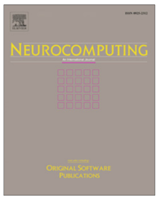
Neurocomputing 523 (2023) 157–169

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Graph representation learning based on deep generative gaussian mixture models

Ghazaleh Niknam [a,1], Soheila Molaei [b,1], Hadi Zare [a,*], David Clifton [b,d], Shirui Pan [c]

[a] Faculty of New Sciences and Technologies, University of Tehran, Iran
[b] Department of Engineering Science, University of Oxford, UK
[c] School of Information and Communication Technology, Griffith University, Australia
[d] Oxford-Suzhou Centre for Advanced Research (OSCAR), Suzhou, China

ARTICLE INFO

Article history:
Received 5 May 2022
Revised 31 August 2022
Accepted 27 November 2022
Available online 9 December 2022
Communicated by Zidong Wang

Keywords:
Graph representation learning
Node embedding
Variational graph auto-encoder
Adversarial mechanism
Gaussian mixture model

ABSTRACT

Graph representation learning is an effective tool for facilitating graph analysis with machine learning methods. Most GNNs, including Graph Convolutional Networks (GCN), Graph Recurrent Neural Networks (GRNN), and Graph Auto-Encoders (GAE), employ vectors to represent nodes in a deterministic way without exploiting the uncertainty in hidden variables. Deep generative models are combined with GAE in the Variational Graph Auto-Encoder (VGAE) framework to address this issue. While traditional VGAE-based methods can capture hidden and hierarchical dependencies in latent spaces, they are limited by the data's multimodality. Here, we propose the Gaussian Mixture Model (GMM) to model the prior distribution in VGAE. Furthermore, an adversarial regularization is incorporated into the proposed approach to ensure the fruitful impact of the latent representations on the results. We demonstrate the performance of the proposed method on clustering and link prediction tasks. Our experimental results on real datasets show remarkable performance compared to state-of-the-art methods.

1. Introduction

Graphs can model and represent the majority of complex systems across a wide range of domains [1]. These structures play a critical role in many applications, including biological protein–protein interaction networks [2], social media [3], transportation networks [4], and citation networks[5]. Recently, many studies have emerged on learning representations to encode structural information of the graph [6–10]. These graph representation learning algorithms convert graph data into a low-dimensional space. Downstream machine learning tasks such as node classification, link prediction, and clustering employ these representations [11].

Graph representation learning approaches can be divided into two categories: shallow embedding and deep embedding. Despite their popularity in recent years, the shallow embedding approaches suffer from some limitations. Some of these limitations can be mentioned as follows: 1) Lack of parameter sharing, 2) Incapability to handle node features efficiently, and 3) Inability to apply in an inductive manner [12,13]. On the other hand, the emergence of deep embedding approaches, aided by the introduction of the Graph Neural Network (GNN) paradigm, assists in overcoming these shortcomings. GNNs can generate representations of nodes based on the graph's structure and any feature information [14,12].

GNNs ignore the data distribution, leading to poor representations and overfitting [6,15]. Combining them with deep generative models to learn data distribution has significantly improved generated representations. Variational Graph Auto-Encoder (VGAE) framework and adversarial approach are widely studied in this scope. VGAE is an unsupervised framework based on deep generative and variational inference models [16,17]. This framework considers the distribution of the encoder's latent representation, which leads to complex data reasoning and, as a result, efficient embedding [18]. Adversarial Graph Auto-Encoder framework enforces the latent representation to match a prior distribution during the training process [12].

This paper considers some assumptions about observed data distribution for use in unsupervised clustering tasks. To accomplish this, we concentrate on modeling the multimodality of observed data in clustering. Probabilistic modeling is mostly performed by considering simplified assumptions about data. Each sample is generated from one well-known distribution like Gaussian, named unimodality assumption. This assumption has some major drawbacks and is too limited for some certain phenomena. Researchers are usually faced with complex data where each

* Corresponding author.
[1] Equal Contribution.

https://doi.org/10.1016/j.neucom.2022.11.087
0925-2312/© 2022 Elsevier B.V. All rights reserved.

sample may come from one of the multiple known (or unknown) distributions named multimodal distribution. Intuitively, multimodal data contain several regions with a high amount of probability.

Technically, these models are called mixture models as a principled modeling approach to deal with such complex data. This paper uses Gaussian Mixture Model (GMM) in Graph Variational Auto-Encoder to model the prior distribution. This combination improves the interpretability of the proposed model. Along with modelling with multimodality through GMM, the proposed method introduces a regularizer based on the adversarial mechanism concept to improve the results. The outline of our contributions is as follows,

- We leverage the VGAE framework by considering the Gaussian Mixture Model (GMM) to model the prior distribution of observed data.
- We introduce a regularization based on the adversarial mechanism concept to ensure that the latent embeddings boost our results.
- We demonstrate our model's performance on clustering and link prediction tasks on real datasets.

2. Related Work

Deep learning methods have sparked widespread interest due to their high performance, efficiency, and ease of use. As a result, deep graph embedding approaches, based on the GNN paradigm, have become the main focus of graph representation learning methods. GNNs can use parameter sharing and generate node representations based on the graph's structure and any feature information. They can also appear inductively. GNNs are useful and widely used structures for the reasons stated above [14,12,13].

Graph Convolutional Network (GCN) [19] is one of the first GNN structures which generalizes the concept of convolutions to graph-structured data [20,19]. Graph Auto-Encoder (GAE) is another approach introduced following the GCN which is an unsupervised approach that generalizes the auto-encoder framework. GAE is a straightforward and robust framework comprised of an encoder and a decoder. A GCN model, which generates latent representations, is frequently used as the encoder. The decoder then reconstructs the input data, which is frequently an inner product of the latent variables [17].

Multi-Task Graph Auto-encoder (MTGAE) [21] is an auto-encoder-based method that can learn a joint representation of local graph structure and available node features. This method aims to learn link prediction and node classification simultaneously. There are some other studies based on GAE framework such as [22–26]. Although GAE-based methods are effective, they disregard data distribution, leading to poor representations and overfitting [6,15]. The GAE framework and deep generative models are combined for this purpose. By accounting for data distribution, deep generative models can represent complex dependencies and interactions between input and output data[27].

VGAE approach originates from deep generative concepts that generalizes the Variational Auto-Encoder (VAE) [17,27] method on graph structure data. In VGAE, the encoder and the decoder are probabilistic. The main idea behind VGAE is that it embeds input data into a distribution rather than a point. A random sample Z is drawn from the distribution rather than generated directly by the encoder[28]. Using the VGAE framework, Xie et al. [29] provided a representation learning method for networked documents to model document contents and relations. Their proposed method reached from the Higher-order Graph Attention Network (HGAT), which examines and shuffles the document's neighborhood information in each order.

Multiresolution Graph Networks (MGN) and Multiresolution Graph Variational Autoencoders (MGVAE) are presented by [30] for using multiresolution and equivariant methods to learn and generate graphs. MGN performs higher-order message passing while designing graph encoders to cluster the graph and coarsen it to a lower resolution. They have encoded the hierarchy of coarsened graphs by MGVAE, which develops a hierarchical generative model based on MGN. NetVAE developed by Jin et al. [31] is a network embedding approach that addresses the orthogonality of network topology information and node attributes. Here, network structure compression and node attributes share the same encoder. On the other hand, a dual decoder, supported by GMM, is introduced for reconstructing network topologies and node attributes separately.

Adversarial-based approaches leverage alternative deep generative frameworks. Adversarially Regularized Variational Graph Autoencoder (ARVGAE) generalizes the adversarial autoencoder framework [32] to graph data [6]. In the ARVGAE, the latent representation is enforced to match a prior distribution by exploiting a min–max game strategy [6]. Random Walk Regularization for Graph Auto Encoders (RWR-VGAE) is a VGAE with a regularizer based on Random Walk with Restarts (RWR). This regularizer is used to capture the information of the immediate neighbour of each node [33]. Lu et al. [34] proposed MRGAE, a network embedding approach to model network consistency across different views. MRGAE generates a second view of the input network based on node content capturing the relationship between them. By incorporating a multiview adversarial regularization module, they ensure consistency between the two views of the network.

Most of the earlier works assumed a Gaussian distribution to model prior uncertainty of the observed data. This assumption suffers from the inefficiency of modeling complex data with properties like multimodality. Some works employed other distributions rather than the traditional Gaussian assumption. Davidson et al. [35] introduced a method called S-VGAE, which considered the von Mises-Fisher (vMF) unimodal density to model the prior distribution to result in a better representation.

Zheng et al. [36] proposed DBGAN, a method for estimating the prior distribution of latent representation in a structure-aware manner. The prior distribution of the latent embedding in this method implicitly connects the graph and feature space through prototype learning, in contrast to the widely used Gaussian distribution assumption. As a result, discriminative and robust representations are generated for all nodes. The CGCN proposed by Hui et al. [37] consists of two main modules, an attributed graph clustering module and a semi-supervised node classification module. This approach investigates a mixture of Gaussian in the clustering module to assign pseudo-labels to unlabeled samples. CGCN ignores the benefits of the adversarial mechanism.

Our proposed method is based on the VGAE framework and uses a variant of the adversarial mechanism. Along with our primary aim of clustering tasks, we consider the multimodal assumption on the observed data through a theoretically and practically well-known GMM model. More specifically, the correspondence of Gaussian density for each presumed cluster results in a more efficient approach. Furthermore, a regularization based on a variant of the adversarial mechanism is proposed to improve the result.

