Journal of Computer Science and Electrical Engineering
ISSN: 2663-1946
DOI: 10.61784/jcsee3007

GRAPH REPRESENTATION LEARNING BASED ON ONE-SHOT AGGREGATION AND SECOND-ORDER INFORMATION

LiNing Yuan, ZhongYu Xing*, WanYan Huang
School of Information Technology, Guangxi Police College, Nanning 530028, Guangxi, China.
Corresponding Author: ZhongYu Xing, Email: xingxhongyu@gcjcxy.edu.cn

Abstract: The graph autoencoder has emerged as a proficient model for graph representation learning, demonstrating remarkable efficacy in tasks like link prediction. Nevertheless, most graph autoencoders are characterized by their shallow architecture, leading to diminished efficiency as the number of hidden layers increases. Moreover, these approaches predominantly leverage graph convolutional networks for encoding adjacency matrices and attribute matrices, thereby underutilizing higher-order structural characteristics, such as second-order information. To address these issues, the Variational Graph Autoencoder model OS-SeVAE and the Autoencoder model OS-SeAE, which integrate One-Shot aggregation and second-order information, have been introduced. Initially, deep encoders are formulated by combining graph convolution and second-order graph convolution, alongside the incorporation of One-Shot aggregation and the Exponential Linear Unit (ELU) function. Subsequently, the decoder component employs inner product decoding to reconstruct the graph's topological structure. To prevent overfitting during model training, a regularization term is introduced based on the autoencoder loss function. Experimental results show that One-Shot aggregation and ELU function can effectively improve the performance of deep graph autoencoders, enhance the gradient information propagation of the model, and the introduction of second-order information strengthens the model's representation capability. In link prediction tasks conducted on three benchmark citation datasets, the experimental results of OSA-VAE and OS-SeAE are superior to current state-of-the-art baseline models.
Keywords: Graph representation learning; Graph convolutional network; One-shot aggregation; Second-order information

1 INTRODUCTION

As a kind of non-Euclidean data, graphs contain information with high-dimensional implicit characteristics, which makes traditional graph representation learning methods based on manually designed features usually extremely complex [1]. Graph representation learning models based on deep learning algorithms [2] have strong characterization capabilities, which can effectively model the nonlinear structure of graph data while reducing the data dimensionality, retaining important information such as topology and node attributes, and have become a research hotspot for Graph representation learning models. Among them, unsupervised models based on deep learning are able to select representative features from the data in the absence of a priori knowledge or limited labeling information, so that the generated node embeddings take into account smaller embedding dimensions while retaining attributes and edge information as much as possible.

Currently, unsupervised Graph representation learning models based on deep learning are mainly categorized into random walk [3] and autoencoder models [4]. Random walk-based models obtain a training corpus by random walks, and then integrate the corpus into Skip-Gram [5] to obtain low-dimensional embedding representations of nodes. Such methods usually take the entire network structure as input and can effectively capture neighborhood similarity, but fail to fully utilize the node attributes that provide important information about the original graph in the embedding generation process. Autoencoder-based Graph representation learning models take the topology and node attribute information of the graph as encoder inputs to generate low-dimensional embeddings, and then utilize a decoder to reconstruct the graph structure. Although deep graph representation learning methods have achieved great success [6], there are still some problems. A recent study exposes that the performance of Graph representation learning models based on graph convolutional autoencoders decreases with increasing neural network depth and quantitatively measures it [7]. This performance degradation is related to many factors, such as gradient vanishing in backpropagation, overfitting due to parameter increase, and over-smoothing phenomenon [8]. Li et al. were the first to investigate the over-smoothing phenomenon [9], demonstrating that graph convolution employed by Graph Convolutional Networks (GCN) [10] is a special form of Laplace smoothing, which is a smoothing operation capable of fusing the features of a node's own and its neighbors, resulting in the similarity of features of nodes in the same cluster. This property allows shallow GCN models to achieve excellent experimental performance, but as the number of layers increases, the features of each node will converge to similar values, causing nodes from different clusters to become indistinguishable and the phenomenon of over-smoothing. For the graph convolutional autoencoder model, an increase in encoder depth similarly leads to over-smoothing of the model, resulting in the generation of low-dimensional embeddings that perform poorly in downstream graph analysis tasks.

To address the above problems, this paper proposes deep Graph AutoEncoder (GAE) and Variational Graph AutoEncoder (VGAE) models based on One-Shot aggregation [11] and Exponential Linear Unit (ELU) function [12] to

solve the phenomena of gradient vanishing and over-smoothing of the encoder due to depth increase. Meanwhile, a regularization term is introduced into the loss function to prevent the model from parameter overfitting due to depth increase. The above improvement effectively solves the problem that the performance of deep graph embedded model decreases due to the increase of model depth. In addition, a novel Second-Order Proximity Graph Convolutional Networks (SeGCN) is proposed in this paper to construct encoders for deep GAE and VGAE models to enhance the characterization ability.

In summary, the main contributions of this paper are as follows: (1) A modification of the GCN proposes a SeGCN that preserves second-order similarity information; (2) presents OS-SeVAE and OS-SeAE models using One-Shot aggregation, ELU activation function and SeGCN encoder to enhance the representation of the models while addressing the phenomena of vanishing gradient, parameter overfitting and over-smoothing in deep models; (3) in link prediction experiments on the three benchmark citation datasets, OS-SeVAE and OS-SeAE consistently outperform the baseline model at the same depth; (4) OS-SeVAE and OS-SeAE performances do not show large fluctuations with the increase in the number of hidden layers, and the experimental performances are also more stable compared to the baseline model.

The rest of the paper is organized as follows: the section 2 gives an introduction to the basic algorithms and related theories used in the models, the section 3 describes the model structure and loss function in detail, the section 4 proves the reasonableness and effectiveness of the method through many experiments, and finally concludes the research in the section 5.

2 PRELIMINARIES

2.1 Variational AutoEncoder

VAE utilizes neural networks to construct two probability density distribution models on the original AE structure: (1) an inference network, which is used for variational inference of the original input data to generate variational probability distributions of the hidden variables; (2) a generation network, which generates approximate distributions of the original data based on the hidden variable probability distributions. Specifically, the inferential network utilizes an encoder to generate the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ of the Gaussian distribution, and the generative network utilizes a decoder to generate the reconstructed probability distribution. In fact, VAE is the addition of appropriate noise to the AE coding process. First, using the neural network encoder, the mean encoding $\boldsymbol{\mu}=(\mu_1,\mu_2,\mu_3)$ and the variance encoding $\boldsymbol{\sigma}=(\sigma_1,\sigma_2,\sigma_3)$ are computed. The variance encoding $\boldsymbol{\sigma}$ is used to control the weights of the noise encoding $\boldsymbol{\varepsilon}=(\varepsilon_1,\varepsilon_2,\varepsilon_3)$ and thus the degree of noise interference, and an exponential operation is applied to $\boldsymbol{\sigma}$ in order to ensure that the weights assigned to $\boldsymbol{\varepsilon}$ are positive. Then, the mean value encoding and the noise encoding after weight assignment are superimposed to generate a vector representation of the input data $\boldsymbol{y}=(y_1,y_2,y_3)$. Finally, $\boldsymbol{y}$ is fed to the decoder for reduction. In contrast to AE, VAE introduces constraints about noise in addition to the use of reconstruction loss:

$$\sum_{i=1}^{3}(e^{\sigma_i}-(1+\sigma_i)+(\mu_i)^2) \tag{1}$$

Assuming that the above constraints are not added, the model, in order to ensure the quality of the generated results, encodes the results with the hope that the noise will interfere as little as possible with the generated results, and so assigns smaller weights to the noise (the effect of noise can be eliminated when the variance encoding is set to close to negative infinity). This usually results in a model that is well trained, but generates results with poor performance. With the addition of equation (1), the derivation of $\boldsymbol{\sigma}$ yields $e^{\boldsymbol{\sigma}}-1$, and the derivative equation takes a minimal value at $\boldsymbol{\sigma}=0$, which constrains the variance coding from rapidly converging to negative infinity, and thus acts as a regularization constraint. The VAE is constructed as a probabilistic model from a Gaussian mixture model, so the reconstruction loss and noise constraints are usually rewritten in the following form using the prior distribution, the posterior distribution, and the KL dispersion [13]:

$$\mathrm{L}_{VAE}=\mathbb{E}_{q(\boldsymbol{y}|\boldsymbol{x})}[\log(\boldsymbol{x}\mid\boldsymbol{y})]-\mathrm{KL}[q(\boldsymbol{y}\mid\boldsymbol{x})\,\|\,p(\boldsymbol{y})] \tag{2}$$

Compared with the original AE, VAE has the following main differences: (1) the distribution of the hidden layer representation in AE is unknown, whereas the hidden variables in VAE obey a normal distribution; (2) what is learned in AE is only the encoder and the decoder, on top of which VAE learns the distribution of the hidden variables, the mean and the variance of the Gaussian distribution; and (3) while AE is only able to derive the corresponding encoding from the samples, VAE learns the the parameters of the Gaussian distribution obeyed by the hidden variables and is able to generate new samples.

2.2 Deep Modeling Strategy

Theoretically, as the depth of the neural network increases, the model is able to extract more complex features and obtain better experimental results. In practice, the model performance is usually degraded by the increase in network depth, leading to saturation or even a decrease in accuracy, while the gradient may disappear during the training process. To solve the above problems, ResNet [14] introduces residual units, which sum up the inputs and outputs of each layer

to realize cross-layer connections and improve the gradient updating of the deep model. DenseNet [15] uses dense connections, the inputs of each layer are derived from the outputs of all the previous layers, to improve the problem of vanishing gradient. Compared with ResNet, DenseNet is able to retain the feature maps of multiple receptive fields and utilize the feature information more fully, but the dense connection increases the input channels, which leads to a serious reduction in the computational efficiency of the model. VoVnet [11] uses One-Shot aggregation to aggregate all feature maps to the last layer, which solves the problem of inefficiency of the dense connection while inheriting the advantages of DenseNet. The above method significantly improves the problem of gradient disappearance of CNN with increasing depth, and this paper draws on the above modeling ideas to improve the Graph representation learning model based on deep GNN encoder.

For deep neural network models, choosing a suitable activation function can likewise alleviate the gradient vanishing problem. For example, the Rectified Linear Unit (ReLU) [16]:

$$\mathrm{ReLU}(x)=\begin{cases}\max(0,x), & x\geq 0\\ 0, & x<0\end{cases} \tag{3}$$

ReLU overcomes the gradient vanishing problem and dramatically improves the computational speed of the model, but negative gradients are set to zero at $x<0$, causing neurons to necrose and no longer respond to any data. ELU introduces an exponential function on top of ReLU, allowing it to return information even with negative input values:

$$\mathrm{ELU}(x)=\begin{cases}x, & x\geq 0\\ \alpha(\mathrm{e}^{x}-1), & x<0\end{cases} \tag{4}$$

Compared to ReLU, ELU has a certain output when the input is negative, thus eliminating the problem of neuronal necrosis in ReLU. In addition, the output mean of ELU is close to 0, which reduces the bias effect and makes the normal gradient close to the natural gradient; the negative value can saturate quickly when the input is small and is robust to noise.

2.3 Second-order Similarity of Graphs

The second-order similarity of a graph indicates the similarity of nodes' neighborhood structures, reflecting the global structural information of the graph, while the first-order similarity of a graph indicates the pairwise proximity between nodes connected by edges, reflecting the local structural information of the graph. Due to the sparsity of graphs, many links in real-world datasets are unobserved or missing, making it difficult to adequately represent the structural features of graphs using only first-order similarity information. In order to solve the topology and sparsity preservation problems, second-order similarity information needs to be introduced into the learning process of the model. In contrast to first-order similarity, second-order similarity is not determined by visible edges, but by the neighborhood structure shared by the nodes. Simply put, first-order similarity can be interpreted as higher similarity of connected nodes, and second-order similarity can be interpreted as higher similarity of nodes with similar neighborhood structure. Many sociological principles reflect the properties of second-order similarity, the strength of a connection between two people in a social network is related to the degree of overlap in their friendship networks [17], people who have many friends in common are likely to have the same interests and become friends. The introduction of second-order similarity can provide more information as a complement to first-order similarity when characterizing the topology.

3 GRAPH REPRESENTATION LEARNING MODEL

This section proposes OS-SeVAE and OS-SeAE, graph representation learning models based on One-Shot aggregation and second-order information, and discusses their algorithmic principles. This section first introduces the overall framework of the models, then proposes SeGCN that preserves second-order similarity, constructs the model encoder and decoder, and finally discusses the loss function of the models.

3.1 Overall Framework

The structure of OS-SeVAE and OS-SeAE is mainly divided into two parts: the encoder, which takes node attribute matrix $\boldsymbol{X}$ and neighbor matrix $\boldsymbol{A}$ as input, and the decoder, which reconstructs the neighbor matrix as output, to realize the original graph feature extraction and data dimensionality reduction, and to generate the low-dimensional embedding representation of each node.OS-SeVAE is constructed with the VAE as the basic framework, and the encoder part is constructed by using the multilayer GCN and SeGCN based on One-Shot OS-SeVAE is constructed based on VAE, and the encoder part uses multilayer GCN and SeGCN based on One-Shot aggregation to generate the mean encodingthe $\boldsymbol{\mu}$ and variance encoding $\boldsymbol{\sigma}$, and then superimposes $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ to generate the embedding matrix, and the decoder part uses the inner product of the embedding matrix to reconstruct the neighboring matrices. OS-SeAE is constructed based on ordinary AE, and the encoder part uses multilayer SeGCN based on One-Shot aggregation to generate the embedding matrix, and the decoder part uses the inner product of the embedding matrix to reconstruct the neighboring matrix. The depth of OS-SeAE and OS-SeVAE encoders is determined by the number of stacked Graph Layers (including SeGCN Layer and GCN Layer).

