Applied Intelligence (2025) 55:90
https://doi.org/10.1007/s10489-024-06063-z

Variational graph neural network with diffusion prior for link prediction

Hailong Su[1] · Zhipeng Li[2] · Chang-An Yuan[3] · F. Filaretov Vladimir[4] · De-Shuang Huang[5,6]

Accepted: 9 September 2024 / Published online: 6 December 2024

Abstract
Recently, Graph neural networks(GNNs) has achieved tremendous success in a variety of fields. Many approaches have been proposed to address data with graph structure. However, many of these are deterministic methods, therefore, they are unable to capture the uncertainty, which is inherent in the nature of graph data. Various VAE(Variational auto-encoder)-based approaches have been proposed to tackle such problems. Unfortunately, due to the simple a posterior and a prior assumption problems of such methods, they are not well suited to handle uncertainty in graph data. For example, VGAE(Variational graph auto-encoder) assumes that the posterior and prior distributions are simple Gaussian distributions, which can lead to overfitting problems when incompatible with the true distributions. Many methods propose to solve the posterior distribution problem, but most ignore the effect of the prior distribution. Therefore, in this paper, we proposed a novel method to solve the Gaussian prior problem. Specifically, in order to enhance the representation power of the prior distribution, we use the diffusion model to model the prior distribution. We incorporate the diffusion model into VGAE. In the forward diffusion process, noise is gradually added to the latent variables, and then the samples are recovered by the backward diffusion process. To realize the backward diffusion process, we propose a new denoising model which predicts noise by stacking GCN(Graph Convolution Network) and MLP(Multi-layers Perceptron). We perform experiments on different datasets and the experimental results demonstrate that our method obtains state-of-the-art results.

Keywords Graph neural network · Variational graph auto-encoder · Diffusion model · Denoising process · Prior distribution

1 Introduction

Convolution neural networks (CNNs) have achieved enormous success with Euclidean data [30], such as images [19], audio [20], and sequences [46]. Although CNNs have powerful feature extraction capabilities, they perform poorly in handling non-Euclidean data, like graph data [57]. This is because the translation invariance and locality assumptions are not applicable in graph data. However, in practice, there is a significant amount of data in the form of graphs. For example, social networks, where users represent nodes and relationships between users represent edges [10], and E-commerce networks, where products represent nodes and purchase relationships represent edges [31]. To effectively model such data, graph neural networks (GNNs) have been proposed [3, 11, 13, 27–29], with various architectures like Graph Convolutional Networks (GCNs) [8, 22, 25] and Graph Attention Networks (GATs) [50] demonstrating significant promise.

However, existing GNN methods are based on the assumption that the input graph is its ground truth. It is important to note that the graphs used in practical applications are often derived from noisy data or modeling assumptions. A graph may have spurious edges or missing edges between two nodes with strong relationships (see Fig. 1), named uncertainty in graph data. Hence, capturing uncertainty in graph data is critical, but traditional GCNs or GATs are deterministic approaches, lacking the capability to model uncertainty. This motivates the development of probabilistic GNNs to handle uncertainty in graph data more effectively. To solve this drawback, many methods have been proposed [24, 36, 60]. Unfortunately, these methods do not model uncertainty in graph data well. They assume that the prior distribution is a simple Gaussian distribution, which limits the flexibility of variational inference when the distribution of the graph data is incompatible with a Gaussian distribution, which makes the downstream task perform poorly. Additionally, numerous

Extended author information available on the last page of the article

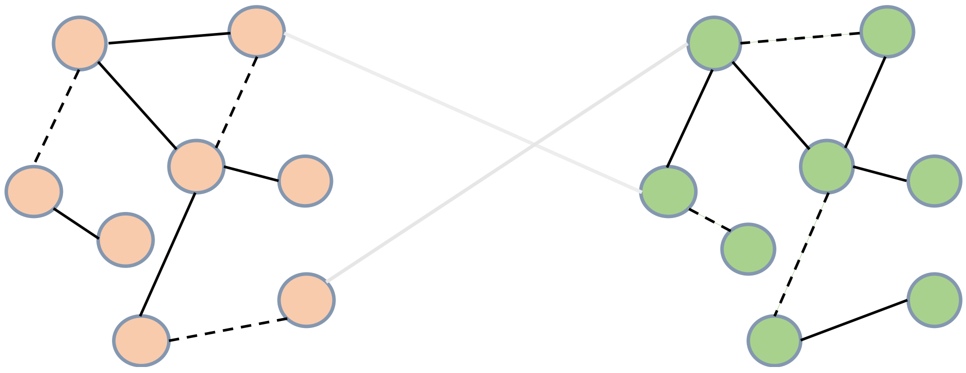
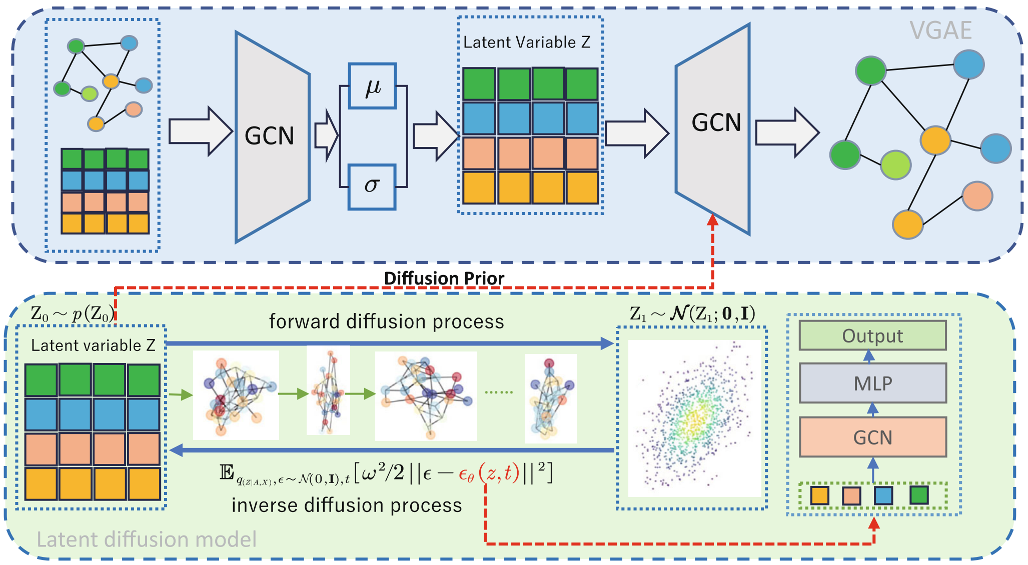


Fig. 1 Uncertainties in graph data. A solid gray line indicates that two nodes should not be connected. Dashed lines indicate missing connections and solid lines indicate valid connections

approaches have predominantly concentrated on enhancing the expressiveness of the posterior distribution, often overlooking the significance of the prior distribution.

To eliminate this drawback and obtain a more expressive prior distribution, we proposed a novel method based on diffusion model [21] in this paper. To the best of our knowledge, we are the first to improve the expressive power of prior distributions in VGAE. Specifically, we first embed the graph into latent space, then, the forward diffusion process in the diffusion model is formed by gradually adding noise to the latent variables, and then the noise is predicted by the denoising model to recover the clean latent variables. The role of diffusion model is modeling the distribution over embeddings of latent variables, hence, a more representational prior distribution can be obtained. Furthermore, we proposed a novel denoising model that is better suitable for graph data by stacking GCN layers and MLP. Finally, we conduct experiments on different datasets and the experimental results show that our method achieves promising results compared to other methods. Especially compared to VGAE, our method improves the AUC by more than 10% on some datasets. The diagram of our method is shown in Fig. 2. The contributions of our approach are summarized as follows:

1. **Integration of VGAE and Diffusion Model:** We introduce an innovative method that combines Variational Graph Auto-Encoders (VGAE) with diffusion models, advancing the learning of more pliable prior distributions. Unlike VGAE's assumption of a Normal distribution, our approach employs a diffusion model to create prior distributions that offer greater flexibility and have been demonstrated to enhance performance in subsequent tasks through empirical evidence.
2. **Denoising model for Graph Data:** To tailor the denoising model within the diffusion model for graph data, we have developed a novel architecture that predicts noise with heightened precision. This is achieved by integrating multiple layers of Graph Convolutional Networks


VGAE
GCN
μ
σ
Latent Variable Z
GCN
Diffusion Prior
$Z_0 \sim p(Z_0)$
Latent variable Z
forward diffusion process
$Z_1 \sim \mathcal{N}(Z_1; \mathbf{0}, \mathbf{I})$
$\mathbb{E}_{q_{(z|A,x)},\epsilon\sim\mathcal{N}(0,\mathbf{I}),t}[\omega^2/2||\epsilon-\epsilon_\theta(z,t)||^2]$
inverse diffusion process
Output
MLP
GCN
Latent diffusion model
Fig. 2 The flowchart of our method. The latent variable Z is obtained by the first GCN (i.e., encoder) and then the prior distribution is learned using the diffusion model. Once the prior distribution is learned, the final representation is then obtained using the second GCN (i.e., decoder)

(GCNs) and Multilayer Perceptrons (MLPs), which work in concert to refine the prediction of noise within the model.

