Interpretable Node Representation with Attribute Decoding

Anonymous authors
Paper under double-blind review

Abstract

Variational Graph Autoencoders (VGAEs) are powerful models for unsupervised learning of node representations from graph data. In this work, we make a systematic analysis of modeling node attributes in VGAEs and show that attribute decoding is important for node representation learning. We further propose a new learning model, interpretable NOde Representation with Attribute Decoding (NORAD). The model encodes node representations in an interpretable approach: node representations capture community structures in the graph and the relationship between communities and node attributes. We further propose a rectifying procedure to refine node representations of isolated notes, which improves the quality of the representations of these nodes. Our empirical results demonstrate the advantage of the proposed model when learning graph data in an interpretable approach.

1 Introduction

Graph data are ubiquitous in real-world applications. Graph data contain rich information about graph nodes. The community structure among graph nodes is particularly interesting. Such structures are modeled by traditional models such as the stochastic blockmodel (SBM) (Wang & Wong, 1987; Snijders & Nowicki, 1997) and its variants, which assign a node to one or multiple communities.

Learning node representations are widely investigated in document networks. One typical branch is Relational Topic Model (RTM) (Chang & Blei, 2009) and its variants (Panwar et al., 2021; Bai et al., 2018). In such models, the links and node attributes are decoded from the node representations, which are learned from the documents with rich textual information. RTM-based models often provide interpretable representations due to their carefully-designed graphical models. However, the effectiveness of the RTM-based models highly relies on the text richness of the dataset.

On par with the RTM-based models, Variational Graph Autoencoder (VGAE) (Kipf & Welling, 2016) utilizes Graph Neural Networks (GNNs) to learn node vectors that encode information that can be used for reconstructing the graph structure. This model is further improved in multiple aspects. Mehta et al. (2019); Li et al. (2020); Sarkar et al. (2020) adopt different priors to learn node representations for different applications. Hasanzadeh et al. (2019) introduces a hierarchical variational framework to VGAE and modifies the link decoder. Pan et al. (2018; 2019) adopt the training scheme of generative adversarial networks (GANs) (Goodfellow et al., 2014) to encode the node latent representations. Following VGAE, most of its variants do not decode node attributes and only use node attributes in the encoder. Though some other models (Mehta et al., 2019; Cheng et al., 2021) use separate decoder heads to generate graph edges and node attributes. There is no systematic approach to analyzing how node attributes should be used to better learn representations in the VGAE framework.

In this work, we analyze VGAE using the information-theoretic framework by Alemi et al. (2018) and show the theoretical strengths of different model constructions. In particular, the analysis indicates that appropriate modeling of node attributes brings benefits to a large class of link prediction tasks.

We further devise a new representation learning model, interpretable Node Representations with Attribute Decoding (NORAD), which learns interpretable node representations from the graph data. To exploit node attributes, the model includes a specially designed decoder for node attributes; the model can learn

good representations for nodes with low degrees. We further propose a rectification procedure to refine representations of isolated nodes.

We conduct extensive experiments to evaluate and diagnose our model. The results show that node representations learned by our model perform well in link prediction and node clustering tasks, indicating the good quality of these representations. We also show that the learned node representations capture community structures in the graph and the relationship between communities and node attributes.

Our contributions can be summarized as following:

- we systematically examine VGAE through an information-theoretic analysis;
- we propose a new model NORAD, which includes a specially designed attribute decoder and a refinement procedure for representations of isolated nodes;
- we conduct extensive experiments to study the quality and interpretability of node representations learned by NORAD.

2 Preliminaries

Let $G = (\mathbf{A}, \mathbf{X})$ denote an attributed graph with n nodes, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the binary adjacency matrix of the graph, and $\mathbf{X} = (\mathbf{x}_i)_{i=1}^n \in \mathbb{R}^{n \times D}$ denotes node attributes, with $\mathbf{x}_i$ being the attribute vector of node i. We consider the problem of jointly modeling $\mathbf{A}$ and $\mathbf{X}$. The goal is to learn interpretable node representations $\mathbf{Z} = (\mathbf{z}_i)_{i=1}^n \in \mathbb{R}^{n \times K}$ that can best explain the data. Then $\mathbf{Z}$ provides essential information for downstream tasks such as node clustering.

Graph Neural Networks. GNN is a neural network designed to extract information from graph data. It typically consists of multiple layers, each of which runs a message-passing procedure to encode information into a node's vector representation. Let $\mathbf{H} = \mathrm{gnn}(\mathbf{A}, \mathbf{X}; \phi)$ denote the network function of a L-layer GNN, which is typically defined by $\mathbf{H}^{(0)} = \mathbf{X}$,

$$\mathbf{H}^{(l)} = \sigma\left(\mathbf{H}^{(l-1)}\mathbf{W}^{(l)} + \mathbf{A}\mathbf{H}^{(l-1)}\mathbf{V}^{(l)}\right), \quad l = 1, \ldots, L.$$

And $\mathbf{H} = \mathbf{H}^{(L)}$. Here $\sigma(\cdot)$ is the activation function. Here $\mathbf{W}^{(l)}$ and $\mathbf{V}^{(l)}$ are the network weights for the l-th layer. We denote the all network weights with ϕ.

Variational Graph Autoencoder. VGAE (Kipf & Welling, 2016) learns node representations in an unsupervised approach based on variational auto-encoder (VAE) (Kingma & Welling, 2013). In VGAE, the prior distribution $p(\mathbf{Z})$ over node presentations $\mathbf{Z}$ is a standard Gaussian distribution. And the generative model $p(\mathbf{A}|\mathbf{Z})$ is defined as

$$p(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j), \quad p(A_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \mathrm{sigmoid}(\mathbf{z}_i^\top \mathbf{z}_j). \tag{1}$$

The inference model of VGAE $q(\mathbf{Z}|\mathbf{A}, \mathbf{X})$ is a Gaussian distribution $\mathrm{Gaussian}(\mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ defined by a GNN.

$$(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathrm{gnn}(\mathbf{A}, \mathbf{X}). \tag{2}$$

The output of the GNN is split into $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, representing the mean and standard derivation, respectively. VGAE maximizes the Evidence Lower Bound (ELBO) of the marginal log-likelihood $\log p(\mathbf{A})$ to learn the encoder and the decoder:

$$\mathcal{L}_g = \mathbb{E}_{q(\mathbf{Z}|\mathbf{A},\mathbf{X})}\big[\log p(\mathbf{A}|\mathbf{Z}) + \log p(\mathbf{Z}) - \log q(\mathbf{Z}|\mathbf{A}, \mathbf{X})\big]. \tag{3}$$

After the encoder is learned, its mean $\boldsymbol{\mu}$ can be used as deterministic node representations.

3 An Information-Theoretic Analysis of Attribute Decoding

In this section, we analyze VGAE from the perspective with the rate-distortion theory (Alemi et al., 2018) and consider the mutual information between the encoding $\mathbf{Z}$ and observed data $(\mathbf{A}, \mathbf{X})$. Let $p_*(\mathbf{A}, \mathbf{X})$ be the data distribution and H be its entropy, which is a constant. Let $I_q = I[(\mathbf{A}, \mathbf{X}); \mathbf{Z}]$ denote the mutual information between $(\mathbf{A}, \mathbf{X})$ and $\mathbf{Z}$. Note that I_q is defined from the encoder distribution $q(\mathbf{Z}|\mathbf{A}, \mathbf{X})$ and the data distribution. The maximization of the ELBO

$$\mathcal{L}_e = \mathbb{E}_{q(\mathbf{Z}|\mathbf{A},\mathbf{X})}\big[\log p(\mathbf{A}, \mathbf{X}|\mathbf{Z}) + \log p(\mathbf{Z}) - \log q(\mathbf{Z}|\mathbf{A}, \mathbf{X})\big] \tag{4}$$

can be viewed as indirect maximization the mutual information $I[(\mathbf{A}, \mathbf{X}); \mathbf{Z}]$ under a rate constraint (Alemi et al., 2018). We further decompose $I[(\mathbf{A}, \mathbf{X}); \mathbf{Z}]$ as follows:

$$I[(\mathbf{A}, \mathbf{X}); \mathbf{Z}] = I[\mathbf{A}; \mathbf{Z}] + I[\mathbf{X}; \mathbf{Z}] + I[\mathbf{A}; \mathbf{X}|\mathbf{Z}] - I[\mathbf{A}; \mathbf{X}] \tag{5}$$

In this decomposition, the last term $I[\mathbf{A}; \mathbf{X}]$ is a constant decided by the data. The first term is the information about $\mathbf{A}$ from $\mathbf{Z}$, and the second term is the information about $\mathbf{X}$. The third term $I[\mathbf{A}; \mathbf{X}|\mathbf{Z}]$ is the information between $\mathbf{A}$ and $\mathbf{X}$ that cannot be explained by $\mathbf{Z}$. When $\mathbf{Z}$ is lossless encoding, $I[\mathbf{A}; \mathbf{X}|\mathbf{Z}] = 0$.