3.2 Second-order Similarity Graph Convolution

In this section, a novel graph convolutional network, SeGCN, is proposed to capture the second-order similarity of graphs, preserving the global structural information of the original graph. The original graph convolution model is usually defined as a linear shift-invariant operation of the adjacency matrix $\boldsymbol{A}$, and a neural network model is constructed using polynomials based on the adjacency matrix $f_{\boldsymbol{\Theta}}(\boldsymbol{x}, \boldsymbol{A})$ [18]:

$$f_{\boldsymbol{\Theta}}(\boldsymbol{x}, \boldsymbol{A}) = \sum_{k=0}^{K} \boldsymbol{\theta}_k \boldsymbol{A}^k \boldsymbol{x} \tag{5}$$

where, $\boldsymbol{\Theta} = (\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K)^T$, denotes the weights of different order neighbor matrix kernels. At this point, the graph convolution is expressed as a linear combination of the aggregated features of different order adjacency matrices. Kipf et al. [19] further simplified on this basis and proposed the neural network model for aggregating first-order neighborhood information $f_{\boldsymbol{1}}(\boldsymbol{x}, \boldsymbol{A})$:

$$f_{\boldsymbol{1}}(\boldsymbol{x}, \boldsymbol{A}) = \theta(\boldsymbol{I}_N + \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}) \boldsymbol{x} \tag{6}$$

where, $\boldsymbol{D}$ is the diagonal matrix of $\boldsymbol{A}$. Since it is difficult to adequately represent the global structural features of the graph by the adjacency matrix $\boldsymbol{A}$ containing first-order similarity, the squared adjacency matrix $\boldsymbol{A}^2$ containing second-order similarity is introduced as a complement to modify $f_{\boldsymbol{1}}(\boldsymbol{x}, \boldsymbol{A})$ to propose $f_{\boldsymbol{2}}(\boldsymbol{x}, \boldsymbol{A})$ that aggregates the first-order and second-order neighborhood information:

$$f_{\boldsymbol{2}}(\boldsymbol{x}, \boldsymbol{A}) = \theta\{\boldsymbol{I}_N + \boldsymbol{D}_{12}^{-\frac{1}{2}} [\boldsymbol{A} + \widehat{\boldsymbol{A}^2}] \boldsymbol{D}_{12}^{-\frac{1}{2}}\} \boldsymbol{x} \tag{7}$$

where, $\boldsymbol{D}_{12}$ is the diagonal matrix of $\boldsymbol{A} + \widehat{\boldsymbol{A}^2}$ and $\widehat{\boldsymbol{A}^2}$ denotes the symmetrically normalized second-order adjacency matrix.

$$\widehat{\boldsymbol{A}^2} = \boldsymbol{D}_2^{-\frac{1}{2}} \boldsymbol{A}^2 \boldsymbol{D}_2^{-\frac{1}{2}} \tag{8}$$

where $\boldsymbol{D}_2$ is the diagonal matrix of $\boldsymbol{A}^2$. Referring to the renormalization trick used by Kipf et al. in GCN, Eq. (7) is modified and the convolution expression for SeGCN is:

$$\boldsymbol{Z} = \tilde{\boldsymbol{D}}_{12}^{-\frac{1}{2}} \widetilde{\boldsymbol{A}_{12}} \tilde{\boldsymbol{D}}_{12}^{-\frac{1}{2}} \boldsymbol{X} \Theta \tag{9}$$

where, $\widetilde{\boldsymbol{A}_{12}} = \boldsymbol{I}_N + \boldsymbol{A} + \widehat{\boldsymbol{A}^2}$, $\tilde{\boldsymbol{D}}_{12}$ is the diagonal matrix of $\widetilde{\boldsymbol{A}_{12}}$, $\boldsymbol{X}$ is the input signal matrix, Θ is the filter parameter matrix, and $\boldsymbol{Z}$ is the convolution signal matrix. Compared with the original GCN, SeGCN introduces the second-order adjacency matrix as a supplement, which is able to aggregate the features of the second-order neighborhood of the nodes, thus retaining the second-order similarity information of the original graphs and improving the model characterization ability.

3.3 Encoder Structure

GCN and SeGCN utilize convolutional operations to extract features from the graph and generate feature vectors containing information about topology and node attributes, with interlayer propagation formulas, respectively:

$$\begin{aligned} \boldsymbol{H}_{GCN}^{(l+1)} &= \delta(\hat{\boldsymbol{A}} \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)}) \\ \boldsymbol{H}_{SeGCN}^{(l+1)} &= \delta(\widehat{\boldsymbol{A}_{12}} \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)}) \end{aligned} \tag{10}$$

where, $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}}$, $\widehat{\boldsymbol{A}_{12}} = \tilde{\boldsymbol{D}}_{12}^{-\frac{1}{2}} \widetilde{\boldsymbol{A}_{12}} \tilde{\boldsymbol{D}}_{12}^{-\frac{1}{2}}$, $\delta(\cdot)$ are the ELU activation functions, $\boldsymbol{W}^{(l)}$ is the trainable parameter matrix for each layer, and $\boldsymbol{H}^{(\cdot)}$ is the activation matrix for each layer (for l = 0, $\boldsymbol{H}^{(0)}$ is the input node attribute matrix $\boldsymbol{X}$). The encoder part of OS-SeAE extracts features using a multilayer SeGCN that introduces One-Shot aggregation to generate the embedding matrix $\boldsymbol{Y}$. The encoder part of OS-SeAE is used for the extraction of features:

$$\boldsymbol{Y} = \widehat{\boldsymbol{A}_{12}} (\boldsymbol{H}_{SeGCN}^{(1)} + \boldsymbol{H}_{SeGCN}^{(2)} + \cdots + \boldsymbol{H}_{SeGCN}^{(L)}) \boldsymbol{W}^{(\text{Final})} \tag{11}$$

where L is the number of layers of SeGCN in the encoder. The encoder part of OS-SeVAE generates the mean vector$\boldsymbol{\mu}$ using a multilayer GCN introduced into One-Shot aggregation and the variance vector $\boldsymbol{\sigma}$ using a multilayer SeGCN introduced into One-Shot aggregation:

$$\begin{aligned} \boldsymbol{\mu} &= \hat{\boldsymbol{A}} (\boldsymbol{H}_{GCN}^{(1)} + \boldsymbol{H}_{GCN}^{(2)} + \cdots + \boldsymbol{H}_{GCN}^{(L)}) \boldsymbol{W}^{(\mu)} \\ \ln \boldsymbol{\sigma} &= \widehat{\boldsymbol{A}_{12}} (\boldsymbol{H}_{SeGCN}^{(1)} + \boldsymbol{H}_{SeGCN}^{(2)} + \cdots + \boldsymbol{H}_{SeGCN}^{(L)}) \boldsymbol{W}^{(\sigma)} \end{aligned} \tag{12}$$

The sampling layer uses $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ to generate the embedding matrix $\boldsymbol{Y}$ from a Gaussian prior distribution:

$$\boldsymbol{Y} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon} \tag{13}$$

where, $\boldsymbol{\varepsilon} \sim \mathbb{N}(0, 1)$. Finally, the embedding is reparameterized as a distribution of probabilities over the latent space [11]:

$$q(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{A}) = \prod_{i=1}^{N} q(\boldsymbol{y}_i \mid \boldsymbol{X}, \boldsymbol{A})$$
$$q(\boldsymbol{y}_i \mid \boldsymbol{X}, \boldsymbol{A}) = \mathrm{N}\ (\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2)) \tag{14}$$

where, $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^{N}$, N is the number of nodes, and $\boldsymbol{y}_i$ is the low-dimensional embedding of node i.