Focusing on positive and negative encoder samplings, we present a novel perspective on the adversarial technique. Unlike the conventional adversarial-based approaches [6,33,34] which treated encoder-generated representations as negative samples, we address them as positive samples. Following our shift from negative samples to positive ones and changing the nature of the regularizer, we appraise the overall loss function. The inverse of KL-divergence is added to the loss function, whereas traditional methods use the min–max technique.

3. Problem Statement

3.1. Notation

A graph G is represented as $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ denotes a set of N nodes and E represents a set of ε edges in this graph. $\boldsymbol{\chi} \in R^{N\times d}$ denotes a feature matrix where each row shows a d-dimensional feature vector for each node v_i in the graph. $\mathbf{A} \in R^{N\times N}$ represents the adjacency matrix where $A_{i,j} = 1$ if $e_{i,j} \in E$, otherwise $A_{i,j} = 0$. The goal is to map the nodes $v_i \in V$ to low-dimensional vectors $\mathbf{z}_i \in R^F$. The mapping function is as follows: $f : (\mathbf{A}, \boldsymbol{\chi}) \rightarrow \mathbf{Z}$, where $\mathbf{Z} \in R^{N\times F}$. Each node embedding $\mathbf{z}_i \in \mathbf{Z}$ is an F-dimensional vector in latent space, where $F \ll d$. Table 1 summarizes the notations used in this paper.

4. Preliminaries

4.1. Variational Graph Auto-Encoders

VGAE was introduced by [28]. This framework is a generalization of the VAE framework [27,17] to graph structure data. There are two values z (embeddings) and x (input data), in the VAE framework. Based on the Bayes theorem, the posterior distribution is calculated as shown in Eq. 1.

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} = \frac{p_\theta(x|z)p_\theta(z)}{\int p_\theta(z)p_\theta(x|z)dz} \tag{1}$$

Since the true posterior density, shown in Eq. 1, is mostly intractable, Rezende et al. [27] introduced a recognition model $q_\phi(z|x)$ as an approximation for the posterior distribution. The recognition model commonly chooses Gaussian distribution with a diagonal covariance structure. Consequently, the recognition model (or inference model), $q_\phi(z|x)$ is the probabilistic encoder of VAE, and the generative model, $p_\theta(x|z)$ is the probabilistic decoder of VAE. A Graphical model for VAE is shown in Fig. 1.

Inference model of VGAE. In VGAE, the inputs are the adjacency matrix A, and the feature matrix $\boldsymbol{\chi}$. Each node in these inputs maps to a latent vector as shown in Eq. 2.

Table 1
The summary of notations.

Symbols	Meaning
$\mathbf{A}$	The adjacency matrix of graph G
N	Number of nodes in the graph
$\boldsymbol{\chi}$	The features matrix of G
F	Number of features in χ
$D_{KL}(q\|\|p)$	KL-divergence between q and p
$\mathbf{z}$	The latent variable in Variational Inference
$\mathbf{X}, \mathbf{W}, \mathbf{C}$	The latent variables in the proposed method
$\mathbf{X}', \mathbf{W}', \mathbf{C}'$	The latent variables in the proposed method of fake data for regularization part
ϕ	The set of parameters of the neural network in the encoder part
θ	The set of parameters of the neural network in the decoder part
β	The set of parameters of the GNN related to each GMM component in the proposed method
ϕ_C	The set of parameters of the GNN related to C in variational factors in the proposed method
ϕ_W	The set of parameters of the GNN related to W in variational factors in the proposed method
K	Number of components in GMM
π	Mixing probability of GMM

$$\begin{aligned} q_\phi(\mathbf{Z}|\mathbf{A}, \boldsymbol{\chi}) &= \prod_{i=1}^{N} q_\phi(\mathbf{z}_i|\mathbf{A}, \boldsymbol{\chi}), \\ q_\phi(\mathbf{z}_i|\mathbf{A}, \boldsymbol{\chi}) &= \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \end{aligned} \tag{2}$$

Here $\boldsymbol{\mu}$ represents a matrix of mean vectors and $\boldsymbol{\Sigma}$ denotes covariance matrix. $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ denote i-th vector of these matrices. The distribution of the recognition model is parameterized by two GNNs as presented in Eq. 3.

$$\begin{aligned} \boldsymbol{\mu} &= GNN_\mu(\mathbf{A}, \boldsymbol{\chi}) \\ \boldsymbol{\Sigma} &= GNN_\Sigma(\mathbf{A}, \boldsymbol{\chi}) \end{aligned} \tag{3}$$

Here, GNN_μ and GNN_Σ are two GNNs that produce $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. These GNNs can be any of the different types of GNNs such as GCN [19], GCN with Chebyshev filters [38], GAT (Graph Attention Network)[39], and GraphSAGE [40]. The two-layer GCN shown in Eq. 4 is commonly used for this purpose [28].

$$GCN(\mathbf{A}, \boldsymbol{\chi}) = \widehat{\mathbf{A}} ReLU\left(\widehat{\mathbf{A}}\boldsymbol{\chi}\mathbf{W}_0\right)\mathbf{W}_1 \tag{4}$$

Here, $\mathbf{W}_0$ and $\mathbf{W}_1$ are the weight matrices of each layer, and $\mathbf{W}_0$ is shared in the first layer of two GCNs. $\widehat{\mathbf{A}}$ is the normalized adjacency matrix introduced as follows,

$$\begin{aligned} \widetilde{\mathbf{A}} &= \mathbf{A} + \mathbf{I} \\ \widetilde{D}_{ii} &= \sum_j \widetilde{A}_{ij} \\ \widehat{\mathbf{A}} &= \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}} \end{aligned} \tag{5}$$

In Eq. 5, $\mathbf{I}$ represents the identity matrix of size $\mathbf{A}$, and $\mathbf{D}$ represents degree matrix of the graph.

Generative model of VGAE. After inferring the latent vectors, the adjacency matrix can be reconstructed with sampling from $p_\theta(\mathbf{A}|\mathbf{Z})$. The generative model can be any kind of GNN, but most of the time an inner product between latent variables as shown in Eq. 6, is used for this purpose.

$$\begin{aligned} p(\mathbf{A}|\mathbf{Z}) &= \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j) \\ p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j) &= Sigmoid(\mathbf{z}_i^T\mathbf{z}_j) \end{aligned} \tag{6}$$

Here, $\mathbf{z}_i$ and $\mathbf{z}_j$ are the ith and jth vectors of $\mathbf{Z}$ and *Sigmoid* denotes the logistic sigmoid function.

Learning of VGAE. To learn the model, the approximate posterior, which is parameterized by variational parameters ϕ, should be close to the true posterior. The Kullback–Leibler divergence (KL-divergence) is used for this purpose. KL-divergence is a metric that measures the distance between two distributions. The goal is to minimize this KL-divergence as shown in Eq. 7 [28].

$$\phi = argmin_\phi D_{KL}\left[q_\phi(\mathbf{Z}|\mathbf{A}, \boldsymbol{\chi})||p_\theta(\mathbf{Z}|\mathbf{A}, \boldsymbol{\chi})\right] \tag{7}$$

In this equation, D_{KL} indicates KL-divergence, ϕ represents the parameters of posterior estimation $(q_\phi(\mathbf{Z}|\mathbf{A}))$, and θ denotes parameters of real posterior $(p_\theta(\mathbf{Z}|\mathbf{A}, \boldsymbol{\chi}))$. Calculation of KL-divergence depends on log marginal likelihood, which is intractable. Instead, maximizing the Evidence Lower Bound (ELBO), shown in Eq. 8, is used for the learning process [28].

$$L_{ELBO} = \mathbb{E}_{q_\phi}\left(\log\frac{p_\theta(\mathbf{A}|\mathbf{Z})}{q_\phi(\mathbf{Z}|\mathbf{A}, \boldsymbol{\chi})}\right) \tag{8}$$

Which can be rewritten as,

$$L_{ELBO} = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{A},\boldsymbol{\chi})}[\log p_\theta(\mathbf{A}|\mathbf{Z})] - D_{KL}\left(q_\phi(\mathbf{Z}|\mathbf{A}, \boldsymbol{\chi})||p_\theta(\mathbf{Z})\right) \tag{9}$$

L_{ELBO} denotes the ELBO loss function, which has two terms. The first term is associated with data reconstruction in a generative model. The second term (the KL-divergence term) is a loss function

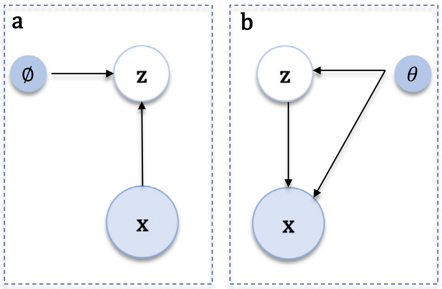


Fig. 1. Graphical models for Variational Auto-Encoders. **a**. The recognition or inference model, which maps input data to the latent variable **z**. **b**. The generative model, which reconstructs the input data by the latent variable.

regularizer that measures the distance between the posterior estimation $q_\phi(\mathbf{Z}|\mathbf{A},\boldsymbol{\chi})$ and the prior distribution $p_\theta(\mathbf{Z})$. This equation should be optimized concerning θ and ϕ by some optimization method such as SGD (Stochastic Gradient Descent), Adagrad, Adam, or mini-batch stochastic gradient descent [41,42].