The decomposition above is derived from the encoder distribution $q(\mathbf{Z}|\mathbf{A}, \mathbf{X})$, but it helps us to design the decoder $p(\mathbf{A}, \mathbf{X}|\mathbf{Z})$, which approximates $q(\mathbf{A}, \mathbf{X}|\mathbf{Z})$ when maximizing the ELBO $\mathcal{L}_e$. One variant of VGAE (Mehta et al., 2019; Cheng et al., 2021) assumes $p(\mathbf{A}, \mathbf{X}|\mathbf{Z}) = p(\mathbf{X}|\mathbf{Z})p(\mathbf{A}|\mathbf{Z})$. In this case, the conditional mutual information $I[\mathbf{A}; \mathbf{X}|\mathbf{Z}]$ tends to be zero. The conditional independence assumption may require $\mathbf{Z}$ to have extra bits, but $\mathbf{Z}$ can explain $\mathbf{A}$ and $\mathbf{X}$ separately. In this model choice, the lower bound becomes

$$\mathcal{L}_a = \mathbb{E}_{q(\mathbf{Z}|\mathbf{A},\mathbf{X})}\big[\log p(\mathbf{A}|\mathbf{Z}) + \log p(\mathbf{X}|\mathbf{Z}) + \log p(\mathbf{Z}) - \log q(\mathbf{Z}|\mathbf{A}, \mathbf{X})\big]. \tag{6}$$

Other possible variants that do not have the assumption, the decoder $p(\mathbf{A}, \mathbf{X}|\mathbf{Z})$ can be decomposed as $p(\mathbf{X}|\mathbf{Z})p(\mathbf{A}|\mathbf{X}, \mathbf{Z})$, then it is more flexible and should fit the data better. However, $\mathbf{Z}$ is less capable in explaining $\mathbf{A}$ because part of the information is encoded in $\mathbf{X}$. The same issue happens in the decomposition $p(\mathbf{A}, \mathbf{X}|\mathbf{Z}) = p(\mathbf{A}|\mathbf{Z})p(\mathbf{X}|\mathbf{A}, \mathbf{Z})$.

Another variant Graphite (Grover et al., 2019) takes a totally different approach: it only lets $\mathbf{Z}$ encode $\mathbf{A}$ and conditions the entire model on $\mathbf{X}$. Therefore, the ELBO is

$$\mathcal{L}_c = \mathbb{E}_{q(\mathbf{Z}|\mathbf{A},\mathbf{X})}\big[\log p(\mathbf{A}|\mathbf{X}, \mathbf{Z}) + \log p(\mathbf{Z}|\mathbf{X}) - \log q(\mathbf{Z}|\mathbf{A}, \mathbf{X})\big] \leq I[\mathbf{A}; \mathbf{Z}|\mathbf{X}]. \tag{7}$$

In this case, the optimal encoding $\mathbf{Z}$ only encodes the part of $\mathbf{A}$ that cannot be explained by $\mathbf{X}$. This results in $\mathbf{Z}$ only containing a small portion of the information about $\mathbf{A}$.

In the ELBO $\mathcal{L}_g$ of the basic VGAE, there is no decoding of $\mathbf{X}$. This means the encoder has no incentive to encode any information about $\mathbf{X}$. By decoding $\mathbf{X}$, the learned representation $\mathbf{Z}$ is usually more expressive.

Our analysis shows that the simple formulation $\mathcal{L}_a$ with conditional independence assumption is a better choice if we want $\mathbf{Z}$ to maximally encode information about $(\mathbf{A}, \mathbf{X})$. Though we are not the first to discover this formulation, our analysis of different formulations helps deepen the understanding of VGAE variants.

Fix the bias of model fitting in link prediction. Node representations of VGAE are often used in link prediction tasks. Therefore, the given adjacency matrix is $\hat{\mathbf{A}}$, which is sparser than the true adjacency matrix $\mathbf{A}$. In this case, we have to do model fitting with $\hat{\mathbf{A}}$ instead of $\mathbf{A}$.

Model training based on $\hat{\mathbf{A}}$ is biased because actual edges missing from $\hat{\mathbf{A}}$ are mixed with non-edges. This issue has been studied by Liang et al. (2016); Liu & Blei (2017) in different contexts. Specifically, the bias is due to the term $E_q\big[\log p(\hat{\mathbf{A}}|\mathbf{Z})\big]$, which leads $\mathbf{Z}$ to treat missing edges and non-edges equally. However, in some datasets, $\mathbf{X}$ and $\mathbf{A}$ have a strong correlation, and $\mathbf{X}$ is observed for all graph nodes. In this case, encoding information in $\mathbf{X}$ helps improve the link prediction performance, so we consider a modified ELBO:

$$\mathcal{L}_\alpha = \mathbb{E}_{q(\mathbf{Z}|\mathbf{A},\mathbf{X})}\big[\log p(\hat{\mathbf{A}}|\mathbf{Z}) + \alpha \log p(\mathbf{X}|\mathbf{Z}) + \log p(\mathbf{Z}) - \log q(\mathbf{Z}|\mathbf{A}, \mathbf{X})\big]. \tag{8}$$

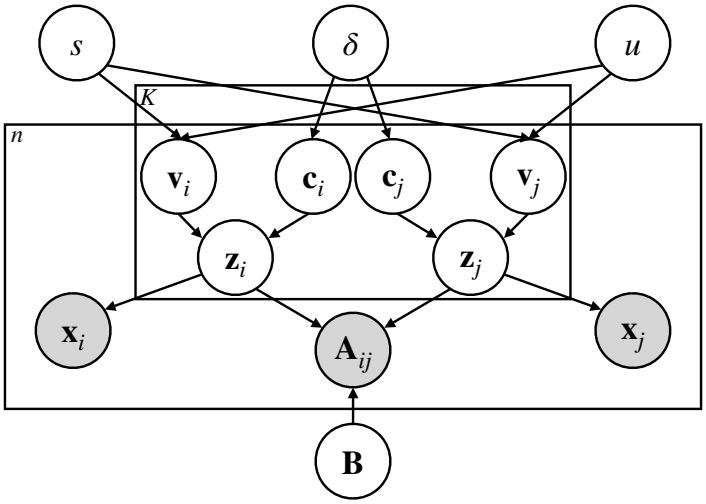


Figure 1: The plate representation of our framework. Here the parameters $(\boldsymbol{\eta}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ are computed from the encoder.

In a link prediction task, we set the ratio factor $\alpha > 1$ so that model training can pay more attention to node features. Note that this modified ELBO is still a lower bound of $I_q - H$ when $\mathbf{X}$ is discrete and $\log p(\mathbf{X}|\mathbf{Z}) < 0$. If we set $\alpha = 0$, it is equivalent to VGAE. If $\hat{\mathbf{A}} = \mathbf{A}$ in a representation learning task, we still use the normal ELBO and set $\alpha = 1$.

4 Interpretable Node Representations with Attribute Decoding

In this section, we introduce a new model, NORAD, to learn node representations. The proposed model has two goals. The primary goal is to learn high-quality node representations that best preserve the information of the attributed graph, and the secondary goal is to make the learned representations interpretable.

We will use the formulation $\mathcal{L}_\alpha$ in Equation (8) to construct our model. We need to specify four distributions: the encoder $q(\mathbf{Z}|\mathbf{A}, \mathbf{X})$, the prior $p(\mathbf{Z})$, the decoder $p(\mathbf{A}|\mathbf{Z})$, and the decoder $p(\mathbf{X}|\mathbf{Z})$. We illustrate the graphical model of our framework in Figure 1.