3. **Addressing High-Dimensional Challenges:** Recognizing the high-dimensionality of graph data and the potential for a "curse of dimensionality" when applying diffusion models to raw data, we have devised a latent diffusion model. This model significantly decreases the time complexity of our approach. By operating the diffusion process in a reduced-dimensional latent space-instead of on the original data-we have streamlined the computational requirements compared to traditional diffusion models.
4. **Empirical Validation and Performance Improvement:** Through rigorous experimentation on various datasets focused on link prediction tasks, our method has achieved outstanding results. Notably, it significantly improves its performance metrics by more than 10% on certain datasets compared to the VGAE benchmark.

2 Related work

2.1 Graph neural network based methods

Graph Neural Networks (GNNs) have been extensively studied to handle non-Euclidean data structures [9]. Kipf et al. [25] proposed the Graph Convolutional Network (GCN), which is inspired by traditional CNNs. GCN is based on the first-order approximation of spectral graph convolution and can effectively learn both the structure and the node features of a graph. However, GCN requires the entire graph as input, leading to higher computational complexity. To address this, Wu et al. introduced a simplified version of GCN [55] that reduces computational complexity by removing nonlinear transformations and collapsing weight matrices between successive layers. Nevertheless, many GCNs are shallow: for example, the best results in node classification are typically achieved with 2-layer GCNs due to the over-smoothing problem. To enable deeper GCNs, Chen et al. proposed GCNII [6], which incorporates identity mapping and initial residuals to improve model depth.

The essence of GNN is to aggregate information from neighboring nodes, therefore, it is important to aggregate useful neighbor information. For this reason, Graph Attention Networks (GATs) was proposed by Veličković et al. [50] to improve the aggregation of neighbor information using attention mechanisms. GAT calculates the importance of neighboring nodes and assigns different weights to them, enabling more effective information aggregation. However, GAT is limited by its static attention mechanism, which may not perform well in all scenarios. To overcome this limitation, Brody et al. proposed GATv2 [4], which employs a dynamic attention mechanism, providing more flexibility and improved performance in certain tasks. For inductive frameworks that generalize GNNs to unseen nodes, Hamilton et al. proposed GraphSAGE [16], which learns an embedding function that generalizes to unseen nodes by utilizing node features.

2.2 Bayesian-based methods

GCN, GAT, etc. are deterministic methods and cannot capture uncertainty in graph data. To address these challenges, Bayesian approaches have been explored. Zhang et al. proposed Bayesian GCN (BGCN) [60], which integrates Bayesian theory with GCNs, treating the observed graph as a realization of a family of random graph parameters. However, this method heavily relies on the selection of random graphs. To address this issue, Pal et al. introduced a non-parametric BGCN [36], which utilizes non-parametric methods for posterior inference conditional on the observed graph, features, and labels. Other Bayesian-based methods have also been proposed for graph data [5, 18, 34].

2.3 Variational inference-based methods

Variational Graph Auto-Encoders (VGAE), based on Variational Auto-Encoders (VAE) [24, 40], have been proposed to learn latent representations of graphs by embedding nodes into random variables. However, VGAE assumes a Gaussian distribution for the variational distribution, limiting the flexibility of variational inference when the true posterior distribution deviates from the Gaussian assumption. SIG-VAE [17] enhances the expressiveness of the posterior distribution using semi-implicit variational inference [59], which allows for a more flexible generative model.

However, the majority of existing methods concentrate on enhancing the posterior distribution's expressiveness, often sidelining the critical role of the prior distribution. To bridge this gap, our proposed methodology pioneers the use of diffusion models to endow VGAEs with a more articulated prior distribution. This innovative approach represents a significant stride in augmenting the prior's capacity for expression within these models, potentially leading to more robust and flexible graph representations.

3 Preliminaries

In this paper, we denote graph as $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$, where $\mathcal{E}$ represent edges and $\mathcal{V}$ indicate nodes in the graph respectively. Furthermore, node features are represented as $\mathrm{X} \in \mathbb{R}^{m\times n}$, $m = |\mathcal{V}|$ is the number of nodes, and n denotes feature dimension. The graph is represented as symmetric adjacency

Table 1 Summary of Main Mathematical Symbols and Their Meanings

Symbol	Meaning
$\mathcal{G}$	The graph
$\mathcal{E}$	Edges of graph
$\mathcal{V}$	Nodes of graph
Z	Random variables in the latent space
A	Adjacency matrix of graph
X	Matrix of node features
μ	Mean of the latent variables
σ	Standard deviation of the latent variables
$\tilde{A}$	Normalized adjacency matrix
ζ	Activation function
θ	Parameters diffusion model
ϕ	Parameters of encoder
W_i	Shared parameters in GCN
β_t	Amount of noise added at each step
T	Diffusion modeling steps
ϵ_θ	Noise prediction model
σ_t	Noise variance at time step t
H_i^0	Graph embedding

matrix $\mathrm{A} \in \mathbb{R}^{m\times m}$, where $\mathrm{A}_{i,j} = 1$ when node i and j are connected by an edge, and vice versa $\mathrm{A}_{i,j} = 0$. For the latent variable model, we define a latent variable $\mathrm{Z} \in \mathbb{R}^{m\times h}$, where h is the dimension of Z. The symbols involved in this paper are summarized in Table 1.

3.1 Variational graph auto-encoder

Based on VAE [24], Kipf *et.al.* introduced a latent variable model, named variational graph auto-encoder (VAGE) [26], which was an unsupervised method for graph data. VGAE uses GCN [25] as an encoder that maps node features into latent space along with adjacency matrix A:

$$q(\mathrm{Z}|\mathrm{A},\mathrm{X}) = \prod_{i=1}^{m} q(z_i|\mathrm{A},\mathrm{X}) \tag{1}$$

where $q(z_i|\mathrm{A},\mathrm{X}) = \mathcal{N}(z_i; \mu_i, diag(\sigma_i^2))$. μ and σ are parametric by two layers GCN with weight matrices W_i as follows:

$$\mu, \log\sigma = \mathrm{GCN}_{\mu,\sigma}(\mathrm{A},\mathrm{X}) = \tilde{\mathrm{A}}\zeta(\tilde{\mathrm{A}}\mathrm{X}W_0)W_1 \tag{2}$$

where W_0 is the shared parameter for GCN. ζ is activation function ReLU [1], and $\tilde{\mathrm{A}} = \mathrm{D}^{-\frac{1}{2}}\mathrm{A}\mathrm{D}^{-\frac{1}{2}}$ is normalized adjacency matrix, D is degree matrix.

Furthermore, VGAE utilizes an inner product decoder as the generative model with latent variable Z:

$$p(\mathrm{A}|\mathrm{Z}) = \prod_{i=1}^{m}\prod_{j=1}^{m} p(\mathrm{A}_{i,j}|z_i, z_j) \tag{3}$$

with $p(\mathrm{A}_{i,j} = 1|z_i, z_j) = \mathrm{sigmoid}(z_i^T, z_j)$. To infer the model, VGAE optimizes the variational lower bound:

$$\mathcal{L}_{vgae} = \mathbb{E}_{q(\mathrm{Z}|\mathrm{A},\mathrm{X})}[\log p(\mathrm{A}|\mathrm{Z})] - \mathbb{KL}[(q(\mathrm{Z}|\mathrm{A},\mathrm{X})||p(\mathrm{Z})] \tag{4}$$

where $\mathbb{KL}$ means Kullback-Leibler divergence between two distributions. $p(\mathrm{Z})$ is Gaussian distribution in VGAE.

3.2 Diffusion model

Inspired by non-equilibrium thermodynamics [23], diffusion model has been proposed for generative tasks, especially in the field of image generation. They defined a **forward diffusion process** using Markov chain that adds random noise to the raw data gradually, then learned a **reverse diffusion process** to recover "*clean*" samples (i.e.: input samples) from noise. Several methods based on the diffusion theory, the most representative are: diffusion probabilistic models [43], denoising diffusion probabilistic models(DDPM) [21], score-based diffusion model [44].

Forward diffusion process $q(\mathrm{x}_{1:T}|\mathrm{x}_0) = \prod_{t=1}^{T} q(\mathrm{x}_t|\mathrm{x}_{t-1})$: We sample from the data distribution $\mathrm{x}_0 = q(\mathrm{x})$, then the forward diffusion process is defined by adding a small amount of Gaussian noise to the sample in T steps, which can obtain a sequence of noisy data $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_T$. Formulate the above process as follows:

$$q(\mathrm{x}_t|\mathrm{x}_{t-1}) = \mathcal{N}(\mathrm{x}_t; \sqrt{1-\beta_t}\mathrm{x}_{t-1}, \beta_t, \mathrm{I}) \tag{5}$$

where β_t controls the amount of noise added in each step t. As t gradually increases, x_0 gradually loses its discriminative features, when reaches T, where x_T can be viewed as following a standard Gaussian distribution $q(\mathrm{x}_\mathrm{T}) \approx \mathcal{N}(0, \mathrm{I})$.

Reverse diffusion process: The reverse diffusion process utilizes a denoising model to progressively recover "*clean*" samples from noisy samples:

$$p_\theta(\mathrm{x}_{0:\mathrm{T}}) = p(\mathrm{x}_\mathrm{T})\prod_{t=1}^{T}\mathcal{N}(\mathrm{x}_{t-1}; \mu_\theta(\mathrm{x}_t, t), \hat{\beta}\mathrm{I}) \tag{6}$$

where μ_{theate} can be parametric by neural networks such as U-Net [41], and Transformer [49].