3.4 Decoder Structure

For the OS-SeAE model, the decoder is a non-probabilistic model that reconstructs the adjacency matrix using the embedded inner product of two nodes:

$$\boldsymbol{A}' = \varphi(\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}) \tag{15}$$

where $\boldsymbol{A}'$ denotes the reconstructed adjacency matrix and φ denotes the sigmoid function. For the OS-SeVAE model, the decoder is the probabilistic model of reconstructing the adjacency matrix using the embedded inner product of two nodes:

$$p(\boldsymbol{A} \mid \boldsymbol{Y}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{ij} \mid \boldsymbol{y}_i, \boldsymbol{y}_j) \tag{16}$$

where $\boldsymbol{A}_{ij}$ is an element of the adjacency matrix $\boldsymbol{A}$.

3.5 Loss function

OS-SeAE is trained by minimizing the reconstruction loss of $\boldsymbol{A}$ and $\boldsymbol{A}'$ with the expression:

$$\mathrm{L}_{OS\text{-}SeAE} = \mathbb{E}_{q(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{A})}[\ln p(\boldsymbol{A} \mid \boldsymbol{Y})] + \mathrm{L}_{reg} \tag{17}$$

OS-SeVAE is trained by maximizing the variational lower bound as well as minimizing the reconstruction loss with the expression:

$$\mathrm{L}_{OS\text{-}SeVAE} = \mathbb{E}_{q(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{A})}[\ln(\boldsymbol{A} \mid \boldsymbol{Y})] - \mathrm{KL}[q(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{A}) \,\|\, p(\boldsymbol{Y})] + \mathrm{L}_{reg} \tag{18}$$

where $\mathrm{KL}[q(\cdot)||p(\cdot)]$ is the KL scatter of $q(\cdot)$ and $p(\cdot)$, the

$$p(\boldsymbol{Y}) = \prod_{i} p(\boldsymbol{y}_i) = \prod_{i} \mathrm{N}\ (\boldsymbol{y}_i \mid 0, \boldsymbol{I}) \tag{19}$$

In order to avoid parameter overfitting due to increased model depth, an L2 normalization regularization term is introduced in the OS-SeVAE and OS-SeAE loss functions L_{reg} :

$$\mathrm{L}_{reg} = \frac{1}{2} \sum_{k=1}^{K} \left\| \boldsymbol{W}^{(k)} \right\|_F^2 \tag{20}$$

During training, the input and output dimensions of the encoder's GCN and SeGCN layers must be the same in order to use One-Shot aggregation. In addition, both OS-SeVAE and OS-SeAE perform full batch gradient descent and are trained using the reparameterization trick.

4 EXPERIMENT

4.1 Dataset

In this paper, we use three benchmark citation network datasets, Cora, Citeseer, and Pubmed [20], to evaluate the performance of low-dimensional embedding representations generated by OS-SeVAE and OS-SeAE in a link prediction task. In the datasets, nodes denote papers, edges denote citations of one paper to another, node features are bag-of-words representations of papers, and node labels are manually set academic topics of papers. Table 1 shows the statistics of the three datasets.

Table 1 Statistical Information of Citation Networks

Dataset	#Nodes	#Edges	#Attributes	#Labels
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
Pubmed	19717	44338	500	3

4.2 Baseline

This paper uses the following model as a baseline:

- VGAE: This model migrates VAE to graph representation learning with the basic idea of using GCN to obtain the probability distribution of node representations, then sampling in the distribution to generate the node representations, and finally reconstructing the adjacency matrix of the graph using inner product decoding.

- GAE: This model directly generates node embeddings using a GCN-based encoder and then reconstructs the adjacency matrix using an inner product decoder.
- VGNAE [21]: This model replaces the GCN encoder in VGAE using the Graph Normalized Convolutional Network (GNCN) [21] which introduces L2 normalization, and the decoder is the same as VGAE.
- GNAE [21]: This model replaces the GCN encoder in GAE using the GNCN that introduces the L2 normalization, and the decoder is the same as in GAE.
- Linear-VAE [22]: This model replaces the GCN encoder in VGAE using a simple linear model of the normalized adjacency matrix, with the same decoder as in VGAE.
- Linear-AE [22]: This model replaces the GCN encoder in GAE using a simple linear model of the normalized adjacency matrix with the same decoder as in GAE.
- Res-VGAE [23]: This model introduces residual connectivity in multilayer GCN encoders to improve the experimental performance of deep VGAE models.
- Res-GAE [23]: This model introduces residual connectivity in multilayer GCN encoders to improve the experimental performance of deep GAE models.
- OSA-VGAE [24]: This model introduces One-Shot aggregation in multilayer GCN encoders to improve the experimental performance of deep VGAE models.
- OSA-GAE [24]: This model introduces One-Shot aggregation in multilayer GCN encoders to improve the experimental performance of deep GAE models.

4.3 Evaluation Metrics

The link prediction task compares the model performance based on the ability of the model to correctly classify edges and non-edges, and usually adopts the Area Under the Curve (AUC) and Average Precision (AP) as the evaluation metrics.The AUC is calculated by setting the threshold value immediately below each positive example, calculating the checking rate of the negative class, and then taking the average value. The AUC is calculated by setting the threshold immediately below each positive example, calculating the detection rate of the positive category, and then taking the average value; AUC is calculated by taking into account the classifier's ability to categorize both positive and negative examples, which can still provide a reasonable evaluation of the classifier in the case of sample imbalance, while AP is used to measure the model's classification performance on each category.