4.1 The prior distribution

We assign a prior such that the representation $\mathbf{z}_i$ of node i is a sparse vector. We use a prior that separately decides the sparse pattern and non-zero values in $\mathbf{z}_i$. Let $\mathbf{C} = (\mathbf{c}_i)_{i=1}^n$ be the random binary vectors indicating the sparse pattern and $\mathbf{V} = (\mathbf{v}_i)_{i=1}^n$ be the random real value vectors deciding non-zero entries of $\mathbf{Z}$. Both of them have the same dimension as $\mathbf{Z}$. Then

$$\mathbf{c}_{ik} \sim \text{Bernoulli}(\delta), \quad \mathbf{v}_{ik} \sim \text{Gaussian}(u, s), \quad \text{where } i = 1, \ldots, n;\ k = 1, \ldots, K \tag{9}$$

Here $\delta = 0.5$, $u = 0$, $s = 1$, and $\mathbf{Z} = \mathbf{C} \odot \mathbf{V}$ with $\odot$ being the Hadamard product. The prior $p(\mathbf{Z})$ in our framework is equivalent to $p(\mathbf{C}, \mathbf{V})$. We can view $\mathbf{z}_i$ as i-th node's membership assignment in K communities. Each entry of $\mathbf{z}_i$ is from a spike and slab distribution: the spike indicates whether the i-th node belongs to the corresponding community while the slab indicates the (positive or negative) strength of i-th node's membership in the community.

4.2 The conditional distribution of the adjacency

We then define the generative process of graph edges. We use the OSBM (Latouche et al., 2011) to define the decoder for $\mathbf{A}$. In OSBM, two nodes' community memberships indicate how they interact. We use parameter $\mathbf{B} \in \mathbb{R}^{K \times K}$ to indicate the interactions between different communities. A large value in the entry $B_{kk'}$ means a node in community k has a higher chance of connecting with a node in community k', and vice versa. Formally, we define the graph edge distribution as follows:

$$p_{\mathbf{B}}(A_{ij} = 1 | \mathbf{z}_i, \mathbf{z}_j) = \text{sigmoid}(\mathbf{z}_i^\top \mathbf{B} \mathbf{z}_j), \quad p_{\mathbf{B}}(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^{n} \prod_{j=1}^{n} p_{\mathbf{B}}(A_{ij}|\mathbf{z}_i, \mathbf{z}_j). \tag{10}$$

By using the OSBM distribution, the model is encouraged to learn meaningful community structures in the graph.

4.3 The conditional distribution of attributes

We also define an interpretable decoder for node attributes. Here we first focus on binary attributes: $\mathbf{x}_i \in \{0, 1\}^D$ is a binary vector.

We assume that each $\mathbf{z}_i$ independently generates $\mathbf{x}_i$: $p(\mathbf{X}|\mathbf{Z}) = \prod_{i \in V} p(\mathbf{x}_i|\mathbf{z}_i)$. Then we design an Attention-based Topic Network (ATN) to consider the community-attribute relation in $p(\mathbf{x}_i|\mathbf{z}_i)$. We use θ to denote the network parameters in ATN.

Similar to topic models, ATN assumes that each community is represented as an embedding vector. All such embedding vectors are in a matrix $\mathbf{T} \in \mathbb{R}^{K \times d'}$. Each attribute also has an embedding vector, and all attributes' vectors are denoted in a matrix $\mathbf{U} \in \mathbb{R}^{d' \times D}$. We compute attention weights from the two embedding matrices to decide the probabilities of the node's attributes from the two embedding matrices. We first compute the aggregated community vector

$$\mathbf{g}_i = \text{relu}(\mathbf{T}^\top \mathbf{z}_i). \tag{11}$$

Then we use the vector $\mathbf{g}_i$ as a query to compute attention weights against attribute vectors $\mathbf{U}$.

$$\boldsymbol{\lambda}_i = \text{sigmoid}\left(\frac{\mathbf{g}_i^\top \mathbf{W}_q \mathbf{W}_k^\top \mathbf{U}}{\sqrt{d''}} \right). \tag{12}$$

Here the parameters $\mathbf{W}_q$ and $\mathbf{W}_k$ both have size $(d' \times d'')$. We apply the sigmoid function, instead of softmax in standard attention calculation, to get binary probabilities $\boldsymbol{\lambda}_i$.

We denote the procedure above as $\boldsymbol{\lambda}_i = \text{atn}(\mathbf{z}_i; \mathbf{T}, \mathbf{U})$, then we have

$$p(\mathbf{x}_i|\mathbf{z}_i) = \prod_{d=1}^{D} p(\mathbf{x}_{id}|\mathbf{z}_i), \quad p(\mathbf{x}_{id} = 1|\mathbf{z}_i) = \boldsymbol{\lambda}_{id}. \tag{13}$$

The proposed decoder can also be extended for the bag-of-words (BoG) feature. This is achieved by first computing the vector O, whose d-th entry records the maximum occurrence of the d-th word in the vocabulary. Then given a BoG feature x_i, we fit $\boldsymbol{\lambda}_i$ to a vector of Bernoulli parameters x_i/O. We can sample each word O_d times and recover the BoG feature during reconstruction.

4.4 The encoder

Here we introduce our model's encoder, which is designed for two purposes: computing node representations and model fitting through variational inference, which we will discuss right after this subsection.

The encoder $q_\phi(\mathbf{Z}|\mathbf{A}, \mathbf{X})$ is computed by a GNN, whose parameters are collectively denoted by ϕ. Since $\mathbf{Z}$ is computed from $\mathbf{C}$ and $\mathbf{V}$, we use $q_\phi(\mathbf{C}, \mathbf{V}|\mathbf{A}, \mathbf{X})$ instead, this can be further represented as $q_\phi(\mathbf{Z}|\mathbf{A}, \mathbf{X}) =$

Algorithm 1 Varaitional EM for NORAD

Input: network graph $G = (\mathbf{A}, \mathbf{X})$, initialized model and variational parameters (θ, ϕ) and blockmodel $\mathbf{B}$, iteration steps T_e, T_m, ratio factor α, regularization factor γ.
Output: learned model and variational parameters (θ, ϕ) and blockmodel $\mathbf{B}$.
repeat
  [E-step]
  Fix blockmodel $\mathbf{B}$
  for $t = 1, \ \ldots, \ T^e$ **do**
    Compute $(\boldsymbol{\eta}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \text{gnn}(\mathbf{X}, \mathbf{A}; \phi)$.
    Sample $\mathbf{C} \sim \text{Bernoulli}(\boldsymbol{\eta})$, $\mathbf{V} \sim \text{Gaussian}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ via reparameterization tricks.
    Compute Hadamard product $\mathbf{Z} = \mathbf{C} \odot \mathbf{V}$.
    Compute loss $\mathcal{L}(\theta, \phi) = \alpha \log p_\theta(\mathbf{X}|\mathbf{Z}) + \log p_{\mathbf{B}}(\mathbf{A}|\mathbf{Z}) + \log p(\mathbf{C}, \mathbf{V}) - \log q_\phi(\mathbf{C}, \mathbf{V}|\mathbf{A}, \mathbf{X})$.
    Update (ϕ, θ) by maximizing $\mathcal{L}(\phi, \theta)$ via SGD.
  end for
  [M-step]
  Fix parameters (ϕ, θ)
  for $t = 1, \ \ldots, \ T^m$ **do**
    Compute $(\boldsymbol{\eta}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \text{gnn}(\mathbf{X}, \mathbf{A}; \phi)$.
    Compute Hadamard product $\mathbf{Z} = \boldsymbol{\eta} \odot \boldsymbol{\mu}$.
    Compute loss $\mathcal{L}(\mathbf{B}) = \log p_{\mathbf{B}}(\mathbf{A}|\mathbf{Z}) - \gamma \|\mathbf{B}\|$.
    Update $\mathbf{B}$ by maximizing $\mathcal{L}(\mathbf{B})$ via SGD.
  end for
until convergence of the ELBO $\mathcal{L}(\phi, \theta, \mathbf{B})$

$q_\phi(\mathbf{V}|\mathbf{A}, \mathbf{X}) q_\phi(\mathbf{C}|\mathbf{A}, \mathbf{X})$ by using mean-field distribution. In the variational distribution, $\mathbf{V}$ and $\mathbf{C}$ are respectively sampled from Bernoulli and Gaussian distributions, which are parameterized by a GNN.

$$q_\phi(\mathbf{C}|\mathbf{A}, \mathbf{X}) \sim \text{Bernoulli}(\boldsymbol{\eta}), \quad q_\phi(\mathbf{V}|\mathbf{A}, \mathbf{X}) \sim \text{Gaussian}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2). \tag{14}$$

We use a GNN to compute these distribution parameters from $\mathbf{X}$ and $\mathbf{A}$.

$$(\boldsymbol{\eta}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \text{gnn}(\mathbf{A}, \mathbf{X}; \phi). \tag{15}$$

Here $\boldsymbol{\eta}$, $\boldsymbol{\mu}$, and $\boldsymbol{\sigma}$ are all matrices of size $n \times K$. The output of the GNN has size $n \times (3K)$, then it is split into the three matrices. All parameters of the two variational distributions are network parameters of the GNN.