In the diffusion model, forward diffusion process can be considered as fixed posterior and, therefore does not have any training parameter. For the reverse diffusion process, similar to VAE, diffusion model maximizes the variational lower bound(VLB) of the data likelihood:

$$\mathcal{L}_{vlb} = \mathbb{E}_{q(x_{1:T}|x_0)}[\log \frac{q(\mathrm{x}_T|\mathrm{x}_0)}{p_\theta(\mathrm{x}_T)} + \sum_{t=2}^{T} \log \frac{q(\mathrm{x}_{t-1}|\mathrm{x}_0, \mathrm{x}_t)}{p_\theta(\mathrm{x}_{t-1}|\mathrm{x}_t)} - \log p_\theta(\mathrm{x}_0|\mathrm{x}_1)] \tag{7}$$

However, Nichol et al. [35] argued that direct optimization of $\mathcal{L}_{vlb}$ leads to instability of training. Hence, Song and Ho et al. introduce a simple objective for this optimize problem [21, 44]:

$$\mathcal{L}_{dm} = \mathbb{E}_{\mathrm{x}_0, \epsilon \sim \mathcal{N}(0,\mathrm{I}), t} \left[\omega(t) ||\epsilon - \epsilon_\theta(\mathrm{x}_t, t)||^2 \right] \tag{8}$$

where $\mathrm{x}_t = \alpha_t \mathrm{x}_0 + \sigma_t \epsilon$, $\alpha_t = \sqrt{\prod_{k=1}^{t}(1-\beta_k)}$ and $\sigma_t = \sqrt{1-\alpha_t^2}$. ϵ_θ is parameterized by means: $\mu_\theta(\mathrm{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}(\mathrm{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t^2}}\epsilon_\theta(\mathrm{x}_t, t))$, $\omega(t) = \frac{\beta_t^2}{2\hat{\beta}_t^2(1-\beta_t)(1-\alpha_t^2)}$. For simplicity, $\omega(t)$ setting 1 in the practice.

4 Proposed method

In this section, we introduce our method, **V**ariational **G**raph **N**eural **N**etwork with **D**iffusion **P**rior(VGNNDP). Our approach integrates VGAE with diffusion model, which can learn the powerful prior distribution. Therefore, instead of using diffusion model in raw input data directly, VGNNDP first maps the raw data to the latent space and then uses the diffusion model in the latent space. Hence, compared with VGAE, which assumes the prior distribution is Normal distribution, VGNNDP can learn more expressive prior.

We rewrite (4) as follows:

$$\begin{aligned}\mathcal{L}_{vgae} &= \mathbb{E}_q[\log p_\varphi(\mathrm{A}|\mathrm{Z}) + \mathbb{KL}[q_\phi(\mathrm{Z}|\mathrm{A},\mathrm{X})||p_\theta(\mathrm{Z})] \\ &= \underbrace{\mathbb{E}_q[\log p_\varphi(\mathrm{A}|\mathrm{Z})}_{\text{Reconstruction term}} + \underbrace{\mathbb{E}_q[\log q_\phi(\mathrm{Z}|\mathrm{A},\mathrm{X})]}_{\text{Entropy } \mathbb{H}[q_\phi(\mathrm{Z}|\mathrm{A},\mathrm{X})]} \\ &+ \underbrace{\mathbb{E}_q[\log p_\theta(\mathrm{Z})]}_{\text{Cross Entropy}}\end{aligned} \tag{9}$$

where q is $q_\phi(\mathrm{Z}|\mathrm{A},\mathrm{X})$

In (9), similar to VGAE, the role of distribution $q_\phi(\mathrm{Z}|\mathrm{A},\mathrm{X})$ is approximate the true posterior distribution $p(\mathrm{Z}|\mathrm{A},\mathrm{X})$. The first term is reconstruction loss, which is easy to compute using reparameterization trick [24], use a simple decoder, inner production, can obtain final reconstruction graph. The second term is entropy and it is a constant. Hence, the challenging in the (9) is how to compute $\mathbb{E}_{q_\phi(\mathrm{Z}|\mathrm{A},\mathrm{X})}[\log p_\theta(\mathrm{Z})]$.

Diffusion model is a latent variable model, hence, the forward process can be viewed as a fixed posterior:

$$q(\mathrm{z}_t|\mathrm{z}_{t-1}) = \mathcal{N}(\mathrm{z}_t; \sqrt{1-\beta_t}\mathrm{z}_{t-1}, \beta_t \mathrm{I}) \tag{10}$$

After the forward diffusion process is complete, if we can reverse the forward process, i.e., sampling from $q(\mathrm{z}_{t-1}|\mathrm{z}_t)$, we can then reconstruct the raw samples from the Gaussian distribution progressively. However, sampling from $q(\mathrm{z}_{t-1}|\mathrm{z}_t)$ is difficult because of iterating through the entire dataset. Essentially, the reverse process of the diffusion model is to learn a parameterized denoising function whose purpose is to remove noise from the noisy data Z_T to produce clean samples Z_0. Therefore, we can utilize a learnable model p_θ to approximate conditional distribution and we can obtain reverse diffusion process:

$$p_\theta(\mathrm{z}_{t-1}|\mathrm{z}_t) = \mathcal{N}\left(\mathrm{z}_{t-1}; \mu_\theta(\mathrm{z}_t, t), \left(\sqrt{\frac{\sigma_{t-1}}{\sigma_t}}\beta_t\right)^2 \mathrm{I}\right) \tag{11}$$

Hence, in order to learning model $p_\theta(\mathrm{z}_{t-1}|\mathrm{z}_t)$, we need to parameterize $\mu_\theta(\mathrm{z}_t, t)$, the form of $\mu_\theta(\mathrm{z}_t, t)$ as follows:

$$\mu_\theta(\mathrm{z}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathrm{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t^2}}\epsilon_\theta(\mathrm{z}_t, t)\right) \tag{12}$$

The reverse diffusion process $p_\theta(\mathrm{z}_{0:T})$ can be trained with variational lower bound(VLB):

$$\begin{aligned}\mathcal{L}_{vlb} &= \mathbb{E}_{q(\mathrm{z}_{0:T})}\Big[\log \frac{q(\mathrm{z}_{1:T}|\mathrm{z}_0)}{p_\theta(\mathrm{z}_{0:T})}\Big] \\ &= \mathbb{E}_{q(\mathrm{z}_{1:T}|\mathrm{z}_0)}\Big[\log \frac{q(\mathrm{z}_T|\mathrm{z}_0)}{p_\theta(\mathrm{z}_t)} + \sum_{t=2}^{T} \log \frac{q(\mathrm{z}_{t-1}|\mathrm{z}_0, \mathrm{z}_t)}{p_\theta(\mathrm{z}_{t-1}|\mathrm{z}_t)} \\ &- \log p_\theta(\mathrm{z}_0|\mathrm{z}_1)\Big] \\ &= \mathbb{E}_{q(\mathrm{z}_{0:T})}\Big[\mathbb{KL}(q(\mathrm{z}_T|\mathrm{z}_0)||p_\theta(\mathrm{z}_T)) \\ &+ \sum\nolimits_{t=2}^{T} \mathbb{KL}(q(\mathrm{z}_{t-1}|\mathrm{z}_t, \mathrm{z}_0)||p_\theta(\mathrm{z}_{t-1}|\mathrm{z}_t)) \\ &- \log p_\theta(\mathrm{z}_0|\mathrm{z}_1)\Big]\end{aligned} \tag{13}$$

However, it is difficult to optimize $\mathcal{L}_{vlb}$ directly and can suffer from significant training instability [35]. Therefore, based on [21], we optimize using a simple form:

$$\mathcal{L}_{LDM} = \mathbb{E}_{t, \epsilon, q_\phi(\mathrm{z}_0, \mathrm{z}_T|\mathrm{A},\mathrm{X})}\left[\frac{\omega(t)}{2}||\epsilon - \epsilon_\theta(\mathrm{z}_t, t)||_2^2\right] \tag{14}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, $t \sim [1, T]$. For this simple equation, it means we can remove the predicted noise by subtracting it from the Gaussian noise and hence can be expected to be able to recover the original input in the diffusion inverse process.

In order to parameterize ϵ_θ, and the data in our task is graph data, we proposed a novel model. Specifically, we utilize multi-layer GNN to obtain the latent representation of input alone with node feature X, then, MLP is used to predict noise:

$$\left.\begin{aligned} \mathrm{H}_i^0 &= ([\mathrm{z}_t, \mathrm{X}]) \\ m_i^{(l)} &= AGG^{(l)}\{(H_j^{l-1})_{j \in \mathcal{N}(i)}\} \\ \mathrm{H}^L &= COMBINE(\mathrm{H}_i^{l-1}, m_i^{(l)}) + pe(t) \\ \epsilon_\theta(\mathrm{z}_t, t) &= \mathrm{MLP}(\mathrm{H}^L) \end{aligned}\right\} \tag{15}$$

where AGG is an aggregate function in GNN. $pe(t)$ is positional embedding, This is important to remember where the signals appeared, allowing the model to be precisely reconstructed, we use sinusoidal positional embedding [49] in this paper.