4.2. Gaussian Mixture Models

GMM belongs to the parametric probability density function family. It is a weighted sum of K components. Each of these components follows a multivariate Gaussian distribution. The weight of mixing, also known as mixture probability or mixture coefficient, is denoted by π_i for i-th component, where $\sum_{i=1}^{K}\pi_i = 1$. The components of GMM can capture the multimodal nature of the data [43].

5. The proposed model

5.1. Gaussian Mixture Variational Graph Auto-Encoder

In regular VGAE, the prior distribution over the latent variables is assumed to be a Gaussian distribution [28]. While VGAE is applied to many domains, it suffers from facing multimodality observed data. Fig. 2 shows the general framework of the regular VGAE and our proposed method, called Gaussian Mixture Variational Graph Auto-Encoder (GM-VGAE). There is a set of latent variables $\mathbf{X},\mathbf{W}$, and $\mathbf{C}$ in the GM-VGAE instead of just one latent variable in VGAE.

Generative model of GM-VGAE. The definition of the generative model, based on the latent variables and the observed sample, is shown in Eq. 10.

$$\begin{aligned}\mathbf{W} &\sim \mathcal{N}(0,I)\\ \mathbf{C} &\sim Mult(\pi)\\ \mathbf{X}|\mathbf{W},\mathbf{C} &\sim \prod_{k=1}^{K}\mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{c}_k}(\mathbf{W};\beta),\boldsymbol{\Sigma}_{\mathbf{c}_k}(\mathbf{W};\beta)\right)^{\mathbf{c}_k}\end{aligned} \tag{10}$$

Here, K represents the number of components in the mixture model. This parameter is defined as a hyperparameter of the model. $\mathbf{W}$ follows a Gaussian distribution with mean zero and covariance matrix $\mathbf{I}$. Eventually, $\mathbf{C}$ is a one-hot vector, which represents the mixing coefficient of the Gaussian mixture components. This vector is sampled from the mixing probability, which is denoted by π. Here, π is equal to $\frac{1}{K}$ so that, it is uniformly distributed.

In this model, there is a GNN parameterized by β, which $\mathbf{W}$ feeds to it as input. This GNN produces a set of K numbers of $(\boldsymbol{\mu}_{\mathbf{c}_k})$ and $(\boldsymbol{\Sigma}_{\mathbf{c}_k})$. Here, each of $\boldsymbol{\mu}_{\mathbf{c}_k}$ and $\boldsymbol{\Sigma}_{\mathbf{c}_k}$ computed with a GCN with parameter sharing. Finally, according to the definitions provided, $\mathbf{X}|\mathbf{W}$ is a Gaussian mixture. The adjacency matrix $\mathbf{A}$ is reconstructed from another GNN parameterized by θ. The generative model is shown in Eq. 11.

$$\mathbf{A}|\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{X};\theta),\boldsymbol{\Sigma}(\mathbf{X};\theta)) \tag{11}$$

In this study, the choice of decoder is followed by [28] to avoid complexity. So we use an inner product between latent variables to reconstruct the adjacency matrix instead of a GNN. This decoder is shown in Eq. 12.

$$\begin{aligned}p(\mathbf{A}|\mathbf{X}) &= \prod_{i=1}^{N}\prod_{j=1}^{N}p(A_{ij}|\mathbf{x}_i,\mathbf{x}_j)\\ p(A_{ij}|\mathbf{x}_i,\mathbf{x}_j) &= Sigmoid(\mathbf{x}_i^T\mathbf{x}_j)\end{aligned} \tag{12}$$

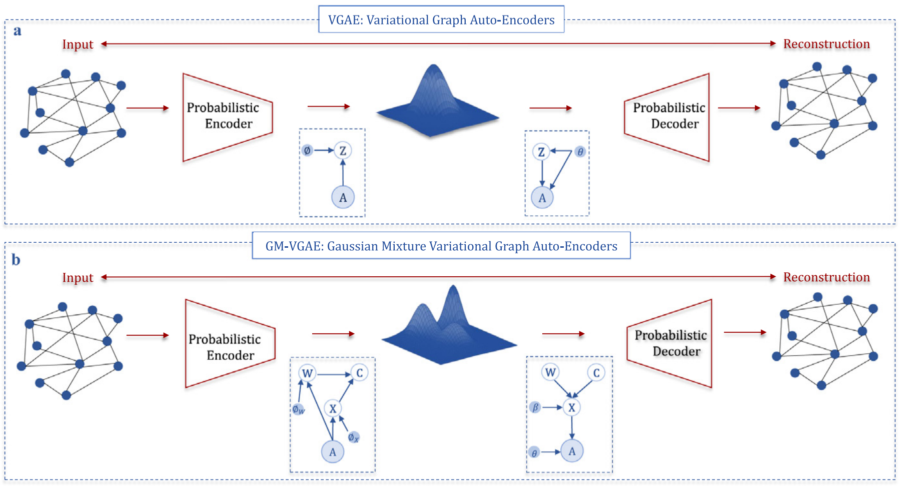


Fig. 2. Comparison of the general framework of Variational graph auto-encoder (VGAE) and Gaussian mixture Variational graph auto-encoder(GM-VGAE). a) In VGAE, the input data as an adjacency matrix **A** maps to a unimodal latent space. The inference model (encoder), with parameter ϕ, maps input data to the latent variable **Z**. The generative model (decoder), with parameter ϕ, reconstructs the input data by the latent variable. b) In GM-VGAE, input data (**A**) maps to a multimodal latent space. This latent space comprises three latent variables X, W, and C, which follow a GMM. The inference model maps input data to the latent variables. ϕ_C and ϕ_W are GNN parameters related to C and W in variational factors of the encoder, respectively. The decoder with parameters θ and β reconstructs the input data..

Here A_{ij} represents the elements of the adjacency matrix. Note that in this case, the decoder, which generates the reconstruction of observed data, lacks parameters.

Inference model of GM-VGAE. Based on the mean-field variational family,

$$q(\mathbf{X},\mathbf{W},\mathbf{C}|\mathbf{A},\chi)=\prod_{i=1}^{N}q_{\phi_\mathbf{X}}(\mathbf{x}_i|\mathbf{A}_i,\chi_i)q_{\phi_\mathbf{W}}(\mathbf{w}_i|\mathbf{A}_i,\chi_i)p_\beta(\mathbf{c}_i|\mathbf{x}_i,\mathbf{w}_i) \quad (13)$$

In this equation, β represents the set of GNN parameters related to each GMM component, ϕ_X demonstrates the set of GNN parameters related to X in variational factors, and ϕ_W shows the set of GNN parameters related to W in variational factors. The z-posterior is as follows,

$$p_\beta(\mathbf{c}_j=1|\mathbf{X},\mathbf{W}) = \frac{p(\mathbf{c}_j=1)p(\mathbf{X}|\mathbf{c}_j=1,\mathbf{W})}{\sum_{k=1}^{K}p(\mathbf{c}_k=1)p(\mathbf{X}|\mathbf{c}_j=1,\mathbf{W})}=\frac{\pi_j\mathcal{N}(\mathbf{X}|\mu_j(\mathbf{W};\beta),\sigma_j(\mathbf{W};\beta)}{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{X}|\mu_k(\mathbf{W};\beta),\sigma_k(\mathbf{W};\beta)} \quad (14)$$

The graphical illustration of generative and inference models of our method is demonstrated in Fig. 3.

Learning of GM-VGAE. The ELBO of the proposed model is shown in Eq. 15.

$$L_{ELBO}=\mathbb{E}_q\left(\frac{p(\mathbf{A},\mathbf{X},\mathbf{W},\mathbf{C})}{q(\mathbf{X},\mathbf{W},\mathbf{C}|\mathbf{A},\chi)}\right) \quad (15)$$

in which,

$$p(\mathbf{A},\mathbf{X},\mathbf{W},\mathbf{C})=p(\mathbf{W})p(\mathbf{C})p(\mathbf{X}|\mathbf{W},\mathbf{C})p(\mathbf{A}|\mathbf{X}) \quad (16)$$

Based on Eq. 13 and Eq. 16 the ELBO can be written as,

$$\begin{aligned}L_{ELBO}=&\ \mathbb{E}_{q(\mathbf{X}|\mathbf{A},\chi)}[\log p(\mathbf{A}|\mathbf{X})]\\ &-\mathbb{E}_{q(\mathbf{W}|\mathbf{A},\chi)p(\mathbf{C}|\mathbf{X},\mathbf{W})}\left[KL\left(q_{\phi_\mathbf{X}}(\mathbf{X}|\mathbf{A},\chi)||p_\beta(\mathbf{X}|\mathbf{W},\mathbf{C})\right)\right]\\ &-KL\left(q_{\phi_\mathbf{W}}(\mathbf{W}|\mathbf{A},\chi)||p(\mathbf{W})\right)\\ &-\mathbb{E}_{q(\mathbf{X}|\mathbf{A},\chi)q(\mathbf{W}|\mathbf{A},\chi)}[KL(p_\beta(\mathbf{C}|\mathbf{X},\mathbf{W})||p(\mathbf{C}))]\end{aligned} \quad (17)$$

Here i indices have been dropped to more simplify the notations and presume one node at each time. In this equation, the first term is the reconstruction term and the following are X-conditional prior term, W-prior term, and C-prior term [44] in order. The reconstruction term is used to calculate the difference between the input and the reconstruction. This term has the same form as the VGAE's ELBO reconstruction term. It can be estimated using Monte-Carlo samples from $q(X|A,\chi)$. The conditional prior term has a negative sign and a positive value. This term should be small to maximize the objective function. As a result, the conditional prior term serves as a regularizer. To approximate this term, Monte Carlo can be used as shown in Eq. 18.