Deterministic node representations $\mathbf{Z}^\circ$ are computed from $\boldsymbol{\eta}$ and $\boldsymbol{\mu}$ directly.

$$\mathbf{Z}^\circ = \boldsymbol{\mu} \odot \mathbf{1}(\boldsymbol{\eta} > 0.5). \tag{16}$$

Here $\mathbf{1}(\boldsymbol{\eta} > 0.5)$ gets a binary matrix indicating which elements are greater than the threshold 0.5.

4.5 Model fitting through variational inference

In this section, we discuss the learning procedure of our model. Besides the ELBO we have discussed in the previous section, we also add regularization terms over the parameter $\mathbf{B}$.

$$\mathcal{L}(\theta, \phi, \mathbf{B}) = \mathrm{E}_{q_\phi(\mathbf{C}, \mathbf{V}|\mathbf{A}, \mathbf{X})} \big[\log p_{\mathbf{B}}(\mathbf{A}|\mathbf{C}, \mathbf{V}) + \alpha \log p_\theta(\mathbf{X}|\mathbf{C}, \mathbf{V}) + \log p(\mathbf{C}, \mathbf{V}) - \log q_\phi(\mathbf{C}, \mathbf{V}|\mathbf{A}, \mathbf{X}) \big] + \gamma \|\mathbf{B}\|. \tag{17}$$

The hyper-parameter α controls the strength of the attribute decoder. We maximize the objective $\mathcal{L}(\theta, \phi, \mathbf{B})$ with respect to model parameters (θ and $\mathbf{B}$) and variational parameters ϕ. The objective gradient is estimated through Monte Carlo samples from the variational distribution. Note that the random variable $\mathbf{C}$ is binary, and we use the Gumbel-softmax trick (Jang et al., 2016), then we can estimate the gradient of the objective efficiently through the reparameterization trick.

We find alternatively updating (θ, ϕ) and $\mathbf{B}$ in a variational-EM fashion gives better optimization performance. In our optimization procedure, we fix $\mathbf{B}$ and update (θ, ϕ) for a few iterations at E-step, and then fix model and variational parameters (θ, ϕ) and optimize $\mathbf{B}$ for a few iterations at M-step. The parameters are all optimized with SGD. The intuition behind this is that the training of $\mathbf{B}$ needs to depend on somewhat clear relations between different communities. The training procedure is shown in Algorithm 1.

4.6 Rectifying representations of isolated nodes

In the training process, the representation of isolated nodes are learned to predict zero connections from these nodes. As we have analyzed, representations learned in this approach is biased. Here we use the rectification strategy to post-process the learned representations of isolated nodes. For an isolated node i, we first compute the deterministic node representation $\mathbf{z}_i^\circ$ from Equation (16) and then update $\mathbf{z}_i^\circ$ through ATN to improve the recovery of $\mathbf{x}_i$. The update rule is shown as following:

$$\mathbf{z}_i^\circ = \mathbf{z}_i^\circ + \epsilon \nabla_{\mathbf{z}_i^\circ} \log p_\theta(\mathbf{x}_i | \mathbf{z}_i^\circ), \tag{18}$$

where ϵ is the update learning rate. We run the update for multiple iterations and obtain the final representation. Empirically, 50 to 100 iterations usually give a clear improvement of these nodes' representations.

5 Experiments

In this section, we study the proposed model with real datasets. The first aim of the study is to examine the quality of node representations: whether the model learns node representations of high quality, and how each component contributes the learning. We examine our model through extensive ablation studies and sensitivity analysis. The second aim is to examine the interpretability of learned node representations. We look into the data and show how learned representations encode interpretable structures in the data.

Datasets. We use four benchmark datasets, including Cora, Citeseer, Pubmed, and DBLP (Morris et al., 2020). The details of datasets can be found in Table 7 in Appendix.

Baselines. We benchmark the performance of NORAD against existing models that share the same traits as our model. The experiment setups and the implementation details of our model are shown in Appendix A.2. We consider five baseline methods. (1) DeepWalk (Perozzi et al., 2014) learns the latent node representations by treating truncated random walks sampled within the network as sentences. (2) VGAE (Kipf & Welling, 2016) is the first variational auto-encoder based on GNN. (3) KernelGCN (Tian et al., 2019) proposes a learnable kernel-based framework, which decouples the kernel function and feature mapping function in the propagation of GCN; (4) DGLFRM (Mehta et al., 2019) adopts IBP and Normal prior when encoding the node representations and adds a perceptron layer to the node representations before decoding the edges. (5) VGNAE (Ahn & Kim, 2021) shows that L2 normalization on node hidden vector in GCN improves the representation quality of isolated nodes.

5.1 Link prediction and node clustering

We consider the quality of node representations in two tasks: link prediction and node clustering. In particular, we pay attention to the representation of isolated nodes in sparse graphs.

In the link prediction task, all models predict the missing edges of the graph based on the training data, including node attributes and links. We evaluate our model and the baselines in terms of Area Under the ROC Curve (AUC) and Average Precision(AP) on four datasets. We follow the data splitting strategy in Kipf & Welling (2016) and report the mean and standard deviation over 10 random data splits. The results are reported in Table 1.

The table shows that NORAD significantly outperforms the baselines on almost all datasets, illustrating the superior capability of our model in encoding graph information. The advantage is more obvious on DBLP, which has relatively rich node attributes and links.

		DeepWalk	VGAE	KernelGCN	DGLFRM	VGNAE	NORAD
Cora	AUC	84.6±0.01	92.6±0.01	93.1±0.06	93.4±0.23	94.9±0.43	**95.6±0.56**
	AP	88.5±0.00	93.3±0.01	93.2±0.07	93.8±0.22	94.9±0.39	**96.1±0.44**
Citeseer	AUC	80.5±0.01	90.8±0.02	90.9±0.08	93.8±0.32	**96.0±0.74**	**95.6±0.28**
	AP	85.0±1.00	92.0±0.02	91.8±0.04	94.4±0.73	**96.1±0.89**	**96.5±0.19**
Pubmed	AUC	84.2±0.02	94.2±0.76	94.5±0.03	94.0±0.08	95.0±0.26	**97.1±0.25**
	AP	87.8±1.00	94.0±0.88	94.2±0.01	95.0±0.35	94.7±0.36	**97.3±0.28**
DBLP	AUC	80.4±0.65	90.8±0.37	93.6±0.22	93.7±0.41	91.8±0.34	**96.3±0.22**
	AP	83.1±0.55	91.4±0.44	93.9±0.18	94.0±0.54	92.6±0.26	**97.0±0.18**

Table 1: Performance comparison of all models in the link prediction task: We used an unpaired t-test to compare models' performance values. Not significantly worse than the best at the 5% significance level are bold.

		DeepWalk	VGAE	KernelGCN	DGLFRM	VGNAE	NORAD
Cora	NMI	40.0±1.26	42.6±2.64	44.2±1.07	48.0±2.02	**51.1±0.84**	**50.3±3.72**
	ACC	56.5±1.71	56.7±4.49	60.9±2.43	63.1±4.11	**67.5±2.29**	**66.5±5.89**
Citeseer	NMI	13.2±1.55	15.5±2.83	25.6±2.56	28.8±1.63	35.6±3.76	**38.9±1.18**
	ACC	38.8±2.12	36.1±2.38	52.8±4.36	51.9±2.38	57.8±4.41	**64.5±1.17**
Pubmed	NMI	**28.5±0.73**	**30.1±2.56**	**28.6±1.27**	25.0±4.21	26.2±0.47	24.5±5.97
	ACC	67.1±0.54	**67.6±2.88**	**68.6±0.82**	65.2±3.64	64.1±0.37	61.1±6.36
DBLP	NMI	19.7±1.76	23.6±2.70	30.5±0.56	30.0±2.66	26.0±3.40	**40.2±4.59**
	ACC	54.8±1.31	47.8±2.81	56.0±2.61	55.8±3.19	52.4±6.66	**64.4±6.09**

Table 2: Performance comparison of all models in the node clustering task: We used an unpaired t-test to compare models' performance values. Not significantly worse than the best at the 5% significance level are bold.

In the node clustering task, we generate the node representations from different learning models and then use K-means to obtain the clustering results. We set the hyperparameter in K-means as the number of classes in each dataset. We compare the clustering performance against node embeddings learned from the baselines listed above. We compare NORAD against the baselines in terms of Normalized Mutual Information (NMI) and Accuracy (ACC) on four datasets. Before calculating ACC, we use the Hungarian matching algorithm (Kuhn, 1955) to match the K-means predicted cluster labels with true labels. We report the mean and standard derivation over 10 runs. The results can be found in Table 2.