4.4 Link Prediction

In order to validate the experimental performance of the model in the link prediction task, the benchmark citation network dataset needs to be preprocessed by (1) retaining the attribute information of all nodes and removing some edges from the graph; (2) randomly sampling the number of node pairs without edges other than the removed edges with the same number of edges as the number of edges previously removed; and (3) dividing the removed edges and edgeless nodes into a validation set and a test set, which contain, respectively, 5 % and 10% real links and an equal number of edgeless node pairs. All models are trained using the preprocessed dataset, and then the neighbor matrix is reconstructed using the node embedding inner product for link prediction.
OS-SeVAE, OS-SeAE, and each baseline model were developed from VGAE and GAE.To ensure the fairness and consistency of the experiments, all the models were initialized according to the parameters suggested in the original papers of VGAE and GAE.The dimensionality of the hidden layer was set to 32, the dimensionality of the generative embedding was set to 16, and the training was performed using Adam's optimizer[97] with the learning rate set to 0.01, and the number of iterations was set to 200. Each model was trained 10 times, and the dataset was randomly initialized for each training, and the mean and standard deviation of the AUC and AP scores (%) were recorded. The experimental results of link prediction are shown in Tables 2 to 4. From the results, the following analysis is made:
(1) In most cases, the experimental results predicted by the OS-SeVAE and OS-SeAE links outperform the baseline model at the same depth with smaller standard deviations, especially on the Cora and Citeseer datasets, where the mean values of AUC and AP scores are consistently higher than those of the baseline model for OS-SeVAE and OS-SeAE. The above results indicate that the One-Shot aggregation and ELU function can significantly improve the deep model performance to avoid gradient vanishing and oversmoothing. In addition, SeGCN, which preserves second-order information, enhances the model's ability to characterize the topology, further improving the experimental performance of the link prediction task.
(2) Table 2 compares the experimental results of different models using 1 layer. On the three datasets, OS-SeVAE and OS-SeAE have the highest AUC and AP, OSA-VGAE and OSA-VGAE using the same deep layer strategy are the next highest, Linear-VAE and Linear-AE using linear coding have the worst experimental performance, and the other baseline models are very close in terms of AUC and AP. The experimental results show that the use of One-Shot aggregation, ELU activation function and SeGCN encoder in the shallow model can improve the experimental performance of the model.
(3) Table 3 compares the experimental results of different models using the three layers. On the three datasets, Linear-VAE and Linear-AE using the linear encoder have the worst experimental performance, Res-VGAE and Res-GAE introducing the residual linkage perform slightly better than the model with the direct superimposition of the GCN layer, VGNAE and GNAE using the GNCN have similar performance and less degradation than the residual model, OSA-

VGAE, OSA-GAE, OSA-SeVAE and OSA-SeVAE using the One-Shot aggregation and the ELU activation function of OSA-VGAE, OSA-GAE, OS-SeVAE and OS-SeAE perform significantly better than the other models. The experimental results show that the normalized operation of GNCN makes the model performance does not show significant changes with the increase of depth, the residual connection can improve the performance of the deep model to a certain extent, and the One-Shot aggregation and ELU activation function can effectively improve the deep modelability. In addition, comparing OSA-VGAE and OSA-GAE, OS-SeVAE and OS-SeAE using SeGCN encoder are able to extract the second-order information of the graph during the embedding generation process, which further improves the experimental performance of the link prediction task.

(4) Table 4 compares the experimental results of different models using six layers. On the three datasets, the experimental performance of VGAE, GAE, Linear-VAE, Linear-AE, Res-VGAE, and Res-GAE without the use of the normalization operation, One-Shot aggregation, and ELU activation function drops sharply, the experimental performance of VGNAE and GNAE with the use of the normalization operation decreases marginally, and OSA-VGAE, OSA-GAE, OS-SeVAE and OS-SeAE perform well in the majority of cases. The experimental results show that the performance of the models without normalization operation and deep strategy decreases sharply due to the increase of depth, the residual linkage improves the deep model very limitedly, the normalization operation can effectively keep the stability of the experimental results, and the One-Shot aggregation and ELU activation function can significantly improve the performance of the deep model.

Table 2 AUC and AP Scores (1-Graph Layer)

Model	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
GAE	89.49±0.95	91.23±0.85	88.27±1.18	90.54±0.98	95.10±0.26	95.82±0.20
VGAE	90.22±0.89	91.48±0.71	88.35±0.91	90.02±0.60	95.34±0.34	95.82±0.22
GNAE	88.73±0.34	89.74±0.60	89.13±0.70	90.61±0.60	93.26±0.22	93.79±0.25
VGNAE	89.22±0.50	90.11±0.61	90.43±0.82	91.54±0.81	93.42±0.34	93.92±0.36
Linear-AE	86.38±0.93	88.38±0.76	82.94±0.96	85.79±1.41	91.64±1.80	92.03±1.67
Linear-VAE	87.45±0.95	88.74±0.76	85.50±1.17	86.93±1.14	93.52±0.26	93.86±0.26
Res-GAE	90.34±1.25	91.20±0.73	88.86±0.55	89.70±0.80	94.26±1.07	95.05±0.86
Res-VGAE	90.84±0.48	92.05±0.39	89.63±1.30	91.24±1.13	95.39±0.23	95.77±0.18
OSA-GAE	93.09±0.46	93.64±0.60	93.20±0.39	93.81±0.51	95.91±0.53	96.11±0.49
OSA-VGAE	93.16±0.68	93.46±0.62	93.37±0.46	93.87±0.44	95.35±0.94	95.52±0.77
OS-SeAE	93.39±0.69	93.90±0.63	93.50±0.55	94.15±0.55	96.26±0.11	96.50±0.08
OS-SeVAE	93.48±0.62	94.04±0.51	93.44±0.38	94.19±0.37	96.50±0.24	96.73±0.28

Table 3 AUC and AP Scores (3-Graph Layer)

Model	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
GAE	87.44±1.13	89.62±0.93	84.64±1.04	87.79±0.82	91.62±0.90	93.42±0.61
VGAE	87.30±1.23	88.68±1.00	85.54±0.60	87.77±0.77	92.77±0.66	94.12±0.45
GNAE	87.74±0.48	89.06±0.49	88.24±0.83	89.96±0.73	92.79±0.24	93.40±0.27
VGNAE	88.15±0.32	89.40±0.74	89.44±0.85	90.68±0.68	92.99±0.24	93.60±0.26
Linear-AE	83.51±0.83	83.51±0.83	80.23±1.03	83.36±1.04	90.07±1.64	91.18±1.44
Linear-VAE	84.00±1.36	85.62±0.90	81.83±1.15	83.89±1.17	91.80±0.25	92.34±0.21
Res-GAE	88.43±1.11	90.19±0.72	85.95±1.23	88.44±0.91	92.89±0.39	94.13±0.38
Res-VGAE	89.19±0.64	90.35±0.64	86.31±1.16	88.28±1.02	93.03±1.33	93.98±1.09
OSA-GAE	92.47±0.78	92.90±0.76	92.40±0.68	93.18±0.53	94.22±1.21	94.58±1.08
OSA-VGAE	92.14±1.09	92.54±0.85	92.14±0.35	92.69±0.56	95.18±0.44	95.30±0.49
OS-SeAE	92.83±0.95	93.20±0.88	92.75±0.58	93.51±0.50	95.72±0.17	96.00±0.18
OS-SeVAE	93.10±0.70	93.58±0.80	92.89±0.71	93.71±0.62	95.37±0.97	95.66±0.78

Table 4 AUC and AP Scores (6-Graph Layer)

Model	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
GAE	82.97±1.80	84.90±1.67	80.79±1.74	83.94±1.88	88.72±1.25	91.22±0.97
VGAE	84.44±1.12	85.63±1.04	83.25±1.29	85.30±0.98	88.88±1.95	91.29±1.11