Recall the cross entropy term $\mathbb{E}_{q_\phi(\mathrm{Z}|\mathrm{A},\mathrm{X})}[\log p_\theta(\mathrm{Z})]$ in (9), we can minimize it as learning objective:

$$\begin{aligned}\mathcal{L}_{CE} &= -\mathbb{E}_{q_\phi(\mathrm{z}_0|\mathrm{A},\mathrm{X})}[\log p_\theta(\mathrm{z}_0)]\\ &\le -\mathbb{E}_{q(\mathrm{z}_{0:T})}\log\frac{p_\theta(\mathrm{z}_{0:T})}{q(\mathrm{z}_{1:T}|\mathrm{z}_0)}\\ &= \mathbb{E}_{q(\mathrm{z}_{0:T})}\log\frac{q(\mathrm{z}_{1:T}|\mathrm{z}_0)}{p_\theta(\mathrm{z}_{0:T})} = \mathcal{L}_{vlb}\end{aligned} \tag{16}$$

The last term in (13), due to the reason that q does not have any learnable parameters and z_T is Gaussian noise, the $\mathbb{KL}(q(\mathrm{z}_T|\mathrm{z}_0)||p_\theta(\mathrm{z}_T)) = L_T$ term is a constant and can be ignored during the training. Hence, we have:

$$\begin{aligned}\mathcal{L}_{CE} \le \mathcal{L}_{vlb} = &\underbrace{\sum_{t=2}^{T}\mathbb{KL}(q(\mathrm{z}_{t-1}|\mathrm{z}_t,\mathrm{z}_0)||p_\theta(\mathrm{z}_{t-1}|\mathrm{z}_t))}_{\mathcal{L}_{LDM}^{(t-1)}}\\ &- \log p_\theta(\mathrm{z}_0|\mathrm{z}_1)\end{aligned} \tag{17}$$

For $\log p_\theta(\mathrm{z}_0|\mathrm{z}_1)$, we can proceed with the Gaussian form of the mean directly:

$$\mu_\theta(\mathrm{z}_1, 1) = \frac{\mathrm{z}_1 - \sigma_1\epsilon_\theta(\mathrm{z}_1, 1)}{\alpha_1} \tag{18}$$

with

$$\mathrm{z}_0 = \frac{\mathrm{z}_1 - \sigma_0\epsilon}{\alpha_1} \tag{19}$$

Finally, we can obtain the form of $\log p_\theta(\mathrm{z}_0|\mathrm{z}_1)$ as follows:

$$\log p_\theta(\mathrm{z}_0|\mathrm{z}_1) = -\log \mathrm{z}^{-1} + ||\epsilon - \epsilon_\theta(\mathrm{z}_1, 1)||_2^2 = -\mathcal{L}_{LDM}^{(0)} \tag{20}$$

Hence, we have $\mathcal{L}_{CE} \le \mathcal{L}_{LDM}$. See Appendix A for detailed derivation. Therefore, (9) can be written as:

$$\begin{aligned}\mathcal{L}_{vgae} \le &\underbrace{\mathbb{E}_{q_\phi(\mathrm{Z}|\mathrm{A},\mathrm{X})}[\log p_\varphi(\mathrm{A}|\mathrm{Z})]}_{\mathcal{L}_{recon}}\\ &+ \mathbb{E}_{t\sim[1,T],\epsilon\sim\mathcal{N}(0,\mathbf{I}),q_\phi(\mathrm{z}_0,\mathrm{z}_T|\mathrm{A},\mathrm{X})}\left[\frac{\omega(t)}{2}||\epsilon - \epsilon_\theta(\mathrm{z}_t, t)||_2^2\right]\end{aligned} \tag{21}$$

Finally, we can write the loss function as follows:

$$\mathcal{L} = \mathcal{L}_{recon} + \lambda * \mathcal{L}_{LDM} + \mathcal{L}_{reg} \tag{22}$$

where, $\mathcal{L}_{recon}$ is reconstruction loss, equals first term in (9), which play a role for measures the difference between the reconstructed adjacency matrix and the original adjacency matrix. $\mathcal{L}_{LDM}$ is latent diffusion loss, which used for predict noise model. λ is hyper-parameter, $\mathcal{L}_{reg}$ is KL penalty, measuring the difference between the latent representation distribution obtained by encoding and the prior distribution learned through diffusion modeling. Our algorithm is shown in Algorithm 1.

Algorithm 1 Our proposed VGNNDP.

```
Require: adjacency matrix A and node feature X, Encoder $E_\phi$, denoising model $\epsilon_\theta$
1: First stage: VGAE Training
2: while $i$ in range(epochs) do
3:     $\mu \Leftarrow E_\phi(\mathrm{X}, \mathrm{A})$    {Encoding stage}
4:     $\epsilon \sim \mathcal{N}(0, \mathrm{I})$
5:     $\mathrm{Z} \Leftarrow \epsilon \odot \sigma_0 + \mu$    {Reparameterization}
6:     $p(\mathrm{A} = 1) \Leftarrow \mathrm{sigmoid}(\mathrm{ZZ}^T)$
7:     optimize $\mathcal{L}_{vgae}$ with (4)
8:     return $\phi$
9: end while
10: (see Section 3.1)
11: Second Stage: Diffusion model training with latent variable
12: Fix encoder parameters $\phi$
13: while $i$ in range(epochs) do
14:     $\mathrm{Z} = q_\phi(\mathrm{Z}|\mathrm{A}, \mathrm{X})$ (As in First stage)
15:     $t \sim \mathrm{Uniform}(0, T)$, $\epsilon \sim \mathcal{N}(0, \mathrm{I})$
16:     $\mathrm{z}_t \Leftarrow \alpha_t \mathrm{z}_0 + \sigma_t \epsilon$
17:     $\epsilon_\theta(\mathrm{z}_t, t) \Leftarrow \mathrm{MLP}(\mathrm{H}^L)$    {equation (15)}
18:     $\mathcal{L}_{LDM} = \frac{\omega(t)}{2}||\epsilon - \epsilon_\theta(\mathrm{z}_t, t)||_2^2$
19:     return $\theta$
20: end while
21: (see Section 4)
22: RETURN $\phi, \theta$
```

5 Experiments

In this section, we demonstrate the effectiveness of our proposed method VGNNDP. We conduct experiments on the link prediction task and compare it with other methods on different datasets.

5.1 Datasets

We conduct experiments using two groups of different datasets to validate the performance of our approach. One group is datasets with node features and another group is datasets without node features.

Without node features: USAir, Airlines network, including 332 nodes and 2126 edges [52]; NS, a collaborative network of network science researchers, including 1589 nodes and 2742 edges [33]; Yeast, a PPI network in yeast, which has 2375 nodes and 11693 edges [53]; Power, an energy network with 4941 nodes and 6594 edges [54], and Router, a router Internet with 5022 nodes and 6258 edges [45].

With node features:

1. Citation network datasets [58]: Cora, CiteSeer, and PubMed. These three datasets are widely used for link prediction, in which nodes represent papers and edges mean citation relation between papers.

2. WebPage dataset [38]: Cornell, Texas, Wisconsin, where web pages are represented as nodes and the hyperlinks between web pages are represented as edges.
3. Amazon datasets [42]: Computers, Photo, where nodes denote products and edges denote that two products are usually purchased together. These two datasets are large and are more challenging to handle.

The detailed information of these datasets is summarized in Table 2.

5.2 Baseline and metric

We compare our approach with the following categories of methods to verify the advantages of our approach:

(1) VGAE and VGAE-based approaches: GAE [26], VGAE [26], $\mathcal{S}$-VAE [7], AEVGE [37], Graphite [15], GNAE [2], VGNAE [2], DGAE [12], VDGAE [12], NESS [48].
(2) The classical unsupervised graph learning approaches: DeepWalk(DW) [39], Node2Vec(N2V) [14], Spectral clustering(SC) [47].
(3) Graph-level methods: GIC [32] and DGI [51].

Area under the ROC curve (AUC) and average precision (AP) are used to evaluate the effectiveness of the methods.

5.3 Experiments setup

We validate our method on the link prediction task. As in previous methods, the dataset is partitioned with a partitioning rule of 5% for the validation set, 10% for the test set, and the remaining for the training set. We randomly select 10 seeds to run each experiment 10 times and use the average as the final result. Our approach is divided into two stages: a VGAE pre-train stage and a latent variable diffusion stage. In the first stage, we pre-train the VGAE. The training parameters are: learning rate is 0.01, hidden feature dimension is 64, GCN is used for the encoder, the simple inner product is used for the decoder, and we run 500 epochs, using Adam optimization function. In the second stage, the parameters are set as follows: learning rate is 0.002, to avoid overfitting, weight decay is used with a value of 0.0005, and the number of diffusion steps, T, is 1000. In the forward diffusion process, the noise schedule uses polynomial noise [56], the parametric α is decays from T to 0 linearly. For denoising model (Equation (15)), GCN uses two layers with hidden dimension 64, the hidden dimension of MLP is 128. All experiments were performed on the Ubuntu operating system, utilizing the A100 GPU, and implemented using Pytorch-geometric, a Pytorch-based graph learning framework.