We can observe that in the node clustering task, NORAD works better on Cora, Citeseer and DBLP against almost all baselines. For Citeseer and DBLP, NORAD significantly outperforms all four baselines; this can be explained that these two datasets provide more node information or edge information than the other two. We find that our performance on Pubmed is not very competitive as on other datasets. One possible reason is that Pubmed has the least node information among all the four datasets, and node attributes are not informative about node classes.

5.2 Ablation study

In this section, we do ablation studies to understand the benefit of each model's component. Specifically, we investigate different variants of the decoder for $\mathbf{A}$ and $\mathbf{X}$, respectively, and benefits of employing representation rectification for isolated nodes. The results are tabulated in Table 3, implementation details can be found in Appendix A.2.

We compare our edge decoder against two variants. For the first variant, we construct the decoder using $p(\mathbf{A}|\mathbf{Z},\mathbf{X})$ assuming $p(\mathbf{A},\mathbf{X}|\mathbf{Z}) = p(\mathbf{X}|\mathbf{Z})p(\mathbf{A}|\mathbf{X},\mathbf{Z})$. For the second variant, we consider the dot product decoder $p(\mathbf{A}|\mathbf{Z})$ used by the VGAE model. We observe that our edge decoder performs better than the chosen variants. We reason that the first variant strongly relies on the node features, which is again reconstructed

Model Variants	Cora	Citeseer
NORAD	95.6±0.56	95.6±0.28
construction w/ $p(\mathbf{A}\|\mathbf{X},\mathbf{Z})$	94.5±0.42	94.3±0.76
VGAE decoder $p(\mathbf{A}\|\mathbf{Z})$	90.4±1.21	89.9±1.01
w/o decoder $p(\mathbf{X}\|\mathbf{Z})$	94.6±0.58	93.4±0.99
NRTM decoder $p(\mathbf{X}\|\mathbf{Z})$	95.3±0.41	94.0±0.42
TAN decoder $p(\mathbf{X}\|\mathbf{Z})$	95.0±0.69	94.7±0.55
w/o rectification	95.0±0.62	95.0±0.29

Table 3: AUC of link prediction on Cora and Citeseer using different model variants.

from $\mathbf{Z}$ via $p(\mathbf{X}|\mathbf{Z})$. The reconstruction error in $p(\mathbf{X}|\mathbf{Z})$ is then accumulated to $p(\mathbf{A}|\mathbf{X},\mathbf{Z})$. In the second variant, the dot product decoder drastically decrease the link prediction performance. We hypothesize that the node representation $\mathbf{Z}$ need to trade-off between both edge and node reconstructions. By introducing the blockmodel $\mathbf{B}$, the edge reconstruction task for $\mathbf{Z}$ is relaxed, and reducing the performance drop caused by the trade-off. At the same time, a better reconstruction of $\mathbf{X}$ can be obtained with a well-designed decoder $p(\mathbf{X}|\mathbf{Z})$.

We then investigate the power of using different node decoders $p(\mathbf{X}|\mathbf{Z})$. Specifically, we choose decoders proposed by Neural Relational Topic Model (NRTM) (Bai et al., 2018) and Topic attention Networks (TAN) (Panwar et al., 2021). In addition, the effect of the node decoder absence is also considered. We find that our ATN decoder performs better than the decoders in other works. Moreover, we can see that the use of the node decoder significantly contributes to model performance. Interestingly, we also observe the richer the node attributes are, the more the model can benefit from utilizing an node decoder.

Finally, we study how rectifying the representation of the isolated nodes brings additional performance rise to our model. Even though there is only a small portion of the nodes are isolated, we still observe an non-trial improvement on the performance, which indicates the effectiveness of rectification. We give a more thorough analysis of it under the circumstance when graphs are sparse in the next section,

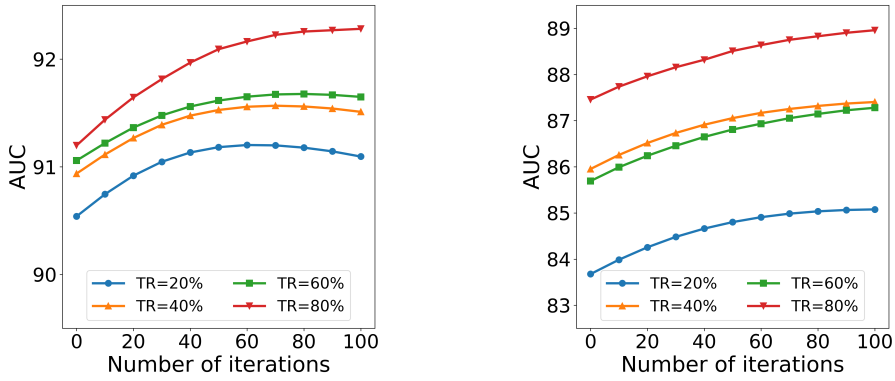
5.3 Attribute decoding for sparse graphs

Our experiments show strong evidence that decoding attribute benefits node representation learning when graphs are sparse. Here we report our observations.

We create sparser graphs by masking more edges during training. Specifically, we only keep 20%, 40%, 60%, and 80% edges in the training set, and for the remaining edges, 1/3 are used for validation, and 2/3 are used for testing. We again choose Cora and Citeseer datasets for analysis. Results are reported in Table 4. This result shows more benefit is gained from the node decoder ATN when graphs are sparser.

We then consider the case of strengthening the node decoder by using α values greater than 1. We run the same experiment as above but varying α values in Equation (17) and then report our observations in Table 5. We see that large α values slightly increase the performance for sparse graphs.

Suggesting links for isolated nodes is often considered as the cold-start problem (Schein et al., 2002), which is important in recommender systems. Here we examine the benefit of rectification procedure proposed in Equation (18) by checking the isolated nodes representations. We consider dataset with differnt sparsity. Specifically, we split graphs with different training ratios (TRs): 20%, 40%, 60%, 80%, lower training ratio indicates more isolated nodes exist. The details of isolated nodes for each dataset can be found in Table 8 in Appendix. The results in Figure 2 show that our rectification procedure greatly improves the quality of isolated nodes' representations. Another interesting observation is that as TR decreases (graph becomes much sparser), the model performance benefits less from the rectification. Our explanation is that the ATN provides less correct guidance to the rectification procedure when the graph is sparser and the ATN itself is not well trained.



(a) Cora (b) Citeseer

Figure 2: Rectifying representations of isolated nodes : for both (a) Cora and (b) Citeseer datasets, we can find that the AUC of isolated nodes related to link prediction is increasing with the rectification iteration on all training ratios (TRs).

Ratio	Cora		Citeseer	
	w/o ATN	w/ ATN	w/o ATN	w/ ATN
20%	82.7±0.61	86.4±0.58	88.5±0.76	92.2±0.55
40%	88.5±0.71	90.7±0.44	91.7±0.51	93.7±0.39
60%	91.9±0.63	93.2±0.37	92.8±0.58	94.5±0.42
80%	94.0±0.75	95.2±0.56	93.9±0.62	95.6±0.48

Table 4: Benefits of using ATN versus different sparsity levels: The percentage(%) indicates the fraction of graph edges in the training set. We report the link prediction performance (AUC).

α	Cora		Citeseer	
	20%	40%	20%	40%
1.0	86.4	90.7	92.2	93.7
1.5	86.5	90.8	92.6	93.9
2.0	86.7	90.9	92.8	94.0
2.5	86.8	91.1	92.9	94.1
3.0	86.9	91.4	93.1	94.3

Table 5: Link prediction performance (AUC) on Cora and Citeseer with different training ratios as α increases.

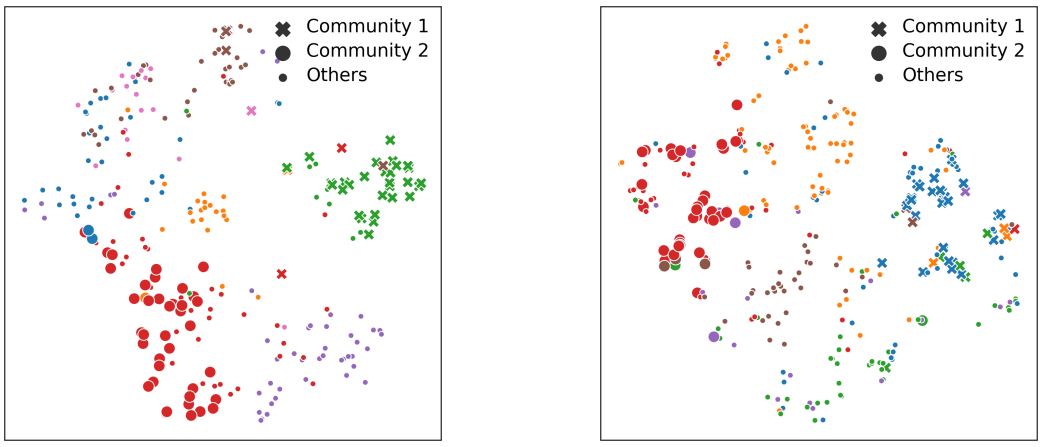
5.4 Interpretability of node representations

We inspect node representations learned by our model qualitatively and interpret them in the context of the application. Specifically, we focus on two questions: whether representations capture community structures in the data and whether representations explain node attributes from the perspective of community structures.