$$\begin{aligned}&\mathbb{E}_{q(\mathbf{W}|\mathbf{A},\chi)p(\mathbf{C}|\mathbf{X},\mathbf{W})}\left[KL\left(q_{\phi_\mathbf{X}}(\mathbf{X}|\mathbf{A},\chi)||p_\beta(\mathbf{X}|\mathbf{W},\mathbf{C})\right)\right]\\ &\approx\frac{1}{M}\sum_{j=1}^{M}\sum_{k=1}^{K}p_\beta(\mathbf{c}_k=1|\mathbf{x}^j,\mathbf{w}^j)KL\left(q_{\phi_\mathbf{x}}(\mathbf{X}|\mathbf{A},\chi)||p_\beta(\mathbf{X}|\mathbf{w}^j,\mathbf{c}_k=1)\right)\end{aligned} \quad (18)$$

The **W**-prior term in the ELBO equation is also a regularizer and can be computed analytically. **C** is a discrete latent variable in our model. **C**-posterior measures how far **X** is from each cluster position created by **W** to determine cluster assignment probability. ELBO's **C**-prior term tries to reduce the KL divergence between the **C**-posterior and the uniform prior. It would aim to bring the clusters together by maximizing overlap and bringing the means closer together. This term has a positive value and appears with a negative sign in the ELBO equation, indicating that it is a regularizer. Fig. 4 shows a high-level overview of our proposed method.

5.2. Regularizer based on adversarial mechanism concept

This paper introduces an additional regularizer inspired by the adversarial mechanism concept. In the adversarial mechanism, a generator provides fake data and tries to deceive a discriminator. For this purpose, the cost function is changed in such a way to involve the competition between the generator and the discriminator. Here, to generate fake data, the feature matrix of input data (χ) and the hidden variables inferred from the input data (**X** and **W**) concatenate together as $concat(\mathbf{X},\mathbf{W},\chi)$. Then a shuffling process performs on them as $Shuffle(concat(\mathbf{X},\mathbf{W},\chi))$. After that, these fake data are fed to the inference model, and hidden variables of fake data are inferred $(\mathbf{X}',\mathbf{W}',\mathbf{C}')$. In the following, the regularization part of the ELBO is calculated by these fake latent variables, and the negative inverse of that is added to the ELBO. Thus, if KL-divergence cannot detect the difference between the fake and real, the denominator is low, so the fraction is high, and an additional negative burden is added to the ELBO. This part acts as the discriminator. The process is shown in Fig. 4. Using the loss function defined for our model and the newly introduced regularizer, the ELBO can be rewritten as L_{RELBO} as shown in Eq. 19. To simplify the equation, subscripts have been removed.

$$\begin{aligned}L_{RELBO}=L_{ELBO}-&\left[\mathbb{E}\left[KL\left(q(\mathbf{X}'|\mathbf{A},\chi')\right)||q(\mathbf{X}|\mathbf{A},\chi)\right]\right.\\ &+KL\left(q(\mathbf{W}'|\mathbf{A},\chi')||q(\mathbf{W}|\mathbf{A},\chi)\right)\\ &\left.+\mathbb{E}\left[KL(p(\mathbf{C}'|\mathbf{X}',\mathbf{W}'))||p(\mathbf{C}|\mathbf{X},\mathbf{W}))\right]\right]^{(-1)}\end{aligned} \quad (19)$$

A summary of the main steps of GM-VGAE is as follows:

- Encoder
  - Inferring latent variables $(\mathbf{X},\mathbf{W},\mathbf{C})$ from real data
- Regularizer
  - Concatenating $(\mathbf{X},\mathbf{W},\chi)$ named *Temp*
  - Generating Fake Data by Shuffling *Temp*
  - Inferring latent variables from Fake Data $(\mathbf{X}',\mathbf{W}',\mathbf{C}')$
- Decoder
  - Reconstructing the input data based on latent variables inferred from real data
  - Evaluating the objective function: 1) Reconstruction term, 2) Regularization terms based on $(\mathbf{X},\mathbf{W},\mathbf{C})$, and 3) Additional regularization terms based on $(\mathbf{X}',\mathbf{W}',\mathbf{C}')$

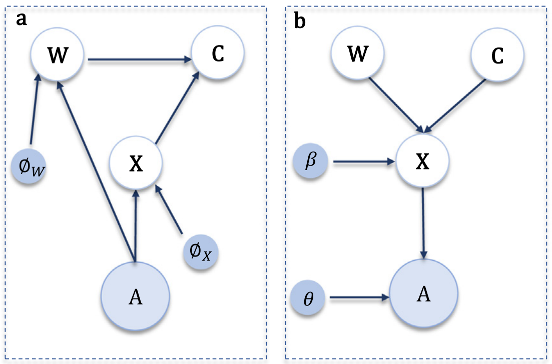


Fig. 3. Graphical models for Gaussian mixture variational graph autoencoder. **a.** The recognition or inference model, which maps input data to the latent variables **X**, **W**, and **C**. **b.** The generative model reconstructs the input data by the latent variables..

6. Experiments

In this section, the results of our experiments are given. These experiments are performed to compare the proposed method

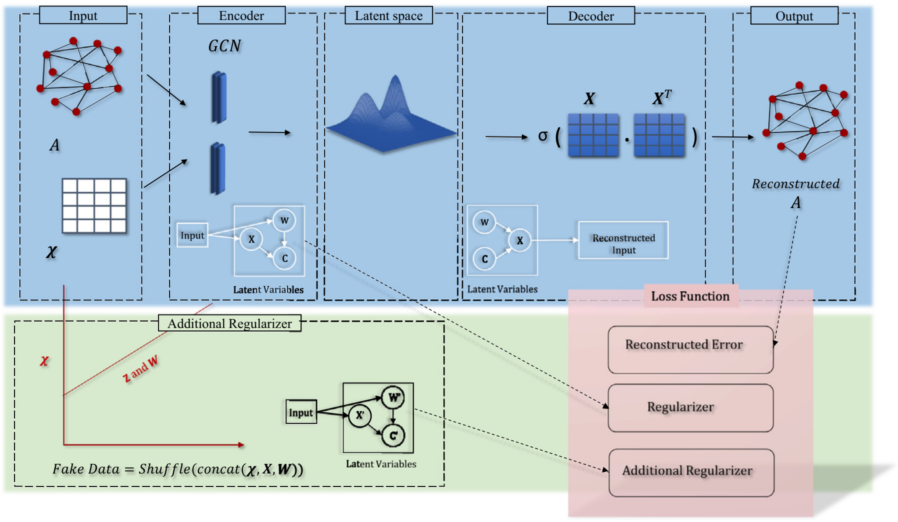


Fig. 4. A high-level overview of our method. In the upper half of the figure, the adjacency matrix and the feature matrix are fed into the GCN-based probabilistic encoder. **X**, **W**, and **C** are the latent variables inferred from the encoder. The decoder reconstructs the adjacency matrix using these latent variables by the inner product of latent vectors. The lower half of the figure is related to the additional regularizer that needs to generate fake data. The cost function consists of three parts, the first part measures the reconstruction error, and the second part is the KL-divergence-based regularizer. The third part is related to the discriminator, in which the fake data is generated from the shuffled combination of the adjacency matrix, **X** and **W**..

Table 2
Summary of the citation network datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3
Cornell	183	295	1703	5
Wisconsin	251	499	1703	5
Texas	183	309	1703	5

versus the state-of-the-art algorithms. We demonstrate the efficiency of our approach in two downstream machine learning tasks, clustering, and link prediction.

6.1. Datasets

Experiments on six graph datasets were performed. The first three of them, Cora, Citeseer, and Pubmed, are three well-known citation networks [45,46]. Furthermore, we utilized Cornell, Wisconsin, and Texas (WebKB dataset) datasets following [47]. Table 2 reports a summary of these datasets.

Citation networks. Cora, Citeseer, and Pubmed fall into this category. The datasets include sparse bag-of-words feature vectors for each document and a list of document-to-document citation links. In these datasets, nodes correspond to publications and (undirected) edges to citations, i.e., there would be an undirected connection between them whenever a document cites another document. Using the citation links as edges, we create a binary, symmetric adjacency matrix **A**. A class label was assigned to each document.

WebKB. Cornell, Wisconsin, and Texas are webpage datasets. Carnegie Mellon University collected these datasets from computer science departments of different universities. Nodes denote web pages, and edges represent hyperlinks between these web pages. The datasets include bag-of-words feature vectors for each web page. The web pages are divided into five classes, project, student, staff, course, and faculty.

6.2. Baselines and state-of-the-art methods

We compared the performance of our method with state-of-the-art algorithms:

Spectral Clustering [48]**:** This approach seeks to learn social embedding, which generates representations from eigenvectors of the normalized graph Laplacian of G.

DeepWalk [49]**:** This approach learns latent representations, which these representations encode social relations in a continuous vector space. To do so, Deepwalk learns structural regularities present within short random walks.

SEAL[50]**:** This approach is a link prediction framework that creates a local subgraph around each target link and uses GNN to learn a function to convert the patterns of the subgraph into link existence.

GAE[28]**:** This approach is an unsupervised framework for graph structure data, consisting of an encoder and a decoder. This approach leverages both topological and content information.