5.4 Results and analysis

First, we conduct experiments on datasets with node features. Tables 3 and 4 show the performance AUC and AP of 3 citation datasets. We can find out in these two tables, that our method obtained best results. Compared to the results of the best-unsupervised methods, on the datasets Cora and Citeseer, our method outperforms these methods by more than 8% in AUC and AP, and on the dataset PubMed, our method improves by nearly 6% in AUC and AP. For graph-level approaches, our method also obtained the best results, with an improvement of more than 6% over DGI and GIC in AUC and

Table 2 The detail information of datasets

Dataset	Nodes	Edges	Node Features	Labels
Cora	2708	5429	1433	7
CiteSeer	3327	4732	3703	6
PubMed	19717	44338	500	3
Cornell	183	298	1703	5
Texas	183	325	1703	5
Wisconsin	251	515	1703	5
Computers	13752	491722	767	10
Photo	7650	238162	745	8
NS	1589	2742	–	–
Power	4941	6594	–	–
Router	5022	6258	–	–
USAir	332	2126	–	–
Yeast	2375	11693	–	–

Table 3 AUC performance on the 3 citation datasets, with best results bolded

Methods	Cora	Citeseer	PubMed
SC	84.60	80.50	84.20
DW	83.10	80.50	84.40
N2V	85.60	89.40	91.90
GIC	93.50	97.00	93.70
DGI	89.80	95.50	91.20
GAE	91.05	89.99	92.33
VGAE	91.37	90.05	84.64
$\mathcal{S}$-VAE	94.10	94.70	96.00
ARVGE	91.27	87.78	96.47
Graphite	94.70	97.30	97.40
GNAE	94.08	96.90	95.41
VGNAE	89.19	95.45	89.66
DGAE	95.80	97.23	97.77
VDGAE	95.90	97.82	97.00
NESS8	97.83	99.11	96.28
Ours	**98.81**	**99.27**	**98.53**

Table 4 AP performance on the 3 citation datasets, with best results bolded

Methods	Cora	Citeseer	PubMed
SC	88.50	85.00	87.80
DW	85.00	83.60	84.10
N2V	87.50	91.30	92.30
GIC	93.30	96.80	93.50
DGI	89.70	95.70	92.20
GAE	92.62	91.00	92.61
VGAE	92.26	90.05	86.02
$\mathcal{S}$-VAE	81.90	95.20	96.00
ARVGE	91.39	88.09	96.52
Graphite	94.90	97.40	97.40
GNAE	94.40	97.19	94.86
VGNAE	89.58	95.70	89.36
DGAE	96.07	97.46	97.82
VDGAE	96.17	98.03	97.82
NESS8	98.26	99.21	96.05
Ours	**98.80**	**99.25**	**98.32**

AP. For Cora and Citeseer, our method achieves about 7.5%, 9.2% on AUC and 6.6%, 9.2% on AP margins of improvements compared with VGAE. On the dataset PubMed, our method is nearly 14% higher on AUC and 12% higher on AP than VGAE. Compared with other VGAE-based methods, our methods also outperform by a wide margin. Particularly, compared with VGNAE, our method obtained about 10% on both AUC and AP for Cora and Citeseer. Furthermore, our method also achieves 2% improvements over the best VGAE-based methods on Cora and PubMed, but small margins were achieved in the dataset Citeseer. Our method performs effectively due to the powerful expressive properties of the prior distribution learned by the diffusion model, which fits the data better and eliminates incompatibility with the posterior distribution compared to a simple Gaussian distribution.

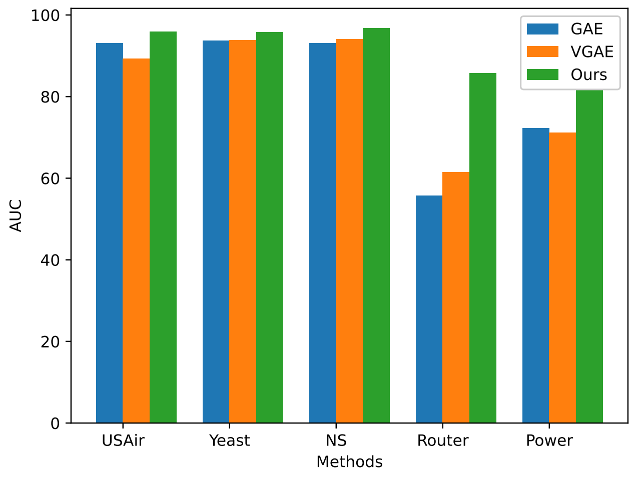
We also demonstrate the effectiveness of our method on the datasets Amazon and webPage. The AUC result is shown in Table 5. As can be observed from the table, our method

Table 5 AUC performance on Amazon and WebPage datasets, with best results bolded

Methods	Cornell	Texas	Wisconsin	Computers	Photo
GAE	73.60	75.34	68.87	86.26	87.32
VGAE	78.30	76.68	66.94	87.14	87.03
ARVGE	78.88	76.46	71.10	82.97	81.07
GNAE	72.93	75.14	78.17	92.17	94.64
VGNAE	73.28	78.93	70.26	80.68	79.48
DGAE	68.05	68.27	75.73	**94.12**	95.49
VDGAE	76.08	81.30	85.03	93.29	94.68
Ours	**98.44**	**94.92**	**92.44**	91.41	**96.11**

achieves the best results for all datasets except Computers. For WebPage datasets, our method achieves large margins of improvement compared with other methods. Particularly, for Cornell, our method obtains nearly 20% margins of improvements than the second best result. Similar results are seen for Texas and Wisconsin, with nearly 16% and 7% improvement, respectively. In the dataset Amazon, Photo obtains best results compared with other methods and has a small margin of improvement. However, for Computers, our methods performed poorly and failed to produce the best results, but the margins were small compared to the best results, approximately 3%. The reason for this may be that the Computers dataset is very large, with 2X the number of nodes and edges of Photo and even 1500X the number of edges of the WebPage dataset, which demonstrates that our model has no advantage in dealing with very large datasets, and this is the focus of our future work.

Subsequently, we conducted comparative experiments on datasets devoid of node features against Graph Autoencoder



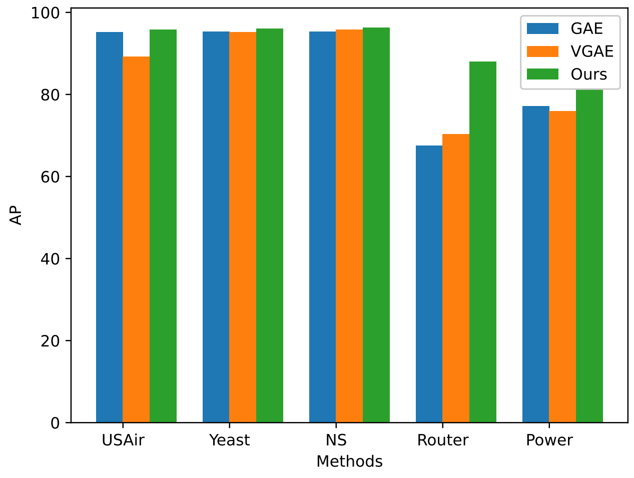


Fig. 3 The performance of 5 datasets without node features. (top) AUC, (bottom) AP

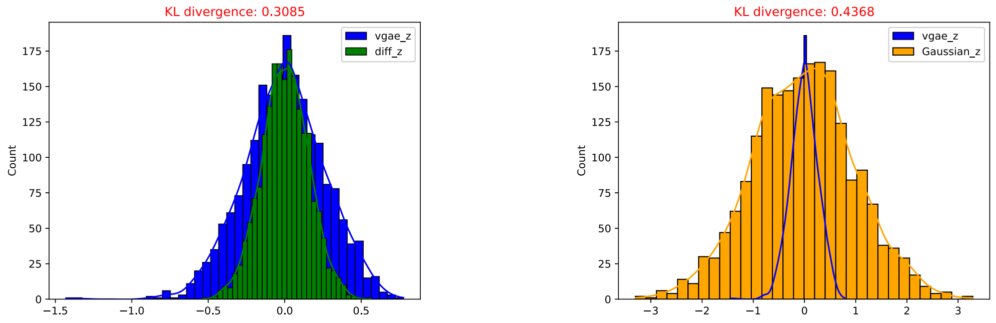


Fig. 4 The distribution compares between Gaussian prior, VGAE-based posterior, and diffusion prior on the Cora dataset. (left) diffusion prior and VGAE posterior, (right) Gaussian prior and VGAE posterior

(GAE) and Variational Graph Autoencoder (VGAE). The comparative results are graphically represented in the bar chart presented in Fig. 3. A clear observation from the chart is that our methodology outperformed the existing approaches. Specifically, for the Router and Power datasets, our method achieved a notable enhancement in performance metrics, with nearly a 23% increase in Area Under the Curve (AUC) and around an 18% rise in Average Precision (AP). Although our approach demonstrated a less pronounced improvement on the other datasets when juxtaposed with VGAE and GAE, it still managed to secure a minimum of a 2% improvement. The substantial gains observed in the Router and Power datasets can be attributed to their larger size, which allowed our model to effectively capture and learn the intricate graph structure information in the absence of node features. In contrast, the modest improvements in the remaining datasets are likely due to their smaller scale, which restricted the model's capacity to glean as much information about the underlying graph structure, thereby capping the extent of performance enhancement.