GNAE	87.26±0.59	88.72±0.61	87.72±0.78	89.59±0.67	92.51±0.23	93.21±0.30
VGNAE	87.63±0.32	89.07±0.66	88.85±0.75	90.27±0.64	92.71±0.29	93.38±0.32
Linear-AE	79.51±0.94	80.99±1.06	77.20±1.29	80.38±1.16	88.89±1.68	90.00±1.64
Linear-VAE	79.17±5.08	79.93±5.02	77.83±1.68	79.06±1.81	89.10±1.73	90.08±1.68
Res-GAE	85.55±1.19	87.20±1.27	82.34±1.36	85.66±1.11	89.02±1.94	91.25±1.38
Res-VGAE	86.06±1.61	87.06±2.22	83.18±1.72	85.07±1.66	87.51±2.76	88.96±3.15
OSA-GAE	91.61±0.73	92.06±0.89	91.24±0.65	91.83±0.73	91.95±1.47	92.78±1.28
OSA-VGAE	89.35±2.98	89.71±2.55	90.38±1.76	91.04±1.55	93.82±1.35	93.94±1.25
OS-SeAE	92.08±0.61	92.46±0.57	91.81±0.78	92.52±0.88	95.27±0.35	95.57±0.35
OS-SeVAE	92.61±0.97	92.96±0.92	92.97±0.65	93.58±0.56	93.39±1.26	93.97±0.98

Figures 1 shows the AUC and AP scores of different models on the cora dataset from layer 1 to 6. Compared with VGAE and GAE, with the increase of depth, Linear-VAE and Linear-AE using linear encoder cannot effectively extract and retain the original graph-related features, and the performance decreases the fastest, and the experimental performance is the worst; Res-VGAE and Res-GAE introducing residual linkage alleviate the problem of performance degradation of deeper models to a certain degree, but the experimental performance is VGAE and GAE are very close; using the normalization operation VGNAE and GNAE have lower performance than VGAE and GAE in the shallow layer, but with the gradual increase of the depth, the experimental performance of VGAE and GAE decreases rapidly, while the performance of VGNAE and GNAE decreases to a lesser extent, and the experimental performance is gradually better than that of VGAE and GAE with the depth increase; the experimental performance of OSA using the One-Shot aggregation and the ELU activation functions, OSA-VGAE, OSA-GAE, OS-SeVAE, and OS-SeAE significantly not only improve the experimental performance of the deep model, but also outperform VGAE and GAE in the shallow model. In addition, comparing OSA-VGAE, OSA-GAE, OS-SeVAE, and OS-SeAE with the same deep strategy, the OS-SeVAE using the SeGCN encoder outperforms OSA-VGAE and OSA-GAE in most cases, and OS-SeAE consistently outperforms OSA-VGAE, OSA-GAE, and the other baseline models. The above experimental results further demonstrate that the use of One-Shot aggregation and ELU function can improve the gradient information transfer and over-smoothing problems of deep models, preventing the model performance from decreasing dramatically due to the increase in depth, while the encoder constructed using SeGCN can enhance the model representation ability and improve the model performance.

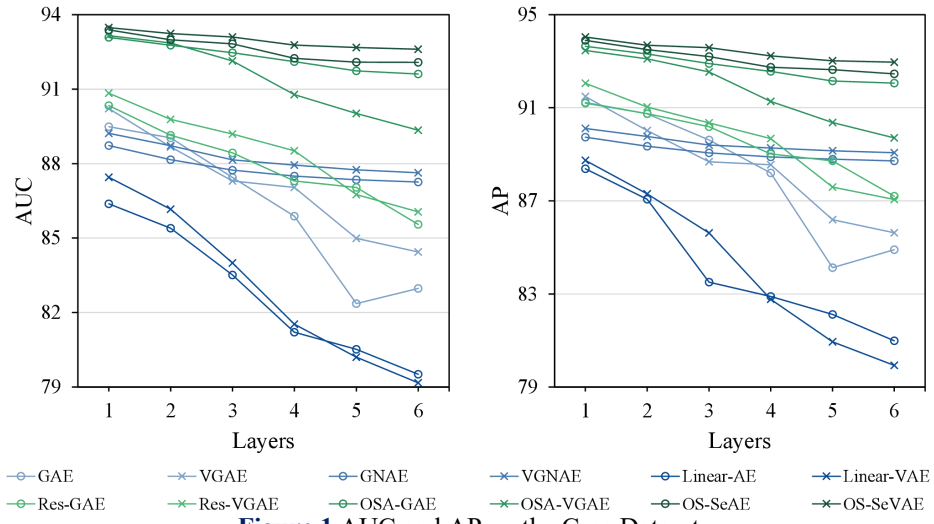


Figure 1 AUC and AP on the Cora Dataset

4.5 Ablation Study

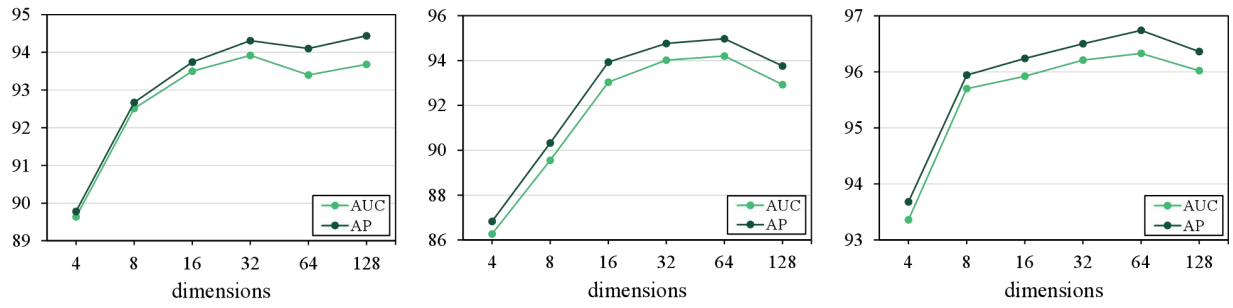
In order to validate the effect of using One-Shot aggregation and ELU functions in the OS-SeAE and OS-SeVAE models on the performance of the algorithms, ablation experiments were performed on the Cora dataset, where the mean values of the AUC and AP scores were recorded, comparing the performance of the variants that used the One-Shot aggregation and the ELU functions alone. The variants using only One-Shot aggregation are SeAE-OSA and SeVAE-OSA, and the variants using only the ELU function are SeAE-ELU and SeVAE-ELU.To ensure fairness of the experiments, the parameters of the learning rate, number of iterations, hidden layer dimensions, and embedding dimensions are kept consistent. The results of the ablation experiments are shown in Table 5. Compared to the variants using One-Shot aggregation or ELU function alone, OS-SeAE and OS-SeVAE obtain the best performance, proving that the simultaneous use of the above two modules can further improve the deep model performance.