Alignment between community membership and node classes. We visualize a few example communities detected from the graph. The visualization of the detected communities along with the node label is done in the following steps: (1) we first choose one experiment trail for each dataset, and obtain the node representation $\mathbf{Z}$ for t-SNE compressing (Van der Maaten & Hinton, 2008); (2) we take 10% of the nodes and choose 2 representative communities. Note that we perform the dimension reduction in full data, then randomly take a subset of the compressed data; (3) we set the threshold to be 0.5, and compute the community membership for each selected node, there are three kinds of community assignment for each node – community 1, community 2, and others.

The visualization results is shown in Figure 3. We use colors to indicate node classes. Then we extract communities from $\mathbf{Z}$. For community k, nodes with $\mathbf{z}_{ik}$ greater than a threshold are considered to be in the community. We identify nodes in two communities and indicate them with markers in the figure. We see that nodes in the same community tend to be in the same node class. They also tend to clump together in the plot.

Topic analysis. Here we inspect the relation between communities and node attributes by analyze the learned topics in Pubmed dataset, which is constructed from a network of clinical articles. The articles in Pubmed are categorized into three Diabetes Mellitus classes: (1) Experimental, (2) Type 1, and (3) Type



(a) Cora (b) Citeseer

Figure 3: t-SNE visualization of node representations on Cora and Citeseer: color indicates the node class, marker indicates the community assignment. We only choose two non-overlapping communities for visualization.

2. The dataset contains the vocabulary corresponding to node attributes, which allows us to examine the meaning of node representations. We treat each community as a topic to which a document may belong. To determine which attributes are turned on for each community k, we manually set a $\mathbf{c}$ to be a one-hot vector with $c_k = 1$. Then we multiply $\mathbf{c}$ by $\mathbf{v} \sim \text{Gaussian}(\mathbf{0}, \mathbf{I})$. We sample $\mathbf{v}$ 10,000 times and reconstruct $\mathbf{x}$ using $\mathbf{z} = \mathbf{c} \odot \mathbf{v}$ via ATN and get a distribution of "on" attributes. Then we extract words corresponding to these attributes from topic k. We remove non-semantic words using tf-idf.

Our model learned 12 related subareas, which highly related to their own corresponding high frequency stemmed keywords. We show the full names of the abbreviations in Table 9 in the appendix. We show the top eight stemmed keywords for each topic in Table 6. For each topic, these keywords are coherently related to a medical subarea such as treatment, disease, etc. Here we manually assign each topic a label according to the area. For example, by inspecting the keywords, the second topic can be labeled as diabetic retinopathy (DR), which is the most common complication of diabetes mellitus (Wang & Lo, 2018); the third topic can be labeled as coronary artery disease (CAD), which is happened at a higher risk in patients with type 2 diabetes mellitus (T2DM) than non-T2DM patients Naito & Kasai (2015); the ninth topic can be labeled as Islet Cell Antibodies (ICA)[1].

It shows that the learned subareas of diabetes mellitus are all meaningful and interpretable after the analysis of high frequency stemmed keywords. Besides, We also find that each topic is often within a node class in the dataset, which again proves that the detected community is coherent.

6 Related Work

The stochastic blockmodel (Wang & Wong, 1987; Snijders & Nowicki, 1997) is frequently used for detecting and modeling community structure within network data. Later variants of SBM make different assumptions on the node-community relationships (Airoldi et al., 2008; Miller et al., 2009; Latouche et al., 2011). For example, the mixed membership stochastic blockmodel (MMSB) (Airoldi et al., 2008) assumes each node belongs to a mixture of communities. The overlapping stochastic blockmodel (OSBM) (Latouche et al., 2011) assumes each node can belong to multiple communities with the same strengths.

[1]According to Narendran et al. (2005): "the **beta** cell: **ICA**, is the first islet (a **pancreatic** cell) '**autoantigen**' to be described. **Antibodies** to **ICA** are present in 90% of **type 1** diabetes patients at the time of diagnosis."

Topic Label	Top Eight Stemmed Keywords
NAFLD	liver, fat, resist, correl, hepat, befor, obes, degre
DR	retinopathi, complic, diagnosi, mortal, famili, cohort, predict, screen
CAD	cholesterol, genotyp, polymorph, lipoprotein, heart, coronari, target, triglycerid
Hypoglycemic agents	exercis, postprandi, oral, presenc, region, metformin, reduct, agent
Diet intervention	intervent, intak, particip, promot, loss, individu, primari, dietari
Eating disorder	depress, chronic, approxim, dietari, secret, cpeptid, defect, vs
Obesity	period, pathogenesi, greater, continu, daili, meal, weight, dose
AGEs	receptor, alter, neuropathi, streptozotocin, oxid, inhibit, stimul, peripher
ICA	antibodi, transplant, beta, pancreat, antigen, ica, immun, tcell
HbA1c	hypoglycaemia, hba1c, manag, symptom, life, glycaem, known, ii
MS neuropathy	abnorm, metabol, children, nondiabet, durat, nerv, rat, sever
MS uric acid	caus, syndrom, evid, acid, diet, urinari, rat, insulindepend

Table 6: Display of top eight words in twelve learned latent topics in Pubmed dataset. The topic labels are abbreviations of clinical subareas related to Diabetes Mellitus. They are highly related to the top eight stemmed keywords. The details can be found in Table 9 in Appendix.

The VGAE models (Kipf & Welling, 2016; Hasanzadeh et al., 2019; Mehta et al., 2019; Sarkar et al., 2020; Li et al., 2020; Cheng et al., 2021) combine a VAE and a GNN to learn the latent node representation of graph data. DGLFRM (Mehta et al., 2019) replaces the Gaussian prior with Indian Buffet Process (IBP) prior (Teh et al., 2007) to promote interpretabilities of learned representations. LGVG (Sarkar et al., 2020) extends the ladder VAE (Sønderby et al., 2016) to modeling graph data and introduces the gamma distribution to enable interpretability of the learned representation. But these models do not explain the relation between communities and attributes. Other models improve the architectures of different components of VGAE. DGVAE (Li et al., 2020) instead uses the Dirichlet prior and shows its application in node clustering and balanced graph cut. Cheng et al. (2021) devises a model that decodes multi-view node attributes.

The RTM models (Nallapati et al., 2008; Chang & Blei, 2009) focus on learning meaningful topics from the document content, with the help of the relation information that reside in the document network. Various works (Bai et al., 2018; Xie et al., 2021; Panwar et al., 2021) are proposed based on this idea by either improving the graphical model or proposing a novel network architecture.

On combining of the ideas of VGAEs and RTMs, our model is able to learn node representations that not only being useful to performing downstream tasks such as link prediction and node clustering, but also provides highly interpretability in both community-wise and topic-wise, where the former concerns the topological structure of the network and the latter concerns the content of the documents.

7 Conclusion

In this work, we have theoretically analyzed the role of node attribute decoder in representation learning in VGAEs. We show that the node attribute decoder helps the model encode information about the graph structure.

We further propose the NORAD model to learn interpretable node representations. We introduce blockmodel in the edge decoder, with the aim of capturing community structure. We carefully designed the ATN as the node decoder, which improves the quality of node representations and makes them interpretable. We also design a rectification procedure to refine representations of isolated nodes in the graph after model training.

However, our model has the following limitations: (1) The node decoder of our model are limited to documents and can not be extended to other type of features. (2) Although The introduced blockmodel, which attempts to capture the interaction between communities, improves the performance of edge decoding. The learned blockmodel itself still lacks enough interpretability.

References

Seong Jin Ahn and MyoungHo Kim. Variational graph normalized autoencoders. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2827–2831, 2021.

Edoardo Maria Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of machine learning research*, 2008.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pp. 159–168. PMLR, 2018.

Haoli Bai, Zhuangbin Chen, Michael R Lyu, Irwin King, and Zenglin Xu. Neural relational topic models for scientific article analysis. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 27–36, 2018.

Jonathan Chang and David Blei. Relational topic models for document networks. In *Artificial intelligence and statistics*, pp. 81–88. PMLR, 2009.