Our methodology leverages a diffusion model to improve the prior assumption in VGAE, expecting the distribution of the latent variables (posterior distribution) as close as possible to the prior distribution, i.e., the KL term "$\mathbb{KL}[q(\mathrm{Z}|\mathrm{A},\mathrm{X})||p(\mathrm{Z})]$" in (4) is as small as possible, ideally, we strive for the condition where $\mathbb{KL}(q[\cdot]||p[\cdot]=0$. Unlike VGAE, which assumes the prior as Gaussian distribution, our approach is capable of learning a more nuanced and expressive prior through the diffusion process. We compare the prior distribution in VGAE and the prior distribution learned by our method with the posterior distribution in VGAE, respectively, and calculate the KL divergence values between Prior and posterior in 3 citation datasets Cora, Citeseer, and PubMed. The results are shown in Figs. 4, 5, and 6.

From these 3 figures, we can observe that the prior distribution learned by our method aligns more closely with the VGAE posterior distribution, i.e., smaller values of KL divergence. Specifically, for the Pubmed dataset, our method achieves a notably lower KL divergence value. This suggests that the diffusion-based prior is a better match for the posterior distribution, approaching the ideal condition where the two distributions are virtually indistinguishable. Such a result is highly desirable for minimizing the $\mathbb{KL}$ term in the ELBO (4), reflecting an optimal fit between the model's assumptions and the empirical data. The same observation is present in the datasets Cora and Citeseer. The same observation occurs in the datasets cora and Citeseer, indicating that our method learns more flexible prior distributions that can be better matched to the posterior distributions, thus improving the flexibility of variational optimization.

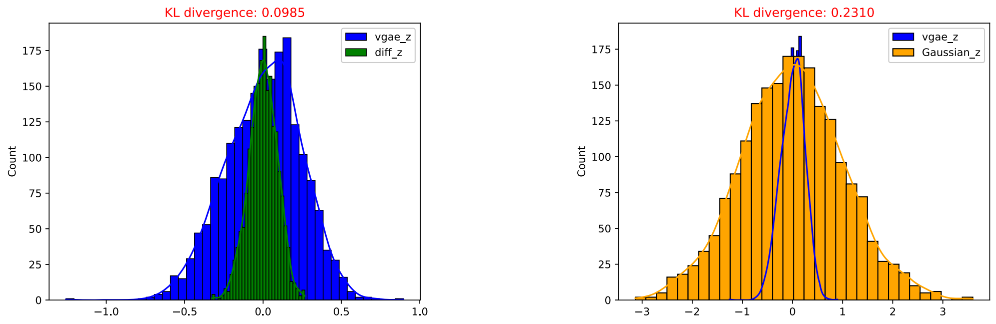


Fig. 5 The distribution compares between Gaussian prior, VGAE-based posterior, and diffusion prior on the Citeseer dataset. (left) diffusion prior and VGAE posterior, (right) Gaussian prior and VGAE posterior

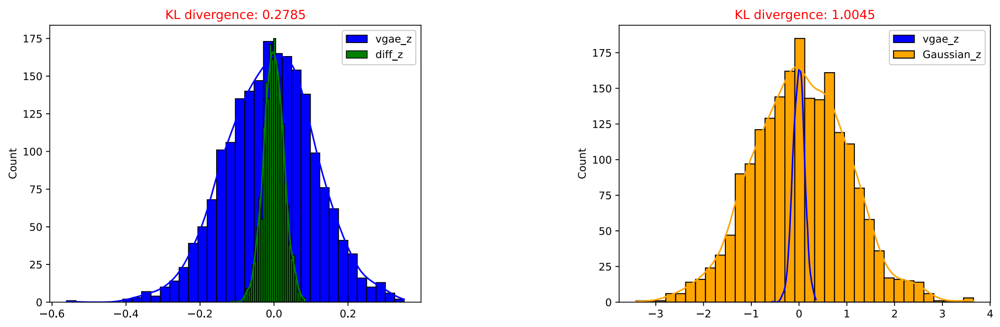


Fig. 6 The distribution compares between Gaussian prior, VGAE-based posterior, and diffusion prior on the PubMed dataset. (left) diffusion prior and VGAE posterior, (right) Gaussian prior and VGAE posterior

5.5 Hyperparametric analysis

In our method, the number of pre-training steps in the first stage of Algorithm 1 has an impact on the results. We demonstrate this experimentally. We select pre-training epochs ranging from [50,500]. The results are displayed in Fig. 7. As we can find in the figure, with the increase of the pre-training epoch, the AUC and AP of all three datasets are growing, and reach the maximum value when epoch=500, since the VGAE has converged at epoch=500, we choose epoch=500 in order to balance performance. The main reason why the epoch of pre-training affects the results is that we utilize diffusion models on the latent variables, hence the quality of the latent variables is crucial. Therefore, different epochs influence the quality of the latent variables and consequently impact the final results.

In the second stage of Algorithm 1, the diffusion step also impacts the results. We conduct experiments that choose diffusion step $T \in [100, 1000]$, the results are shown in Fig. 8. We can find in the figures, that the increase of T, AUC and AP are shocking. For Cora dataset, when $T = 600$, AUC and AP decreased sharply compared to $T = 500$. For Citeseer and Pubmed, as the increase of T, AUC and AP slightly decrease or increase. When $T = 1000$, AUC and AP of these 3 datasets reached the maximum. Considering that the growth is not significant when $T > 1000$, we choose $T = 1000$ in order to reduce the computational complexity, which is the strategy used in most of the papers.

5.6 Computational complexities analysis

In this paper, we first map the graph data to the latent space utilizing VGAE, and then use the diffusion model to model the prior distribution in the low-dimensional latent space. This strategy offers a significant advantage over applying diffusion models directly to the original data: the dimensionality of the latent space is substantially lower than that of the original data, thereby diminishing the computational complexity of the model. To substantiate the efficiency of our approach, we conducted experiments across various datasets, running for 500 epochs each, and recorded the final execution times. The outcomes are detailed in Table 6. The data clearly demonstrate that our method substantially outperforms the direct application of diffusion models on the original datasets in terms of runtime. Our experiments, which encompassed datasets of varying sizes-listed in ascending order of size in the table-indicate that our method is effective in reducing the time complexity for both small and large-scale graph data. For instance, on the Cora dataset, our method conserved approximately 3.7 minutes of runtime compared to the direct use of the diffusion model. Similarly, on the considerably larger Computers dataset, our method achieved a runtime

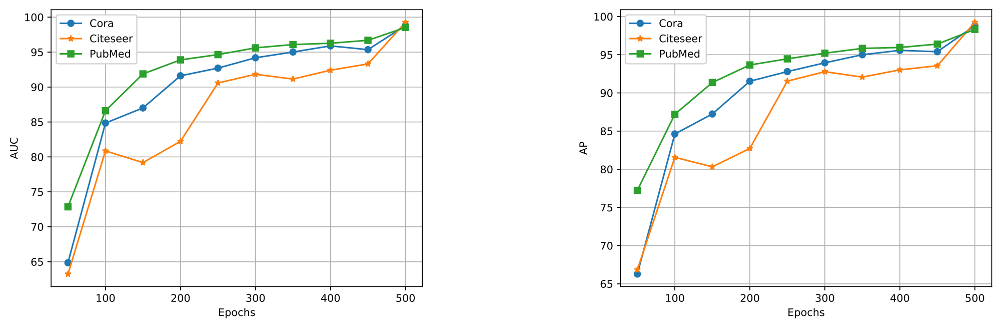


Fig. 7 Performance of the impact of the epoch of the pre-training stage on the results of the 3 citation datasets. (left) AUC, (right) AP

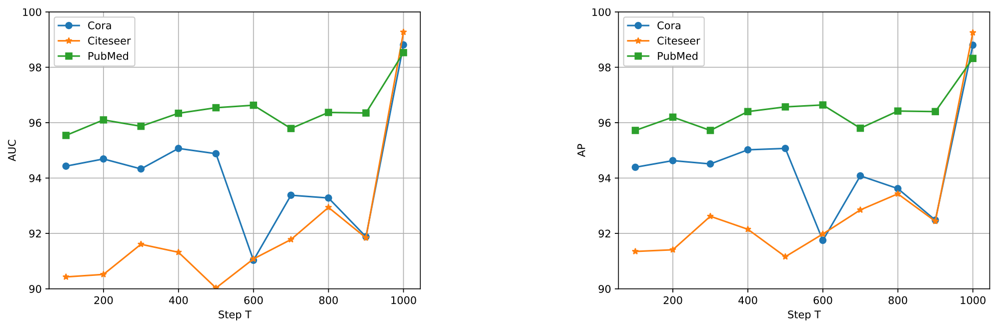


Fig. 8 Performance of the impact of diffusion step T on the results of the 3 citation datasets. (left) AUC, (right) AP

reduction of about 4 minutes. These findings underscore that our method significantly alleviates the time complexity demands, particularly when dealing with very large-scale datasets, thereby enhancing computational efficiency.

6 Conclusions

In this paper, we proposed a novel method VGNNDP, which integrates VGAE and diffusion model. The conventional VGAE, which assumes the prior distribution to be a Normal distribution, restricts its expressive power, when incompatible with the posterior distribution, it is not friendly to the downstream tasks. To solve this problem, we utilize diffusion model learning more powerful prior, which breaks out of the limitations of the assumption of a simple Gaussian distribution. In particular, in order to be able to fit the denoising model to the graph data, we propose a novel denoising model that utilizes a stack of GNNs and MLPs to better capture the graph structure information. Specifically, since we employ the diffusion model on the latent variables rather than on the raw data, the time complexity is significantly reduced compared to the conventional diffusion model. Finally. we conduct experiments on different datasets, and the results show that our method obtains the best results. However, we find that our approach performs poorly on very large datasets, such as Computers of Amazon, which typically have a thousand times more edges or nodes than other datasets. This is caused by the complexity of the diffusion model, and how to utilize the diffusion model on very large datasets is the focus of our future work.