VGAE [28]**:** This approach is based on the GAE framework, which consists of a probabilistic encoder and a probabilistic decoder.

ARGA [6]**:** This approach is based on the GAE. This method enforces the latent representation to match a prior distribution via an adversarial training scheme.

ARVGA [6]**:** This approach is VGAE based version of ARGA.

RWR-GAE [33]**:** This approach is based on the GAE, which uses a random walk-based method to regularize the representations learned by the encoder.

RWR-VGAE [33]**:** This approach is VGAE based version of RWR-GAE.

S-VGAE [35]**:** This approach is based on VGAE, which consider the von Mises-Fisher (vMF) distribution as the prior distribution for link prediction purpose.

DGI [51]**:** In this method, there are some graph patch representations obtained by the robust convolutional graph architecture. Maximization of local mutual information in these graph patch representations can be used to obtain node embeddings that account for the graph's global structural properties. As the number of hidden units has a significant impact on the performance of this method, we employ the technique in two settings: once with 128 hidden units (DGI-128) and once with 512 hidden units (DGI-512).

GIC [52]: This approach is an unsupervised learning technique for representation learning in graphs. GIC recognizes nodes with similar representations, clusters them, and maximizes mutual information based on the DGI technique.

HGCAE-P [53]: This approach is a GAE-based representation learning method that derives its representation from a geometry-aware message passing auto-encoder. All operations in auto-encoding are performed in hyperbolic space.

CGCN [37]: CGCN consists of an attributed graph clustering module and a semi-supervised node classification module that enhances the learning ability of semi-supervised node classification using samples with clustering assignments.

6.3. Tasks

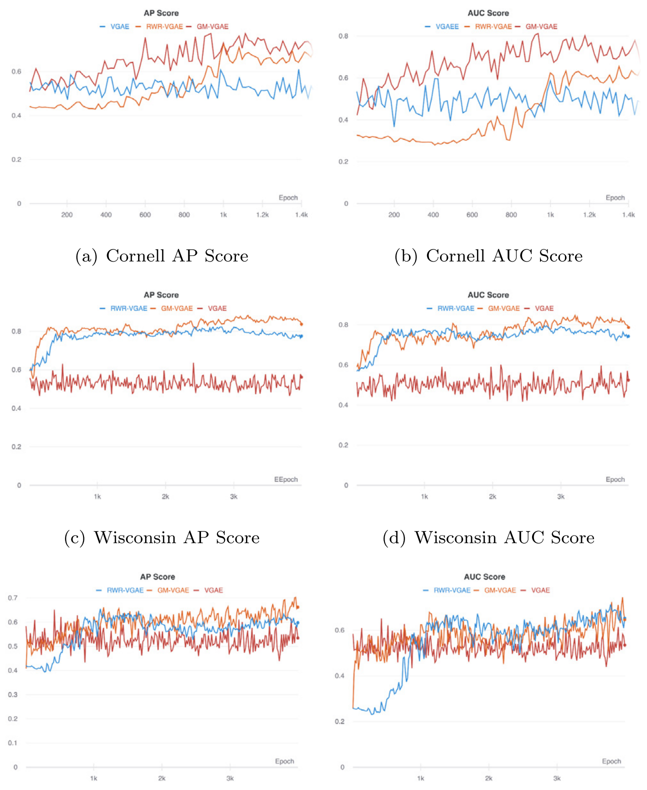
In this study, we perform two tasks, clustering and link prediction. In the link prediction task, the edges and non-edges among the test set nodes are predicted after reconstructing the input graph using a decoder. On the other hand, the K-means clustering algorithm is performed on learned node embeddings to specify the clusters.

6.4. Metrics

In this paper, in order to evaluate the efficiency of clustering, the following criteria were used by [54]: Average rand index (ARI), normalized mutual information (NMI), precision, F-score (F1), and accuracy (ACC). On the other hand, the average precision (AP) and the area under a receiver operating characteristic curve (AUC) were used following [28] to evaluate the link prediction task.

6.5. Settings

Our model is initialized using Glorot initialization [55] and trained with an initial learning rate of 0.01 for a maximum of 100 epochs using Adam optimizer [56]. We terminate the training if the validation accuracy does not improve for 10 consecutive steps; as a result, most runs finish in less than 200 steps. It applied a fixed dropout rate [57] of 0.5 to the input and hidden layers. Also, we considered $\mathscr{L}_2$ regularization of 0.0005 on the weights. We use a two-layer GCN model as an encoder with the parametric ReLU (PReLU) [58] no-linearity. For training the VAE, we use hyper-parameters provided by [28] and apply full-batch gradient descent while using the reparameterization trick [17].



(a) Cornell AP Score (b) Cornell AUC Score

(c) Wisconsin AP Score (d) Wisconsin AUC Score

(e) Texas AP Score (f) Texas AUC Score

Fig. 5. AP and AUC Scores on Cornell, Wisconsin and Texas Datasets.

Table 4
Performance comparison of different models for the Clustering task on Cora.

Model	Acc	NMI	F1	Precision	ARI
SC	36.7	12.7	31.8	19.3	3.1
DW	48.4	32.7	39.2	36.1	24.3
GAE	59.6	37.4	59.5	59.6	27.4
VGAE	62.5	37.1	62.5	62.5	31.9
ARGA	66.8	48.9	66.8	66.8	42.2
ARVGA	54.4	43.3	54.4	54.4	31.0
RWR-GAE	59.3	43.1	57.7	57.7	34.1
RWR-VGAE	57.7	43.1	57.7	57.7	37.2
CGCN	71.5	54.4	67.7	71.5	48.2
DGI-128	59.0	38.6	60.6	59.0	33.6
DGI-512	71.3	56.4	67.2	71.3	51.1
GIC	72.5	53.7	69.2	72.5	50.8
HGCAE-P	74.6	**59.9**	71.73	74.3	52.0
GM-VGAE	**74.8**	59.7	**73.9**	**74.8**	**52.2**

Table 3
Performance comparison of different models for the Link Prediction task.

Model	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
SC	84.6	88.5	80.5	85.0	84.2	87.8
DW	83.1	85.0	80.5	83.6	84.4	84.1
SEAL	91.8	92.9	87.5	88.7	95.4	95.5
GAE	91.0	92.0	89.5	89.9	96.4	96.5
VGAE	91.4	92.6	90.8	92.0	94.4	94.7
ARGA	92.4	93.2	91.9	93.0	96.8	97.1
ARVGA	92.4	92.6	92.4	93.0	96.5	96.8
RWR-GAE	92.9	92.7	92.1	91.5	96.2	96.3
RWR-VGAE	92.6	92.7	92.3	92.4	95.3	95.2
s-VGAE	92.7	93.2	90.3	91.5	97.1	97.1
DGI	89.8	89.7	95.5	95.7	91.2	92.2
GIC	93.5	93.3	97	96.8	93.7	93.5
CGCN	96.38	96.25	96.01	96.25	96.9	95.56
HGCAE-P	95.6	95.5	96.7	97	96.2	96
GM-VGAE	**99.0**	**98.7**	**99.2**	**99.1**	**98.7**	**98.5**

Table 5
Performance comparison of different models for the Clustering task on Citeseer.

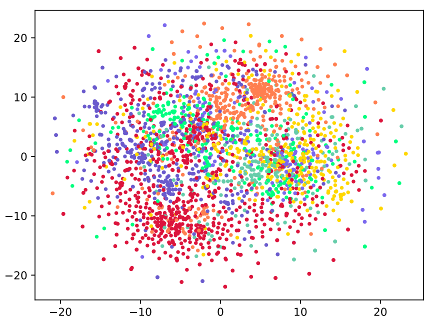
Model	Acc	NMI	F1	Precision	ARI
SC	23.9	5.6	29.9	17.9	3.1
DW	33.7	8.8	27.0	24.8	24.3
GAE	41.2	14.2	41.2	41.2	9.7
VGAE	39.1	13.3	39.1	39.1	7.0
ARGA	50.8	26.9	50.8	50.8	21.4
ARVGA	59.7	33.9	59.7	59.7	33.0
RWR-GAE	44.0	25.2	44.0	44.0	18.0
RWR-VGAE	51.9	28.9	51.9	51.9	25.7
CGCN	67.4	42.3	63.2	66.4	43.5
DGI-128	57.9	30.9	53.4	57.2	27.9
DGI-512	68.8	44.4	64.3	67.8	45
GIC	69.6	**45.3**	65	**69.6**	46.5
HGCAE-P	**71.5**	**45.3**	67.2	69.5	**46.9**
GM-VGAE	69.5	43.2	**69.1**	59.1	45.5

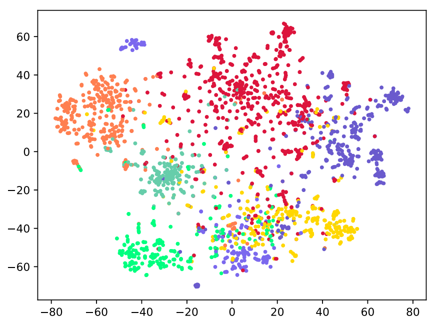
Table 6
Performance comparison of different models for the Clustering task on Pubmed.