Jiafeng Cheng, Qianqian Wang, Zhiqiang Tao, Deyan Xie, and Quanxue Gao. Multi-view attribute graph convolution networks for clustering. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 2973–2979, 2021.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning*, pp. 2434–2444. PMLR, 2019.

Arman Hasanzadeh, Ehsan Hajiramezanali, Nick Duffield, Krishna R Narayanan, Mingyuan Zhou, and Xiaoning Qian. Semi-implicit graph variational auto-encoders. *arXiv preprint arXiv:1908.07078*, 2019.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2 (1-2):83–97, 1955.

Pierre Latouche, Etienne Birmelé, and Christophe Ambroise. Overlapping stochastic block models with application to the french political blogosphere. *The Annals of Applied Statistics*, pp. 309–336, 2011.

Jia Li, Tomasyu Yu, Jiajin Li, Honglei Zhang, Kangfei Zhao, Yu Rong, Hong Cheng, and Junzhou Huang. Dirichlet graph variational autoencoder. *arXiv preprint arXiv:2010.04408*, 2020.

Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. Modeling user exposure in recommendation. In *Proceedings of the 25th international conference on World Wide Web*, pp. 951–961, 2016.

Li-Ping Liu and David M Blei. Zero-inflated exponential family embeddings. In *International Conference on Machine Learning*, pp. 2140–2148. PMLR, 2017.

Nikhil Mehta, Lawrence Carin Duke, and Piyush Rai. Stochastic blockmodels meet graph neural networks. In *International Conference on Machine Learning*, pp. 4466–4474. PMLR, 2019.

Kurt Miller, Michael Jordan, and Thomas Griffiths. Nonparametric latent feature models for link prediction. *Advances in neural information processing systems*, 22:1276–1284, 2009.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Ryo Naito and Takatoshi Kasai. Coronary artery disease in type 2 diabetes mellitus: Recent treatment strategies and future perspectives. *World journal of cardiology*, 7 3:119–24, 2015.

Ramesh M Nallapati, Amr Ahmed, Eric P Xing, and William W Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 542–550, 2008.

P. Narendran, E. Estella, and S. Fourlanos. Immunology of type 1 diabetes. *QJM: An International Journal of Medicine*, 98(8):547–556, 06 2005. ISSN 1460-2725. doi: 10.1093/qjmed/hci088. URL `https://doi.org/10.1093/qjmed/hci088`.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

Shirui Pan, Ruiqi Hu, Sai-fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning graph embedding with adversarial training methods. *IEEE transactions on cybernetics*, 50(6):2475–2487, 2019.

Madhur Panwar, Shashank Shailabh, Milan Aggarwal, and Balaji Krishnamurthy. TAN-NTM: Topic attention networks for neural topic modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3865–3880, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.299. URL `https://aclanthology.org/2021.acl-long.299`.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Arindam Sarkar, Nikhil Mehta, and Piyush Rai. Graph representation learning via ladder gamma variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5604–5611, 2020.

Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 253–260, 2002.

Tom AB Snijders and Krzysztof Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of classification*, 14(1):75–100, 1997.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in neural information processing systems*, 29:3738–3746, 2016.

Yee Whye Teh, Dilan Grür, and Zoubin Ghahramani. Stick-breaking construction for the indian buffet process. In *Artificial Intelligence and Statistics*, pp. 556–563. PMLR, 2007.

Yu Tian, Long Zhao, Xi Peng, and Dimitris Metaxas. Rethinking kernel methods for node representation learning on graphs. *Advances in neural information processing systems*, 32:11686–11697, 2019.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Wei Wang and Amy C. Y. Lo. Diabetic retinopathy: Pathophysiology and treatments. *International Journal of Molecular Sciences*, 19(6), 2018. ISSN 1422-0067. doi: 10.3390/ijms19061816. URL `https://www.mdpi.com/1422-0067/19/6/1816`.

Yuchung J Wang and George Y Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

Qianqian Xie, Yutao Zhu, Jimin Huang, Pan Du, and Jian-Yun Nie. Graph neural collaborative topic model for citation recommendation. *ACM Transactions on Information Systems (TOIS)*, 40(3):1–30, 2021.

Dataset	Nodes	Edges	Words	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,312	4,732	3,703	6
Pubmed	19,717	44,338	500	3
DBLP	17,716	105,734	1,639	4

Table 7: Dataset statistics

Training ratio	Cora				Citeseer			
	20%	40%	60%	80%	20%	40%	60%	80%
# isolated nodes	1,370	702	336	120	2,045	1,275	737	341
% isolated nodes	50.6%	25.9%	12.4%	4.4%	61.8%	38.5%	22.3%	10.3%
#contributed edges	3,584	1,445	547	155	3,883	1,955	958	385
%contributed edges	66.0%	26.6%	10.1%	2.9%	82.1%	41.3%	20.3%	8.1%
Training ratio	Pubmed				DBLP			
	20%	40%	60%	80%	20%	40%	60%	80%
#isolated nodes	11,220	7,196	4,290	1,964	8,290	4,664	2,628	1102
%isolated nodes	56.9%	36.5%	21.8%	10.0%	46.8%	26.3%	14.8%	6.2%
#contributed edges	19,919	9,938	5,156	2,125	19,915	7,765	3,525	1249
%contributed edges	45.0%	22.4%	11.6%	4.8%	18.8%	7.3%	3.3%	1.2%

Table 8: Isolated nodes numbers and percentages in different training ratio.

A Appendix

A.1 Datasets details

Cora. Citation network consists of 2,708 documents from seven categories. The dataset contains bag-of-words feature vectors of length 1,433. The network has 5,278 links.

Citeseer. Citation network consists of 3,312 scientific publications from six categories. The dataset contains bag-of-words feature vectors of length 3,703. The network has 4,732 links.

Pubmed. Citation network consists of 19,717 scientific publications from three categories. The dataset contains bag-of-words feature vectors of length 500. The network has 44,338 links.

DBLP. Citation network consists of 17,716 papers from categories classes. The dataset contains bag-of-words feature vectors of length 1,639. The network 105,734 links.

Isolated nodes in different training ratio. In Table 8, we calculate the numbers and percentages of isolated nodes in different training set split ratios, we also get the numbers and percentages of edges that contribute to generating the isolated nodes.

Abbreviation	Full Name
NAFLD	Nonalcoholic Fatty Liver Disease
DR	Diabetic Retinopathy
CAD	Coronary Artery Disease
AGEs	Advanced Glycation End products
ICA	Islet Cell Antibodies
MS	Metabolic Syndrome

Table 9: Abbreviation and full name of learned topics.

A.2 Model implementation and experimental details

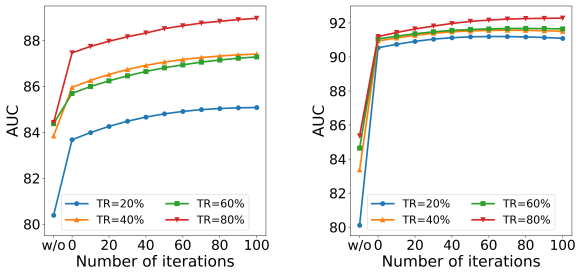
In this section, we introduce the implementation for NORAD and the detailed setups of some experiments.

Hyperparmeters	VGAE	KernelGCN	DGLFRM	VGNAE	NORAD
Layer type	GCN	GCN	GCN	APPNP	GCN
#Layers	2	2	2	1	1
Hidden dimension	{32, 16}	{32, 16}	{256, 50}	{128}	{256}
Prior	Gaussian	Gaussian	Gaussian+IBP	Gaussian	Gaussian+Bernoulli

Table 10: Encoder configuration of each models

Hyperparameters setting. For the encoder of NORAD, we choose 1-layer Graph Convolution Network (GCN) as our encoders. Since we need to output three sets of variational parameters, we use three GCN layers separately. Each encoder shares the same output dimension. We search the number of cluster K over $\{32, 64, 128, 256\}$ and find that our model is insensitive to K. When K becomes larger (usually 64 and above), the model gives a relatively stable performance. We choose $K = 256$ for all models in the reported experiments. For other VGAE baselines, we observe a slight performance drop when increasing the dimension of the hidden layers of the encoder. Though node representation learned by NORAD are in higher dimension, we argue that the actual number of the effective entries is usually much smaller. For example, we only observe 12 effective entries in Pubmed dataset. A detailed configurations of the encoders of the baselines and our models are shown in Table 10. For the node decoder ATN, we search (d_e, d_h) over $\{(128, 64), (64, 32)\}$. We set d_e to be 128 and d_h to be 64 in our experiment.