Table 6 The runtime of different datasets (by minutes)

Dataset	Original data	Latent space data
Texas	0.37	0.32
Cora	8.69	5
Pubmed	15	14.35
Computers	35	31

Appendix A: Proof $\mathcal{L}_{CE} \leq \mathcal{L}_{LDM}$

When we derive the loss function, combined with (16) and (17), we have inequality $\mathcal{L}_{CE} \leq \mathcal{L}_{LDM}$. Here we give the detailed derivation.

Proof First, we give the detailed process in (16) as follows:

$$\begin{aligned}
\mathcal{L}_{CE} &= -\mathbb{E}_{q_\phi(z_0|A,X)}\left(p_\theta(z_0)\right) \\
&= -\mathbb{E}_{q_\phi(z_0|A,X)} \log\left(\int p_\theta(z_{0:T})\, dz_{1:T}\right) \\
&= -\mathbb{E}_{q_\phi(z_0|A,X)} \log\left(\int q(z_{1:T} \mid z_0)\frac{p_\theta(z_{0:T})}{q(z_{1:T} \mid z_0)} dz_{1:T}\right) \\
&= -\mathbb{E}_{q_\phi(z_0|A,X)} \log\left(\mathbb{E}_{q(z_{1:T}|z_0)}\frac{p_\theta(z_{0:T})}{q(z_{1:T} \mid z_0)}\right) \\
&\leq -\mathbb{E}_{q(z_{0:T})} \log\frac{p_\theta(z_{0:T})}{q(z_{1:T} \mid z_0)} \\
&= \mathbb{E}_{q(z_{0:T})} \log\frac{q(z_{1:T} \mid z_0)}{p_\theta(z_{0:T})} = \mathcal{L}_{VLB}
\end{aligned} \tag{A.1}$$

This is similar to VAE, which optimizes negative log-likelihood with variational lower bound (VLB) and can lead to same result. Then, we decompose $\mathcal{L}_{VLB}$ so that it takes the form of an analytic solution:

$$\begin{aligned}
\mathcal{L}_{vlb} &= \mathbb{E}_{q(z_{0:T})}\left[\log\frac{q(z_{1:T} \mid z_0)}{p_\theta(z_{0:T})}\right] \\
&= \mathbb{E}_q\left[\log\frac{\Pi_{t=1}^T q(z_t \mid z_{t-1})}{p_\theta(z_T)\,\Pi_{t=1}^T p_\theta(z_{t-1} \mid z_t)}\right] \\
&= \mathbb{E}_q\left[-p_\theta(z_T) + \sum_{t=1}^T \log\frac{q(z_t \mid z_{t-1})}{p_\theta(z_{t-1} \mid z_t)}\right] \\
&= \mathbb{E}_q\left[p_\theta(z_T) + \sum_{t=2}^T \log\frac{q(z_t \mid z_{t-1})}{p_\theta(z_{t-1} \mid z_t)} + \log\frac{q(z_1 \mid z_0)}{p_\theta(z_0 \mid z_1)}\right]
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
&= \mathbb{E}_q\left[-p_\theta(z_T)\right. \\
&+ \sum_{t=2}^T \log\left(\frac{q(z_{t-1} \mid z_t, z_0)}{p_\theta(z_{t-1} \mid z_t)} \times \frac{q(z_t \mid z_0)}{p_\theta(z_{t-1} \mid z_0)}\right) \\
&\left.+\log\frac{q(z_1 \mid z_0)}{p_\theta(z_0 \mid z_1)}\right]
\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}_q\left[-p_\theta(z_T) + \sum_{t=2}^{T} \log \frac{q(z_{t-1} \mid z_t, z_0)}{p_\theta(z_{t-1} \mid z_t)}\right.\\
&\left.+ \sum_{t=2}^{T} \frac{q(z_t \mid z_0)}{p_\theta(z_{t-1} \mid z_0)} + \log \frac{q(z_1 \mid z_0)}{p_\theta(z_0 \mid z_1)}\right]\\
&= \mathbb{E}_q\left[-p_\theta(z_T) + \sum_{t=2}^{T} \log \frac{q(z_{t-1} \mid z_t, z_0)}{p_\theta(z_{t-1} \mid z_t)}\right.\\
&+ \log \frac{q(z_T \mid z_0)}{q(z_1 \mid z_0)}\\
&\left.+ \log \frac{q(z_1 \mid z_0)}{p_\theta(z_0 \mid z_1)}\right]\\
&= \mathbb{E}_q\left[\log \frac{q(z_T \mid z_0)}{p_\theta(z_T)} + \sum_{t=2}^{T} \log \frac{q(z_{t-1} \mid z_t, z_0)}{p_\theta(z_{t-1} \mid z_t)}\right.\\
&\left.- \log p_\theta(z_0 \mid z_1)\right]\\
&= \mathbb{E}_q\left[\mathbb{KL}\Big(q(z_T \mid z_0) \,||\, p_\theta(z_T)\Big)\right.\\
&+ \sum_{t=2}^{T} \mathbb{KL}\Big(q(z_{t-1} \mid z_t, z_0) \,||\, p_\theta(z_{t-1} \mid z_t)\Big)\\
&\left.- \log p_\theta(z_0 \mid z_1)\right]
\end{aligned}
$$

In (A.2), the term $\mathbb{E}_q\Big[\mathbb{KL}(q(z_T|z_0)||p_\theta(z_T))\Big]$ is a constant because the distribution q is fixed without any learnable parameters and z_T is Gaussian noise, hence, it can be ignored. Therefore, combination of (A.1) and (A.2), we can obtain:

$$
\begin{aligned}
\mathcal{L}_{CE} &= -\mathbb{E}_{q_\phi(z_0|A,X)}\left(p_\theta(z)\right) \le \mathcal{L}_{VLB}\\
&= \underbrace{\sum_{t=2}^{T} \mathbb{KL}\Big(q(z_{t-1} \mid z_t, z_0) \,||\, p_\theta(z_{t-1} \mid z_t)\Big)}_{\mathcal{L}_{LDM}^{(t-1)}}\\
&- \log p_\theta(z_0 \mid z_1)
\end{aligned} \tag{A.3}
$$

the term $\mathcal{L}_{LDM}^{(t-1)}$ can be written as:

$$
\begin{aligned}
&\mathcal{L}_{LDM}^{(t-1)}\\
&= \mathbb{E}_{z_0,\epsilon\sim\mathcal{N}(0,I)}\left[\frac{\beta_t^2}{2\rho_t^2(1-\beta_t)(1-\alpha_t^2)}||\epsilon - \epsilon_\theta(z_t, t)||_2^2\right]
\end{aligned} \tag{A.4}
$$

For $p_\theta(z_0|z_1)$, we have

$$
\begin{aligned}
\mu_\theta(z_1, 1) &= \frac{z_1 - \sigma_1\epsilon_\theta(z_1, 1)}{\alpha_1}\\
z_0 &= \frac{z_1 - \sigma_0\epsilon}{\alpha_1}
\end{aligned} \tag{A.5}
$$

hence,

$$
\begin{aligned}
\log p_\theta(z_0|z_1) &= -\log z^{-1} + ||\epsilon - \epsilon_\theta(z_1, 1)||_2^2\\
&= -\mathcal{L}_{LDM}^{(0)}
\end{aligned} \tag{A.6}
$$

Finally, we obtain:

$$
-\mathbb{E}_{q(z_0|A,X)}\Big(p_\theta(z)\Big) \le \mathcal{L}_{LDM}^{(t-1)} + \mathcal{L}_{LDM}^{(0)} = \mathcal{L}_{LDM} \tag{A.7}
$$

□

Proof Completed.

Funding This work was supported in part by STI 2030-Major Projects, under Grant 2021ZD0200403, and partly supported by grants from the National Natural Science Foundation of China, Nos. 62333018, 62372255, 62432013, 62402250, 62433001, U22A2039, 62073231, 61932008, and 62372318, and supported by the China Post-doctoral Science Foundation under Grant No. 2023M733400, and supported by the Natural Science Foundation of Ningbo City under Grant No. 2023J199, and supported by Key Research and Development Program of Ningbo City under Grant Nos. 2023Z219, 2023Z226, 2024Z112, and supported by the Key Project of Science and Technology of Guangxi (Grant No. 2021AB20147), Guangxi Natural Science Foundation (Grant Nos. 2021JJA170204 and 2021JJA170199) and Guangxi Science and Technology Base and Talents Special Project (Grant Nos. 2021AC19354 and 2021AC19394), and supported by Basic Research Program Project of Department of Science and Technology of Guizhou Province: ZK2024ZD035. We thank the IDT High Performance Computing Platform for providing computational resources for this project.

Declarations

Conflict of interest The author has no competing interests to declare in relation to the content of this article.

Consent for publication The content of the paper is original and not considered for publication in any other journal/proceedings.