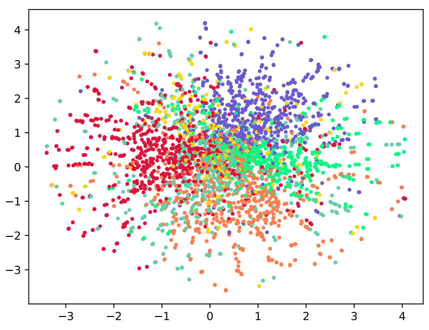
Model	Acc	NMI	F1	Precision	ARI
GAE	67.2	22.4	67.2	67.2	24.5
VGAE	67.3	22.5	67.3	67.3	24.5
ARGA	61.8	21.4	61.8	61.8	19.7
ARVGA	41.8	4.5	41.8	41.8	1.9
RWR-GAE	66.4	26.8	65.1	68.1	26.7
RWR-VGAE	67.2	26.4	67.2	67.3	27.3
CGCN	71	30.2	69.7	71	32.4
DGI-128	49	15.1	45	48.2	14.5
DGI-512	53.3	18.1	47.4	53.1	16.6
GIC	67.3	31.8	65.2	67.3	29.1
HGCAE-P	74.8	**37.7**	73.2	73.2	35.9
GM-VGAE	**74.8**	37.5	**74**	**74.2**	**36.9**

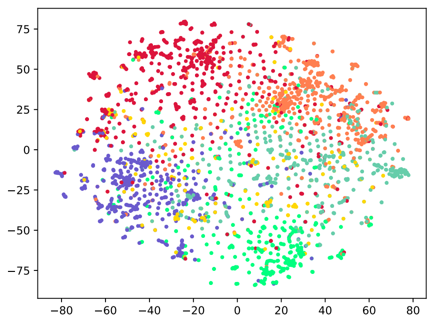
We employ the embedding dimension of 16 for our proposed method. The settings for other methods are as follows. The [59] implementation for SC with an embedding dimension of 128 is used. For DW, we used the implementation provided by [49] with the same settings as in their paper, namely an embedding dimension of 128, 10 random walks of length 80 per node, and a context size of 10, trained for a single epoch. Pytorch implementation of [28] with an embedding dimension of 16 is used for GAE and VGAE. For ARGA and ARVGA, we use the [6] implementation with an embedding dimension of 16.

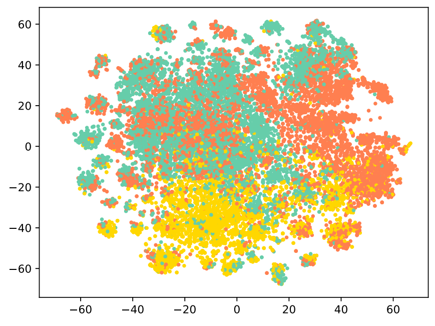
We use an implementation of [33] with hyper-parameters provided by [28] for the autoencoder in RWR-GAE and RWR-VGAE. For the RandomWalk Regularization network's hyper-parameters, the number of walks is set to 50, and the window size and walk length are set to 30. CGCN embedding dimension is set to 16 in [37]


(a) Cora Raw Features


(b) Cora Learned Features


(c) CiteSeer Raw Features


(d) CiteSeer Learned Features


(e) Pubmed Raw Features

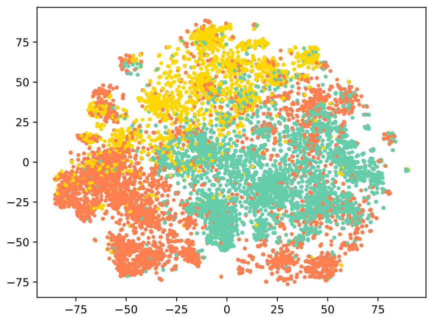

(f) Pubmed Learned Features

Fig. 6. t-SNE embeddings of the nodes in the Cora, CiteSeer, and Pubmed datasets from the raw features (**left**) and features from a learned model (**right**). The clusters of the learned GM-VGAE model's representations are clearly defined.

implementation. DGI implementation is used in two modes, with embedding sizes of 128 and 512. For GIC, we use the implementation of [60] with an embedding size of 16. Eventually, we use the implementation of [53] for HGCAE with embedding size 16.

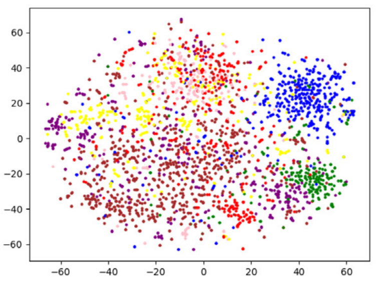
6.6. Complexity

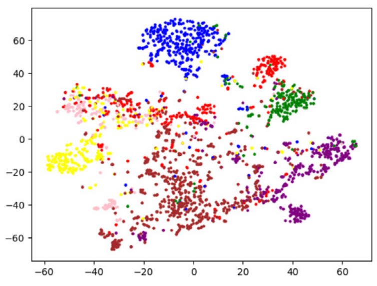
In terms of time complexity, when N is the number of nodes, F is the representation size, and d is the dimension of each node's features, the encoder has a time complexity of $O\left(N^2dF\right)$. An encoder with a complexity of $O(NdF)$ is possible if we deem the adjacency matrix sparse. The time complexity of expectation–maximization steps is $O(N)$. We may change this to $O(NK)$ for large sample sizes. Using an inner product decoder, it has an $O\left(N^2\right)$ time complexity. Overall, $O(NdF) + O(N) + O\left(N^2\right)$ represents the temporal complexity for one epoch.

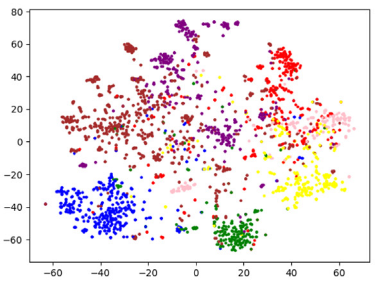
6.7. Results

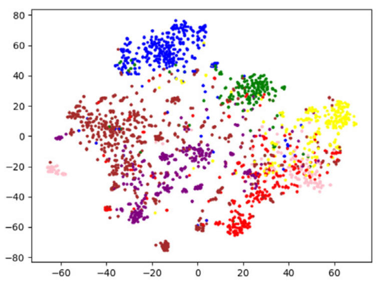
For three citation datasets, Cora, Citseer, and Pubmed, Table 3 shows the mean AP and AUC for the link prediction task over 10 runs. Our proposed method achieves best-in-class results across all three datasets for two assessment criteria. The most promising outcomes are bolded. The comparison methods for the link prediction task can be considered as representative of four categories.

Spectral Clustering and DeepWalk belong to the shallow embedding approaches. SEAL, DGI, and GIC are GCN-based deep embedding approaches. GAE, ARGA, and RWR-GAE are auto-encoder-based methods. VGAE, ARVGA, RWR-VGAE, S-VGAE, HGCAE-p, and CGCN are deep generative-based methods. In a general analysis, it can be seen that deep embedding approaches have improved the results of shallow embedding approaches. Auto-encoder-based approaches have improved the outcomes of deep embedding approaches. DGI and GIC have an implicit auto-encoder structure, which has led to good results. Finally, deep


(a) DeepWalk


(b) DGI


(c) ARGA


(d) RWR-VGAE

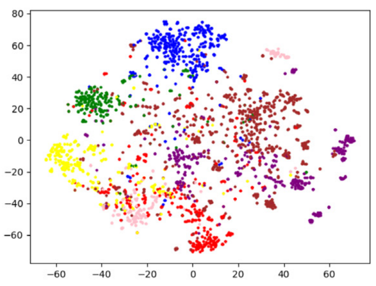

(e) CGCN

Fig. 7. t-SNE embeddings of the nodes in the Cora from the learned features by some of the baseline methods.

generative-based techniques have improved the results of auto-encoder-based approaches. The proposed method has improved the results of all groups. According to Table 3, the comparison between the proposed method and the best results for each group is as follows: GM-VGAE increases AUC by 14.4% and AP by 10.2% compared with Spectral Clustering in Cora, AUC by 18.7%, and AP by 14.1% in Citeseer, AUC by 14.5%, and AP by 10.7% in Pubmed. The analysis of GM-VGAE and GIC demonstrates 5.5% and 5.4%, 2.2% and 2.3% improvement in AUC and AP in the Cora and Citeseer datasets, respectively, and 5% enhancement in both AUC and AP in Pubmed dataset.

GM-VGAE outperforms AUC by 6.1%, 7.1%, 2.51%, and AP by 6%, 7.6%, 2.2% in Cora, Citeseer, and Pubmed datasets, respectively, compared with RWR-GAE. The superior result of previous methods belongs to CGCN. Our proposed method evinces 2.62% and 2.4%, 3.19% and 2.85%, and 1.8% and 2.94% progress in AUC and AP in Cora, Citeseer, and Pubmed, respectively, in contrast to CGCN.

In general, it can be said in the link prediction task that our method showed notable performance on all three benchmarks compared to all state-of-the-art techniques. We repeated the link prediction task on three webpage datasets, Cornell, Wisconsin, and Texas, to further investigate. The results of this experiment are presented in Fig. 5. These results show that our method on these datasets also achieved good results. It can also be seen that, by increasing the number of runs, the model showed more improvement and better performance compared to other methods.

Tables 4–6 show how our model works on three different citation datasets when it comes to node clustering. After 10 runs of each experiment, we announce the mean of the clustering metrics.