Table 5 Results of Ablation Experiments

Model	1 layer		2 layers		3 layers		4 layers		5 layers		6 layers	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
SeAE-OSA	92.04	92.08	91.84	92.03	91.45	91.73	91.75	92.01	91.81	92.03	91.56	91.88
SeAE-ELU	92.96	93.36	92.52	92.74	92.21	92.50	91.76	92.02	91.08	91.29	90.14	90.61
OS-SeAE	93.39	93.90	92.99	93.50	92.83	93.20	92.24	92.74	92.09	92.64	92.08	92.46
SeVAE-OSA	93.26	93.62	93.20	93.49	92.98	93.28	92.74	93.05	92.46	92.75	92.50	92.86
SeVAE-ELU	93.10	93.57	92.74	93.11	92.37	92.65	92.19	92.47	91.26	91.69	91.18	91.62
OS-SeVAE	93.48	94.04	93.24	93.69	93.10	93.58	92.78	93.23	92.68	93.02	92.61	92.96

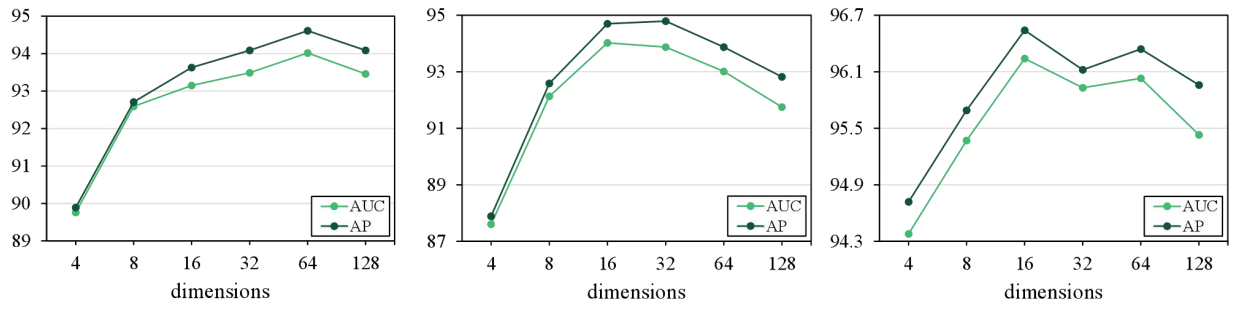
4.6 Hyperparameter Study

Among the parameters of OS-SeAE and OS-SeVAE, the embedding dimension is an important parameter for Graph representation learning models compared to the common parameters such as learning rate and number of iterations. Too small or too large a dimension can significantly affect the model performance [25]. In order to further explore the relationship between embedding dimension and model performance, parameter experiments were conducted using OS-SeAE and OS-SeVAE (learning rate of 0.01, number of iterations of 200, and hidden layer dimension of 200) with 1 graph layer. Figures 2 and 3 record the AUC and AP scores on the three citation datasets using different dimensions. From Figures 2 and 3, it can be seen that initially AUC and AP rise with increasing dimensionality, which is due to the fact that more dimensions encode more useful information about the original data in the embedding and enhance the experimental performance. However, with further increase in dimensionality, AUC and AP fluctuate, and higher dimensionality does not lead to better experimental results, this is because the number of training samples is limited, and for each class of nodes there exists an optimal embedding dimensionality that maximizes the model performance, and when the actual dimensionality exceeds the optimal dimensionality, noisy information is encoded in the embedding, leading to a decrease in model performance. Therefore, it is very important to choose the appropriate dimension when generating node embeddings. Overall, when observing and analyzing Figures 2 and 3, when the embedding dimensions are set at 16~64, the performance of OS-SeAE and OS-SeVAE is relatively less affected by the embedding dimensions, and at the same time, they are able to maintain relatively good experimental performance.



(a) Cora (b) Citeseer (c) Pubmed

Figure 2 OS-SeAE Dimensionality Experiments



(a) Cora (b) Citeseer (c) Pubmed

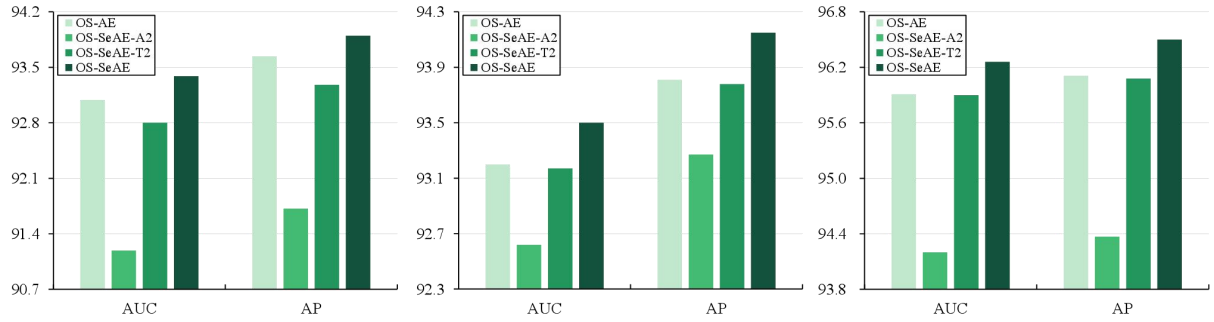
Figure 3 OS-SeVAE Dimension Experiments

4.7 Variant Analysis

To further validate the effectiveness of the symmetric normalized second-order adjacency matrix $\widehat{A^2}$ used by SeGCN, experiments were conducted in the link prediction task on the three citation datasets in comparison with the variant models that do not use second-order information as well as those that use other ways of retaining the second-order information (1 layer and with the same parameters). Among other things, the relevant information for each variant is as follows:

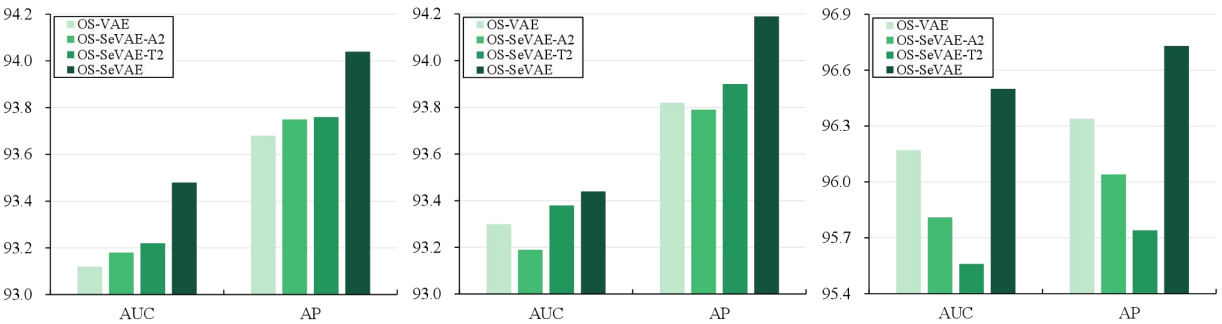
- **OS-AE, OS-VAE:** Replace the modeled SeGCN encoder with a GCN that does not use second-order information.
- **OS-SeAE-A2, OS-SeVAE-A2:** Modify SeGCN in OS-SeAE and OS-SeVAE encoders by replacing $\widehat{A^2}$ with square A of the adjacency matrix.
- **OS-SeAE-T2, OS-SeVAE-T2:** Modify SeGCN in OS-SeAE and OS-SeVAE encoders by replacing $\widehat{A^2}$ with the second-order transfer probability matrix T^2 [26].