References

1. Agarap AF (2018) Deep learning using rectified linear units (relu). arXiv:1803.08375
2. Ahn SJ, Kim M (2021) Variational graph normalized autoencoders. In: Proceedings of the 30th ACM international conference on information & knowledge management. ACM, pp 2827–2831
3. Bai L, Cui L, Wang Y et al (2024) Haqjsk: Hierarchical-aligned quantum jensen-shannon kernels for graph classification. IEEE Trans Knowl Data Eng

4. Brody S, Alon U, Yahav E (2021) How attentive are graph attention networks? arXiv:2105.14491
5. Chandra R, Bhagat A, Maharana M et al (2021) Bayesian graph convolutional neural networks via tempered mcmc. IEEE Access 9:130353–130365
6. Chen M, Wei Z, Huang Z et al (2020) Simple and deep graph convolutional networks. In: International conference on machine learning. PMLR, pp 1725–1735
7. Davidson TR, Falorsi L, De Cao N et al (2018) Hyperspherical variational auto-encoders. arXiv:1804.00891
8. Duan W, Xuan J, Qiao M et al (2022) Learning from the dark: boosting graph convolutional neural networks with diverse negative samples. In: Proceedings of the AAAI conference on artificial intelligence, pp 6550–6558
9. Duan W, Lu J, Wang YG et al (2024) Layer-diverse negative sampling for graph neural networks. arXiv:2403.11408
10. Fan W, Ma Y, Li Q et al (2019) Graph neural networks for social recommendation. In: The world wide web conference. ACM, pp 417–426
11. Feng K, Rao G, Zhang L et al (2023) An interlayer feature fusion-based heterogeneous graph neural network. Appl Intell 53(21):25626–25639. https://doi.org/10.1007/s10489-023-04840-w
12. Fu J, Zhang X, Li S et al (2023) Variational disentangled graph auto-encoders for link prediction. arXiv:2306.11315
13. Gao X, Dai W, Li C et al (2021) ipool-information-based pooling in hierarchical graph neural networks. IEEE Trans Neural Netw Learn Syst 33(9):5032–5044
14. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 855–864
15. Grover A, Zweig A, Ermon S (2019) Graphite: Iterative generative modeling of graphs. In: International conference on machine learning. PMLR, pp 2434–2444
16. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. Adv Neural Inf Process Syst 30
17. Hasanzadeh A, Hajiramezanali E, Narayanan K et al (2019) Semi-implicit graph variational auto-encoders. Adv Neural Inf Process Syst 32
18. Hasanzadeh A, Hajiramezanali E, Boluki S et al (2020) Bayesian graph neural networks with adaptive connection sampling. In: International conference on machine learning. PMLR, pp 4094–4104
19. He K, Zhang X, Ren S et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, pp 770–778
20. Hershey S, Chaudhuri S, Ellis DP et al (2017) Cnn architectures for large-scale audio classification. In: 2017 ieee international conference on acoustics, speech and signal processing (icassp). IEEE, pp 131–135
21. Ho J, Jain A, Abbeel P (2020) Denoising diffusion probabilistic models. Adv Neural Inf Process Syst 33:6840–6851
22. Huang C, Li M, Cao F et al (2022) Are graph convolutional networks with random weights feasible? IEEE Trans Pattern Anal Mach Intell 45(3):2751–2768
23. Jarzynski C (1997) Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. Phys Rev E 56:5018–5035. https://api.semanticscholar.org/CorpusID:119101580
24. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv:1312.6114
25. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
26. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv:1611.07308
27. Li G, Müller M, Ghanem B et al (2021) Training graph neural networks with 1000 layers. In: International conference on machine learning. PMLR, pp 6437–6449
28. Li M, Zhang L, Cui L et al (2023) Blog: Bootstrapped graph representation learning with local and global regularization for recommendation. Pattern Recognit 144:109874
29. Li M, Micheli A, Wang YG et al (2024) Guest editorial: deep neural networks for graphs: theory, models, algorithms, and applications. IEEE Trans Neural Netw Learn Syst 35(4):4367–4372
30. Liu M, Jiao L, Liu X et al (2020) C-cnn: Contourlet convolutional neural networks. IEEE Trans Neural Netw Learn Syst 32(6):2636–2649
31. Liu W, Zhang Y, Wang J et al (2021) Item relationship graph neural networks for e-commerce. IEEE Trans Neural Netw Learn Syst 33(9):4785–4799
32. Mavromatis C, Karypis G (2021) Graph infoclust: Maximizing coarse-grain mutual information in graphs. In: Pacific-Asia conference on knowledge discovery and data mining, Springer, pp 541–553
33. Newman M (2006) Finding community structure in networks using the eigenvectors of matrices. Phys Rev E Stat Nonlinear Soft Matter Phys 74(3 Pt 2):036104. https://api.semanticscholar.org/CorpusID:138996
34. Ng YC, Colombo N, Silva R (2018) Bayesian semi-supervised learning with graph gaussian processes. Adv Neural Inf Process Syst 31
35. Nichol AQ, Dhariwal P (2021) Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning. PMLR, pp 8162–8171
36. Pal S, Regol F, Coates M (2019) Bayesian graph convolutional neural networks using non-parametric graph learning. arXiv:1910.12132
37. Pan S, Hu R, Long G et al (2018) Adversarially regularized graph autoencoder for graph embedding. arXiv:1802.04407
38. Pei H, Wei B, Chang KCC et al (2020) Geom-gcn: Geometric graph convolutional networks. arXiv:2002.05287
39. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 701–710
40. Rezende DJ, Mohamed S, Wierstra D (2014) Stochastic backpropagation and approximate inference in deep generative models. In: International conference on machine learning. PMLR, pp 1278–1286
41. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer, pp 234–241
42. Shchur O, Mumme M, Bojchevski A et al (2018) Pitfalls of graph neural network evaluation. arXiv:1811.05868
43. Sohl-Dickstein J, Weiss E, Maheswaranathan N et al (2015) Deep unsupervised learning using nonequilibrium thermodynamics. In: International conference on machine learning, PMLR, pp 2256–2265
44. Song Y, Ermon S (2019) Generative modeling by estimating gradients of the data distribution. Adv Neural Inf Process Syst 32
45. Spring N, Mahajan R, Wetherall D et al (2004) Measuring isp topologies with rocketfuel. IEEE/ACM Trans Netw 12(1):2–16. https://doi.org/10.1109/TNET.2003.822655
46. Tan Z, Chen J, Kang Q et al (2021) Dynamic embedding projection-gated convolutional neural networks for text classification. IEEE Trans Neural Netw Learn Syst 33(3):973–982
47. Tang L, Liu H (2011) Leveraging social media networks for classification. Data Min Knowl Discov 23:447–478. https://doi.org/10.1007/s10618-010-0210-x

48. Ucar T (2023) Ness: Learning node embeddings from static subgraphs. https://doi.org/10.48550/arXiv.2303.08958. arXiv:2303.08958
49. Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. Adv Neural Inf Process Syst 30
50. Veličković P, Cucurull G, Casanova A et al (2017) Graph attention networks. arXiv:1710.10903
51. Veličković P, Fedus W, Hamilton WL et al (2018) Deep graph infomax. arXiv:1809.10341
52. Vladimir Batagelj AM (2006) Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/
53. Von Mering C, Krause R, Snel B et al (2002) Comparative assessment of large-scale data sets of protein-protein interactions. Nature 417(6887):399–403
54. Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world'networks. Nature 393(6684):440–442
55. Wu F, Souza A, Zhang T et al (2019) Simplifying graph convolutional networks. In: International conference on machine learning. PMLR, pp 6861–6871
56. Wu L, Gong C, Liu X et al (2022) Diffusion-based molecule generation with informative prior bridges. Adv Neural Inf Process Syst 35:36533–36545
57. Wu Z, Pan S, Chen F et al (2020) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24
58. Yang Z, Cohen W, Salakhudinov R (2016) Revisiting semi-supervised learning with graph embeddings. In: International conference on machine learning. PMLR, pp 40–48
59. Yin M, Zhou M (2018) Semi-implicit variational inference. In: International conference on machine learning. PMLR, pp 5660–5669
60. Zhang Y, Pal S, Coates M et al (2019) Bayesian graph convolutional neural networks for semi-supervised classification. In: Proceedings of the AAAI conference on artificial intelligence. AAAI, pp 5829–5836

Authors and Affiliations

Hailong Su[1] · Zhipeng Li[2] · Chang-An Yuan[3] · F. Filaretov Vladimir[4] · De-Shuang Huang[5,6]

✉ Hailong Su
hailongsu@tongji.edu.cn

✉ De-Shuang Huang
dshuang@eias.ac.cn

Zhipeng Li
lizhipengqilu@gmail.com

Chang-An Yuan
yca@gxtc.edu.cn

F. Filaretov Vladimir
filaretov@inbox.ru

[1] School of Electronics and Information Engineering, Tongji University, 200082 Shanghai, China

[2] University of Science and Technology of China, Hefei 230000, Anhui, China

[3] Institute of Big Data and Intelligent Computing Research Center, Guangxi Academy of Science, Nanning 530000, Guangxi, China

[4] Institute of Automation and Control Processes, Far Eastern Branch of Russian Academy of Sciences, Vladivostok 690041, Vladivostok, Russia

[5] Ningbo Institute of Digital Twin, Eastern Institute of Technology, Ningbo 315201, Zhejiang, China

[6] Institute for Regenerative Medicine, Medical Innovation Center and State Key Laboratory of Cardiology, Shanghai East Hospital, School of Life Sciences and Technology, Tongji University, Shanghai 200123, P. R. China