Training and prediction. We alternatively optimize (ϕ, θ) and $\mathbf{B}$. We first update (ϕ, θ) for E steps, then update $\mathbf{B}$ for M steps, and alternate until convergence. In the implementation, we set $E = 10$ and $M = 10$. We use Adam optimizers Kingma & Ba (2014) with learning rate 0.001. Since we use the gumbel-softmax Jang et al. (2016) to relax the binary vector $\mathbf{C}$, in the training process, we use temperature annealing with 0.5 to be the minimum temperature. We use the relaxed binary vector for optimizing (ϕ, θ), and for optimizing $\mathbf{B}$ and prediction phase, we use the binary vector by truncating the Bernoulli parameters π. For isolated nodes rectification, we use the same learning rate to rectify the representation of the isolated nodes. We optimize the representation for multiple iterations and choose the number of iterations to be 50 when reporting the experiment results.



(a) Cora (b) Citeseer

Figure 4: Isolated nodes link prediction and representation rectification: (a) For Cora, AUC of isolated nodes related link prediction is improved by adding ATN decoder (w/o $\rightarrow$ iteration 0) and increasing rectification iterations (iteration 0 $\rightarrow$ 100); (b) For Citeseer, the increase is more obvious with lower training ratio.

Model diagnosis. Here we show the detailed implementation for each extra component mentioned in Section 5.2. For the probability construction $p(\mathbf{Z})p(\mathbf{X}|\mathbf{Z})p(\mathbf{A}|\mathbf{Z}, \mathbf{X})$, $\mathbf{Z}$ is used for two purposes: reconstructs $\mathbf{X}$ and reconstructs $\mathbf{A}$ with the help of $\mathbf{X}$. For $p(\mathbf{X}|\mathbf{Z})$, we keep using the ATN as the node decoder. For $P(\mathbf{A}|\mathbf{Z}, \mathbf{X})$, we first use a node encoder to encode $\mathbf{X}$ into a hidden vector $\mathbf{R}$, which has the same dimension as $\mathbf{Z}$. Then we concatenate $\mathbf{Z}$ with $\mathbf{R}$ and feed it into the edge encoder. For the node encoder, we use a two-layer MLP with ReLU activation function. For the edge decoder in DGLFRM, we use one linear layer along with the ReLU activation function. We set the dimension of the output $\hat{\mathbf{Z}}$ of the MLP to be half the number of the clusters K. We use $\hat{\mathbf{Z}}$ to reconstruct $\mathbf{A}$ via inner product. And for reconstructing $\mathbf{X}$, we still use $\mathbf{Z}$. Isolated node rectification is employed for all the model variants with node decoders.

Ratio	base	w/ norm	w/ ATN	w/ both
		Cora		
20%	82.7±0.61	86.2±1.15	86.4±0.58	87.9±0.49
40%	88.5±0.71	89.8±0.52	90.7±0.44	91.2±0.51
60%	91.9±0.63	91.8±0.61	93.2±0.37	92.6±0.71
80%	94.0±0.75	93.0±1.25	95.2±0.56	93.7±1.14
		Citeseer		
20%	88.5±0.76	85.9±1.34	92.2±0.55	91.2±0.47
40%	91.7±0.51	90.1±1.04	93.7±0.39	93.3±0.72
60%	92.8±0.58	91.6±1.07	94.5±0.42	94.1±0.66
80%	93.9±0.62	92.8±1.09	95.6±0.48	94.8±0.70

Table 11: Performance comparison of normalization trick and ATN decoder: NORAD without ATN (denoted as base), base with normalization trick (denoted as norm), NORAD (denoted as ATN), NORAD with normalization trick (denoted as both).

A.3 Additional Experiments

Further result analysis of isolated nodes. Here we show how using a node decoder can greatly improve the link prediction performance for isolated nodes. Figure 4 shows the superior capability of suggesting links for isolated nodes, especially in Citeseer.

Normalization trick on NORAD. We also experiment with deploying normalization trick (Ahn & Kim, 2021) in our model. We add L2 normalization on the encoded features in the GCN layer. We compare the performance of adding only the normalization trick or only our ATN on the Cora and Citeseer dataset with four different ratios. We also try adding both the normalization trick and our ATN to thoroughly study their effects. We show the performances of four variants in Table 11. We can find that our ATN decoder outperforms the normalization trick on both datasets with all training ratios. The difference is more obvious on the sparser graph dataset Citeseer. We also find that the combination of these two operations yields trivial improvement on Cora with some training ratios, and even gets worse on Citeseer compared with using only one of the operations.

t-SNE visulization of baselines. In Figure 5, we visualize the node embeddings learned from NORAD, KernelGCN, DGLFRM, and VGNAE on Cora and Citeseer. Compared with other baselines, we can better interpret the community structure from the node embeddings learned by our model.

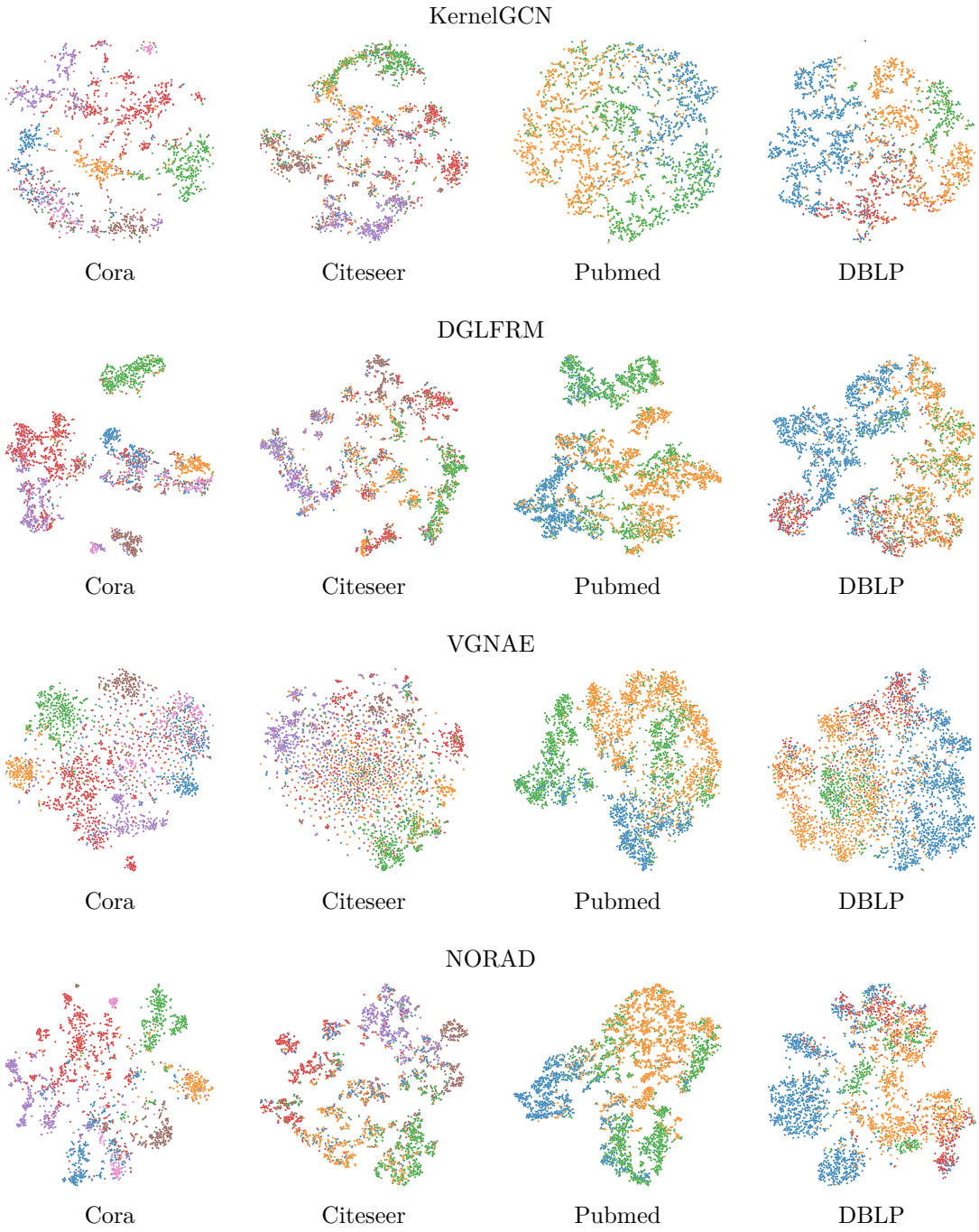


Figure 5: Visualization of latent node embeddings learned by four models on four datasets.