Our proposed method shows notable performance on the clustering problem regarding the clustering results on the Cora dataset. The GM-VGAE approach increases accuracy by 38.1%, NMI by 47%, F1 by 42.1%, precision by 55.5 %, and ARI by 49.4% contrary to Spectral Clustering. The maximum results of clustering among the comparative methods on the Cora dataset are provided by HGCAE-P. Our proposed method augment accuracy by 2.2%, F1 by 2.17%, precision by 0.5%, and ARI by 0.2% against HGCAE-P.

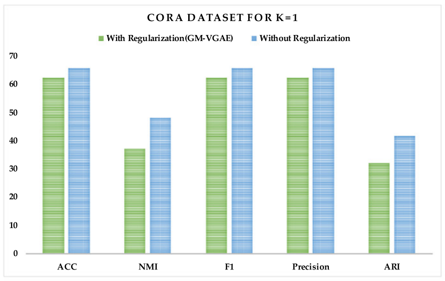
In general, the results of our proposed method on Cora and PubMed showed a significant superiority over other methods in the clustering task. Moreover, our method also represented remarkable performance on CiteSeer in this task over all of the methods but HGCAE-P; however, our results slightly vary with this model.

Table 7
Effect of parameter k on the result of clustering.

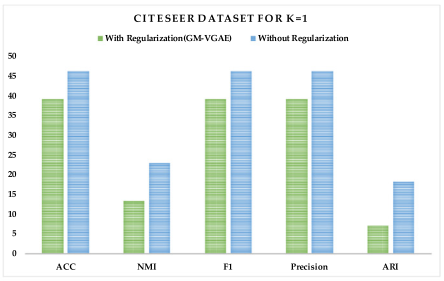
Dataset	Metric	K = 1	K = 2	K = 3
Cora	**ACC**	65.72	73.5	**74.8**
	NMI	48.27	55.3	**59.7**
	F1	65.72	73.9	**73.9**
	Precision	65.72	73.5	**74.8**
	ARI	41.82	52.2	**52.2**
Citeseer	**ACC**	46.20	59.5	**69.5**
	NMI	22.93	33.2	**43.2**
	F1	46.20	59.1	**69.1**
	Precision	46.20	59.1	**59.1**
	ARI	18.26	32.5	**45.5**
Pubmed	**ACC**	68.83	72.6	**74.8**
	NMI	32.0	34.5	**37.5**
	F1	65.72	72.2	**74**
	Precision	68.83	73.2	**74.2**
	ARI	30.91	35.9	**36.9**

Table 8
Effect of parameter k on the result of Link prediction.

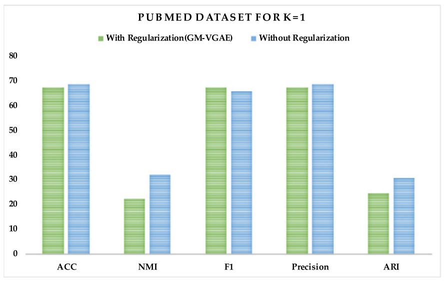
Dataset	Metric	K = 1	K = 2	K = 3
Cora				
	AUC	93.61	98.5	**99**
	AP	93.17	98.22	**98.7**
Citeseer				
	AUC	93.82	98.75	**99.2**
	AP	93.10	98.39	**99.1**
Pubmed				
	AUC	95.38	98.2	**98.7**
	AP	95.72	98.0	**98.5**



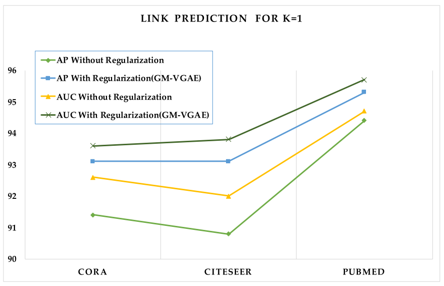
(a) Cora



(b) CiteSeer

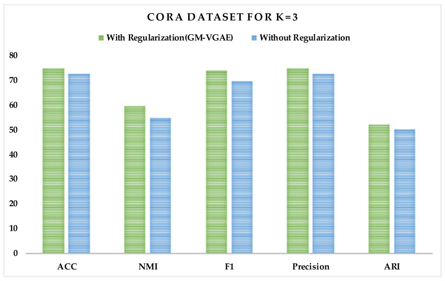


(c) PubMed

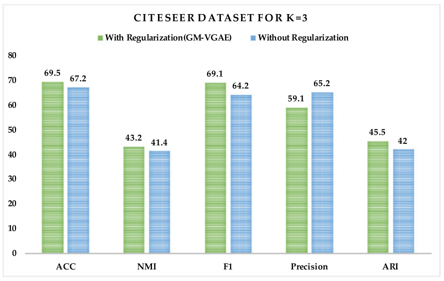


(d) Link Prediction

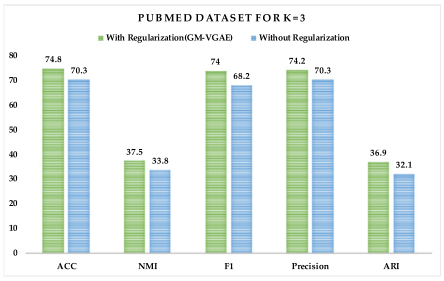
Fig. 8. Impact of the regularization on Datasets for $k = 1$.



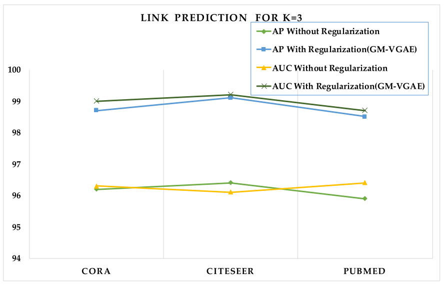
(a) Cora



(b) CiteSeer



(c) PubMed



(d) Link Prediction

Fig. 9. Impact of the regularization on Datasets for $k = 3$.

6.8. Qualitative Analysis

t-SNE is a variant of Stochastic Neighbor Embedding that preserves the data's local structure while revealing some important global structures. It visualizes high-dimensional data in a two- or three-dimensional space. Fig. 6 shows a variety of regular t-SNE plots [61] of the representations learned by the GM-VGAE algorithm on the Cora, CiteSeer, and Pubmed datasets to help understand the efficacy of GM-VGAE. Colors indicate the document's type. Compared to the raw features, the trained representations in 2D space show noticeable clustering proportional to the number of topic classes in each dataset. For better comparison, Fig. 7 shows the plots of some of the baseline methods in the Cora dataset. We plotted the representations learned by DW from shallow embedding approaches, DGI from GCN-based approaches, ARGA and RWR-GAE from auto-encoder-based approaches, and CGCN from deep generative-based approaches.

6.9. Ablation Study

Tables 7 and 8 show the results of comparing the values 1, 2, and 3 for the hyperparameter k in clustering and link prediction tasks. As can be noticed, the best value of k for all datasets is 3. However, the best value of k can be varied depending on the new diverse datasets. The first column of tables shows the affected results of the GMM method. As demonstrated, at $k = 2$ with applying the GMM, remarkable progress in the results is achieved which regarding clustering results on the Cora dataset (Table 7), GM-VGAE with $k = 2$ increases accuracy by 7.78%, NMI by 7.03%, F1 by 8.18%, precision by 7.785 %, and ARI by 10.68% against GM-VGAE with $k = 1$. This boost indicates the validity of our claim that using GMM positively affects results.

To evaluate the effectiveness of adversarial regularization, the proposed method is investigated in two modes: with and without regularization. Fig. 8 shows the effectiveness of the adversarial regularization without considering GMM to further emphasize the impact of adversarial regularization. The analysis illustrates the definite effect of regularization on the results of both the clustering and link prediction tasks on all datasets. Fig. 9 highlights the potency of the adversarial regularization for $k = 3$ to verify the effect of regularization on improving the results of both clustering and link prediction tasks.

7. Conclusion

In this paper, we proposed a variant of the VGAE. In this proposed framework, we assume a GMM to model the prior distribution instead of adopting a Gaussian distribution in a regular VGAE. This assumption intends to perform a specific task. Here, this particular task is unsupervised clustering. Moreover, inspired by the adversarial mechanism concept, a regularization was introduced. Eventually, we examined the performance of the proposed method based on clustering and link prediction tasks on six open graph datasets, Cora, Citseer, and PubMed, which are well-known citation networks, and Cornell, Wisconsin, and Texas, which are webpage datasets. Our proposed method demonstrated remarkable performance on these tasks compared to state-of-the-art algorithms.

Code Availability

A reference GM-VGAE implementation can be found at: https://github.com/SoheilaMolaei/GM-VGAE.

Data availability

Data will be made available on request.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was supported by the Wellcome Trust under grant 217650/Z/19/Z. DAC was supported by the NIHR Oxford Biomedical Research Centre, the InnoHK Hong Kong Centre for Cerebro-cardiovascular Health Engineering (COCHE), and the Pandemic Sciences Institute at the University of Oxford.

