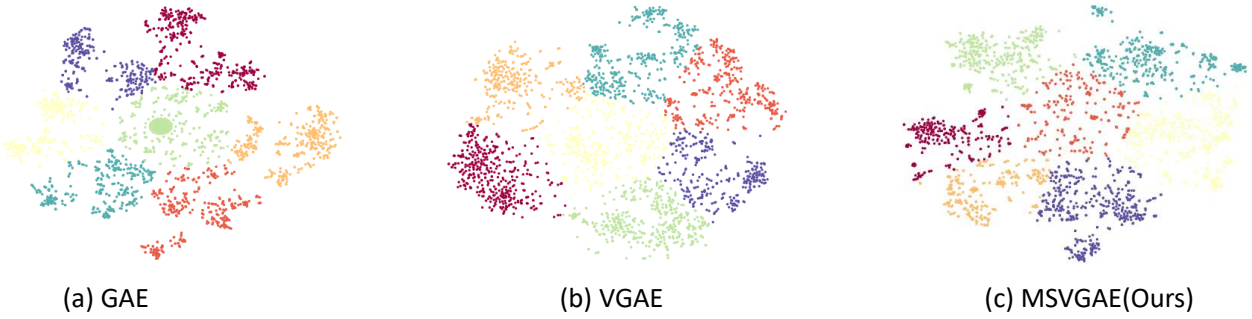


Figure 6: Visualization of the Cora dataset.

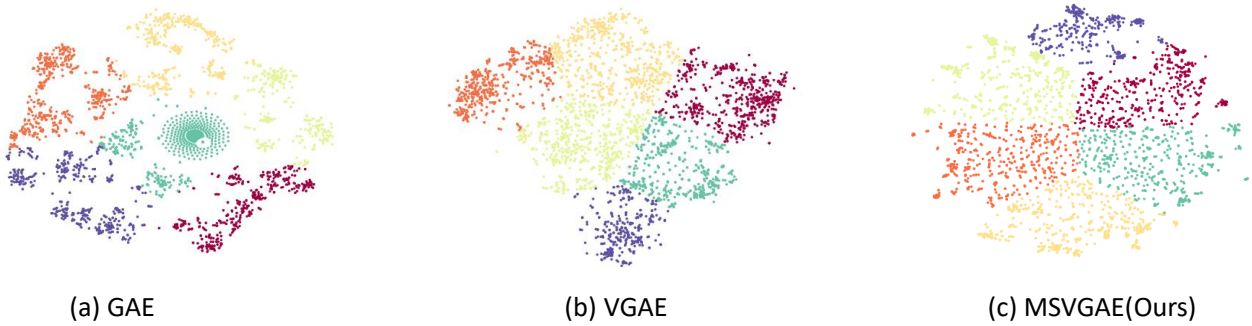


Figure 7: Visualization of the Citeseer dataset.

clustering results, the differentiation between categories in the visualization diagram of the MSVGAE model is more obvious. Therefore, compared with other baselines, MSVGAE's graph representation learning ability is better, and the distribution of low-dimensional representation is more accurate.

6 CONCLUSION

In this paper, we propose a novel multi-scale graph representation learning framework with a GAT-based autoencoder termed MSVGAE for link prediction. Due to the distribution of graph data is a complex mixed probability distribution in many real-world applications, we learn multiple scales of low-dimensional representations to represent the complex graph data. On this basis, a graph feature reconstruction based self-supervised learning is introduced, which can improve the performance of graph structure learning effectively. To verify the effectiveness of the model, five real-world graphs datasets are employed in the experiments. Experimental results further validate the proposed model is feasible and effective for link prediction tasks with attribute information and topology information of nodes being insufficient.

7 ACKNOWLEDGEMENTS

This work is supported by the National Key Research and Development Program of China (under grant 2020AAA0106100), the National Natural Science Foundation of China (61906111, 72171137).

REFERENCES

[1] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 891–900.
[2] Liang Chen, Paul Bentley, Kensaku Mori, Kazunari Misawa, Michitaka Fujiwara, and Daniel Rueckert. 2019. Self-supervised learning for medical image analysis using image context restoration. *Medical Image Analysis* 58 (2019), 101539.
[3] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. 2008. Hierarchical structure and the prediction of missing links in networks. *Nature* 453, 7191 (2008), 98–101.
[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems* 27 (2014), 2672–2680.
[5] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.
[6] Di Jin, Cuiying Huo, Chundong Liang, and Liang Yang. 2021. Heterogeneous Graph Neural Network via Attribute Completion. In *Proceedings of the Web Conference 2021*. 391–400.
[7] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, Philip Yu, and Weixiong Zhang. 2021. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning. *IEEE Transactions on Knowledge and Data Engineering* (2021). https://doi.org/10.1109/TKDE.2021.3104155
[8] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 148–156.
[9] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.
[10] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Bayesian Deep Learning Workshop on NIPS*.
[11] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.
[12] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology* 58, 7 (2007), 1019–1031.
[13] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. 2020. Path Integral Based Convolution and Pooling for Graph Neural Networks. In *Advances*

in Neural Information Processing Systems. 16433–16445.

[14] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1105–1114.

[15] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. *ArXiv Preprint ArXiv:1802.04407* (2018).

[16] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*. 6519–6528.

[17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 701–710.

[18] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. 1067–1077.

[19] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008).

[20] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[21] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep Graph Infomax. In *International Conference on Learning Representations*.

[22] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 889–898.

[23] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the Thirty-second AAAI Conference on Artificial Intelligence*.

[24] Jie Wang, Jianqing Liang, Junbiao Cui, and Jiye Liang. 2021. Semi-supervised learning with mixed-order graph convolutional networks. *Information Sciences* 573 (2021), 171–181.

[25] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *Proceedings of the Thirty-first AAAI conference on artificial intelligence*.

[26] Yu Guang Wang, Ming Li, Zheng Ma, Guido Montufar, Xiaosheng Zhuang, and Yanan Fan. 2020. Haar graph pooling. In *Proceedings of the International Conference on Machine Learning*. 9952–9962.

[27] Zhitao Wang, Chengyao Chen, and Wenjie Li. 2017. Predictive network representation learning for link prediction. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 969–972.

[28] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2021. Self-supervised learning of graph neural networks: A unified review. *ArXiv Preprint ArXiv:2102.10757* (2021).

[29] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7224–7233.

[30] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. 2009. Predicting missing links via local information. *The European Physical Journal B* 71, 4 (2009), 623–630.