From the results of the variants performance experiments in Figures 4 and 5, there is the following analysis: (1) on all three datasets, the experimental results of OS-SeAE and OS-SeVAE using SeGCN consistently outperform the other three variants, demonstrating that SeGCN with the introduction of symmetric normalized second-order adjacency matrices characterizes the graph structure better than the original GCN and outperforms the other methods that retain second-order information; (2) comparing the variants without second-order information (OS-AE and OS-VAE) and with other ways of retaining second-order information (OS-SeAE-A2, OS-SeVAE-A2, OS-SeAE-T2, and OS-SeVAE-T2), the experimental performances of OS-AE and OS-VAE are better in most of the cases, which proves that different ways of retaining second-order information have different effects on model performance. This proves that different second-order information retention methods have different effects on the model performance, and that appropriate second-order information retention methods can enhance the model characterization ability and improve the experimental performance, while inappropriate second-order information retention methods not only fail to improve the model performance, but also perform lower than that of the GCN that only uses first-order information; (3) comparing $\widehat{A^2}$, A^2 and T^2, the symmetric normalization operation retains the symmetry of the square of the adjacency matrix as well as normalization, which not only retains the symmetry of A when superimposed with the adjacency matrix A, but also avoids the interference of the original adjacency relationship by the oversized value in A^2, thus effectively retaining the second-order information of the original diagram.



(a) Cora (b) Citeseer (c) Pubmed

Figure 4 OS-SeAE and its Variants



(a) Cora (b) Citeseer (c) Pubmed

Figure 5 OS-SeVAE and its Variants

5 CONCLUSION

The OS-SeVAE and OS-SeAE models based on One-Shot aggregation, ELU activation function and SeGCN proposed in this paper retain richer structural information and enhance the representation ability of the models while improving the gradient information transfer of the deep models and avoiding the occurrence of over-smoothing in the deep GCN encoder. The experimental results show that it is beneficial to introduce deep strategies in computer vision to Graph representation learning models, which can effectively improve the experimental performance of deep models. In the ablation experiments, the simultaneous use of One-Shot aggregation and ELU function improves the performance more significantly than a single strategy. Meanwhile, the experimental performance of OS-SeVAE and OS-SeAE is further improved compared to OSA-VGAE and OSA-GAE using only GCN encoder, proving the effectiveness of SeGCN. In future work, in addition to improving the structure of the existing encoder models, more efficient neighborhood aggregation and neighborhood interaction encoder modeling, such as the attention mechanism-based approach, will be

used; in the decoder part, experiments with different decoders and probability distributions will be attempted. In addition, subsequent work will quantify and analyze the model complexity, model generalization ability, and model avoidance of overfitting ability.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

FUNDING

This work was supported in part by the Social Science Fund of Guangxi under Grant 23FTQ005.

REFERENCES

[1] BKhoshraftar Shima, An Aijun. A survey on graph representation learning methods. ACM Transactions on Intelligent Systems and Technology, 2024, 15(1): 1-55.
[2] LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning. nature, 2015, 521(7553): 436-444.
[3] Xia Feng, Liu Jiaying, Nie Hansong, et al. Random walks: A review of algorithms and applications. IEEE Transactions on Emerging Topics in Computational Intelligence, 2019, 4(2): 95-107.
[4] Zhang Changqing, Geng Yu, Han Zongbo, et al. Autoencoder in autoencoder networks. IEEE transactions on neural networks and learning systems, 2022, 35(2): 2263-2275.
[5] Wang Yile, Cui Leyang, Zhang Yue. Improving skip-gram embeddings using BERT. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2021, 29: 1318-1328.
[6] Zhou Jingya, Liu Ling, Wei Wenqi, et al. Network representation learning: From preprocessing, feature extraction to node embedding. ACM Computing Surveys, 2022, 55(2): 1-35.
[7] Liu Meng, Gao Hongyang, Ji Shuiwang. Towards deeper graph neural networks. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2020: 338-348.
[8] Li Guohao, Muller Matthias, Thabet Ali, et al. Deepgcns: Can gcns go as deep as cnns?. Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision. Piscataway: IEEE, 2019: 9267-9276.
[9] Li Qimai, Han Zhichao, Wu Xiaoming. Deeper insights into graph convolutional networks for semi-supervised learning. Proceedings of the 32th AAAI Conference on Artificial Intelligence. Menlo Park: AAAI, 2018:3538–3545.
[10] Zhang Si, Tong Hanghang, Xu Jiejun, et al. Graph convolutional networks: a comprehensive review. Computational Social Networks, 2019, 6(1): 1-23.
[11] Lee Youngwan, Hwang Joongwon, Lee Sangrok, et al. An energy and gpu-computation efficient backbone network for real-time object detection. Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. Piscataway: IEEE, 2019: 752–760.
[12] Clevert Djork-Arné, Unterthiner Thomas, Hochreiter Sepp. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv: 1511.07289, 2015.
[13] Kullback S., Leibler R. A. On information and sufficiency. Annals of Mathematical Statistics, 1951, 22(1):79-86.
[14] He Kaiming, Zhang Xiangyu, Ren Shaoqing. Deep residual learning for image recognition. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2016: 770-778.
[15] Huang Gao, Liu Zhuang, Maaten Laurens van der. Densely connected convolutional networks. Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2017: 4700-4708.
[16] Glorot Xavier, Bordes Antoine, Bengio Yoshua. Deep sparse rectifier neural networks. Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR: Boston, 2011, 15: 315-323.
[17] Firth J R. A synopsis of linguistic theory, 1930-1955. Studies in Linguistic Analysis, 1957: 1-32.
[18] Defferrard Michaël, Bresson Xavier, Vandergheynst Pierre. Convolutional neural networks on graphs with fast localized spectral filtering. Proceedings of the 30th International Conference on Neural Information Processing Systems. New York: ACM, 2016: 3844-3852.
[19] Kipf Thomas N., Welling Max. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv: 1609.02907, 2016.
[20] Namata Galileo, London Ben, Getoor Lise, et al. Query-driven active surveying for collective classification. Proceedings of the 10th International Workshop on Mining and Learning with Graphs. New York: ACM, 2012.
[21] Ahn Seong Jin, Kim MyoungHo. Variational Graph Normalized Autoencoders. Proceedings of the 30th ACM International Conference on Information and Knowledge Management. New York: ACM, 2021: 2827-2831.
[22] Salha Guillaume, Hennequin Romain, Vazirgiannis Michalis. Simple and effective graph autoencoders with one-hop linear models. Proceedings of the 2020 Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer: Berlin, 2020: 319-334.
[23] Nallbani Indrit, Ayanzadeh Aydin, Keser Reyhan Kevser, et al. Representation Learning using Graph Autoencoders with Residual Connections. arXiv preprint arXiv: 2105.00695, 2021.
[24] Yuan Lining, Liu Zhao. Graph representation learning by autoencoder with one-shot aggregation. Journal of Computer Applications, 2023, 43(1): 8-14.

[25] Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, et al. Why and When Can Deep-but Not Shallow-networks Avoid the Curse of Dimensionality: A Review. International Journal of Automation and Computing, 2017, 14(05): 503-519.

[26] Cao Shaosheng, Lu Wei, Xu Qiongkai. GraRep: learning graph representations with global structural information. Proceedings of the 24th ACM International Conference on Information and Knowledge Management. New York: ACM, 2015: 891-900.