References

[1] R. Angles, C. Gutierrez, Survey of graph database models, ACM Computing Surveys (CSUR) 40 (1) (2008) 1–39.
[2] A. Fout, J. Byrd, B. Shariat, A. Ben-Hur, Protein interface prediction using graph convolutional networks, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6533–6542.
[3] A. Sanchez-Gonzalez, N. Heess, J.T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, P. Battaglia, Graph networks as learnable physics engines for inference and control, in: International Conference on Machine Learning, PMLR, 2018, pp. 4470–4479.
[4] H. Hamedmoghadam, M. Jalili, H.L. Vu, L. Stone, Percolation of heterogeneous flows uncovers the bottlenecks of infrastructure networks, Nature communications 12 (1) (2021) 1–10.
[5] S. Molaei, H. Zare, H. Veisi, Deep learning approach on information diffusion in heterogeneous networks, Knowledge-Based Systems 189 (2020) 105–153.
[6] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 2609–2615.
[7] F. Hu, Y. Zhu, S. Wu, W. Huang, L. Wang, T. Tan, Graphair: Graph representation learning with neighborhood aggregation and interaction, Pattern Recognition 112 (2021) .
[8] Z. Hou, Y. Cen, Y. Dong, J. Zhang, J. Tang, Automated unsupervised graph representation learning, IEEE Transactions on Knowledge & Data Engineering (01) (2021) 1–13.
[9] S. Molaei, N.G. Bousejin, H. Zare, M. Jalili, Deep node clustering based on mutual information maximization, Neurocomputing 455 (2021) 274–282.
[10] S. Molaei, N.G. Bousejin, H. Zare, M. Jalili, S. Pan, Learning graph representations with maximal cliques, IEEE Transactions on Neural Networks and Learning Systems (2021) 1–8.
[11] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.
[12] W.L. Hamilton, Graph representation learning, Synthesis Lectures on Artifical Intelligence and Machine Learning 14 (3) (2020) 1–159.
[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems 32 (1) (2020) 4–24.
[14] W.L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, IEEE Data(base) (2017) 1–24.
[15] D. Charte, F. Charte, M.J. del Jesus, F. Herrera, An analysis on the use of autoencoders for representation learning: Fundamentals, learning task case studies, explainability and challenges, Neurocomputing 404 (2020) 93–107.
[16] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, R. Zemel, Neural relational inference for interacting systems, in: International Conference on Machine Learning, PMLR, 2018, pp. 2688–2697.
[17] D.P. Kingma, M. Welling, Auto-encoding variational bayes, ICLR (2014) 14–16.
[18] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, A. Lerchner, Early visual concept learning with unsupervised deep learning., CoRR abs/1606.05579 (2016) 1–12.
[19] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, ICLR (2017) 1–14.
[20] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, in: 2nd International Conference on Learning Representations, ICLR 2014, 2014, pp. 1–14.
[21] P.V. Tran, Multi-task graph autoencoders, NIPS (2018) 1–5.
[22] G. Salha, R. Hennequin, J.-B. Remy, M. Moussallam, M. Vazirgiannis, Fastgae: Scalable graph autoencoders with stochastic subgraph decoding, Neural Networks 142 (2021) 1–19.
[23] J. Park, M. Lee, H.J. Chang, K. Lee, J.Y. Choi, Symmetric graph convolutional autoencoder for unsupervised graph representation learning, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE Computer Society, 2019, pp. 6518–6527.
[24] G. Salha, S. Limnios, R. Hennequin, V.-A. Tran, M. Vazirgiannis, Gravity-inspired graph autoencoders for directed link prediction, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 589–598.
[25] A. Salehi, H. Davulcu, Graph attention auto-encoders, in: 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2020, pp. 989–996.
[26] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, B. Wang, One2multi graph autoencoder for multi-view graph clustering, Proceedings of The Web Conference 2020 (2020) 3070–3076.
[27] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: International conference on machine learning, PMLR, 2014, pp. 1278–1286.
[28] T.N. Kipf, M. Welling, Variational graph auto-encoders, NIPS Workshop on Bayesian Deep Learning (2016) 1–12.
[29] Q. Xie, J. Huang, P. Du, M. Peng, Graph relational topic model with higher-order graph attention auto-encoders, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, 2021, pp. 2604–2613.
[30] T.S. Hy, R. Kondor, Multiresolution graph variational autoencoder, arXiv preprint arXiv:2106.00967 (2021) 1–15.
[31] D. Jin, B. Li, P. Jiao, D. He, W. Zhang, Network-specific variational auto-encoder for embedding in attribute networks, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 2663–2669.
[32] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, ICLR (2016) 1–16.
[33] P.-Y. Huang, R. Frederking, et al., Rwr-gae: Random walk regularization for graph auto encoders, arXiv preprint arXiv:1908.04003 (2019) 1–7.
[34] Q. Lu, N. de Silva, D. Dou, T.H. Nguyen, P. Sen, B. Reinwald, Y. Li, Exploiting node content for multiview graph convolutional network and adversarial regularization, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 545–555.
[35] T.R. Davidson, L. Falorsi, N. De Cao, T. Kipf, J.M. Tomczak, Hyperspherical variational auto-encoders, in: 34th Conference on Uncertainty in Artificial Intelligence 2018, Association For Uncertainty in Artificial Intelligence (AUAI), 2018, pp. 856–865.
[36] S. Zheng, Z. Zhu, X. Zhang, Z. Liu, J. Cheng, Y. Zhao, Distribution-induced bidirectional generative adversarial network for graph representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7224–7233.
[37] B. Hui, P. Zhu, Q. Hu, Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 4215–4222.
[38] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 3844–3852.
[39] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018, pp. 1–18.
[40] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.
[41] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of machine learning research 12 (7) (2011) 2121–2159.
[42] J. Duchi, Y. Singer, Efficient online and batch learning using forward backward splitting, Journal of Machine Learning Research 10 (2009) 2899–2934.
[43] D.A. Reynolds, Gaussian mixture models, Encyclopedia of biometrics 741 (2009) 659–663.
[44] N. Dilokthanakul, P.A. Mediano, M. Garnelo, M.C. Lee, H. Salimbeni, K. Arulkumaran, M. Shanahan, Deep unsupervised clustering with gaussian mixture variational autoencoders, ICLR (2017) 1–12.
[45] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (3) (2008), 93–93.
[46] G. Namata, B. London, L. Getoor, B. Huang, U. EDU, Query-driven active surveying for collective classification, in: 10th International Workshop on Mining and Learning with Graphs, Vol. 8, 2012, pp. 1–8.
[47] H. Pei, B. Wei, K.C.-C. Chang, Y. Lei, B. Yang, Geom-gcn: Geometric graph convolutional networks, ICLR (2020) 1–10.
[48] L. Tang, H. Liu, Leveraging social media networks for classification, Data Mining and Knowledge Discovery 23 (3) (2011) 447–478.
[49] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.
[50] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Advances in Neural Information Processing Systems 31 (2018) 5165–5175.
[51] P. Velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, ICLR (Poster) 2 (3) (2019) 1–17.
[52] C. Mavromatis, G. Karypis, Graph infoclust: Maximizing coarse-grain mutual information in graphs, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2021, pp. 541–553.
[53] J. Park, J. Cho, H.J. Chang, J.Y. Choi, Unsupervised hyperbolic representation learning via message passing auto-encoders, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2021, pp. 5512–5522.
[54] R. Xia, Y. Pan, L. Du, J. Yin, Robust multi-view spectral clustering via low-rank and sparse decomposition, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014, pp. 2149–2155.
[55] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[56] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR (Poster), 2015, pp. 1–15.
[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[58] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.
[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, The Journal of machine Learning research 12 (2011) 2825–2830.
[60] C. Mavromatis, G. Karypis, Graph infoclust: Maximizing coarse-grain mutual information in graphs, in: PAKDD, 2021.
[61] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (Nov) (2008) 2579–2605.

Ghazaleh Niknam Shirvan is a Ph.D. candidate in the information technology group at the University of Tehran, Tehran, Iran. She completed her Master of Science in information technology at the University of Tehran in 2017. Her research interest lies in graph mining, graph representation learning, and its applications. Contact her at gh.niknam@ut.ac.ir.

Soheila Molaei received her Ph.D. in Information Technology from the University of Tehran, Tehran, Iran. Her research interest centers around graph-structured data, representation learning, and their applications in the mining of large social networks. She is a Postdoctoral Research Assistant in Healthcare with CHI Lab at University of Oxford. Contact her at soheila.molaei@eng.ox.ac.uk.

Hadi Zare received the PhD degree from the department of Computer science, Amirkabir University of Technology, Tehran, Iran in 2012. He is currently an Associate Professor with the Faculty of New Sciences and Technologies in the University of Tehran, Tehran, Iran. His current main research interests are in probabilistic graphical models, dimension reduction techniques, discovering and summarizing big data structures such as social networks.

David A. Clifton is a Professor of Clinical Machine Learning in the Department of Engineering Science of the University of Oxford and an OCC Fellow in AI & Machine Learning at Reuben College, Oxford. He is a Fellow of the Alan Turing Institute, Research Fellow of the Royal Academy of Engineering, Visiting Chair in AI for Healthcare at the University of Manchester, and a Fellow of Fudan University, China. He studied Information Engineering at Oxford's Department of Engineering Science. His research focuses on the development of machine learning for tracking the health of complex systems. Since 2008, he has focused mostly on the development of AI-based methods for healthcare. Contact him at david.clifton@eng.ox.ac.uk

Shirui Pan received a Ph.D. in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia. He is currently a Professor and an ARC Future Fellow with the School of Information and Communication Technology, Griffith University, Australia. Before joining Griffith in 2022, he was with the Faculty of Information Technology, Monash University. His research interests include data mining and machine learning. To date, Dr Pan has published over 80 research papers in top-tier journals and conferences, including the IEEE Transactions on Neural Networks and Learning Systems (TNNLS), IEEE Transactions on Knowledge and Data Engineering (TKDE), IEEE Transactions on Cybernetics (TCYB), KDD, AAAI, and CVPR. Contact him at s.pan@griffith.edu.au.