Interpretable Deep Graph Generation with Node-edge Co-disentanglement

Xiaojie Guo
xguo7@gmu.edu
George Mason University, USA

Liang Zhao*
lzhao9@gmu.edu
George Mason University, USA

Zhao Qin
zqin02@syr.edu
Syracuse University, USA

Lingfei Wu
wuli@us.ibm.com
IBM Research AI, USA

Amarda Shehu
ashehu@gmu.edu
George Mason University, USA

Yanfang Ye
yanfang.ye@case.edu
Case Western Reserve University, USA

ABSTRACT

Disentangled representation learning has recently attracted significant amount of attentions, particularly in the field of image representation learning. However, learning the disentangled representations behind a graph remains largely unexplored, especially for the attributed graph with both node and edge features. Disentanglement learning for graph generation has substantial new challenges including: 1) the lack of graph deconvolution operations to jointly decode node and edge attributes; and 2) the difficulty in enforcing the disentanglement among latent factors that respectively influence: i) only nodes, ii) only edges, and iii) joint patterns between them. To address these challenges, we propose a new disentanglement enhancement framework for deep generative models for attributed graphs. In particular, a novel variational objective is proposed to disentangle the above three types of latent factors, with novel architecture for node and edge deconvolutions. Qualitative and quantitative experiments on both synthetic and real-world datasets demonstrate the effectiveness of the proposed model and its extensions.

CCS CONCEPTS

• **Computing methodologies** → **Unsupervised learning**; **Neural networks**; *Generative and developmental approaches*; • **Mathematics of computing** → *Graph algorithms*; • **Information systems** → *Data mining*; • **Networks** → *Topology analysis and generation.*

KEYWORDS

Deep generative models, Graph Generation, disentangled learning.

*Corresponding author:lzhao9@gmu.edu

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403221

ACM Reference Format:
Xiaojie Guo, Liang Zhao, Zhao Qin, Lingfei Wu, Amarda Shehu, and Yanfang Ye. 2020. Interpretable Deep Graph Generation with Node-edge Co-disentanglement. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403221

1 INTRODUCTION

Recent advances in deep generative models, such as variational auto-encoders (VAE) [24] and generative adversarial networks (GAN) [16], have made important progress towards generative modeling for complex domains, such as image data. The goal here is to learn the underlying (low-dimensional) distribution of the images, hence image generation is treated as sampling from learned distributions. Building on these techniques for images, which can be considered as grid-structured data, a number of deep learning models for generating general graphs have been proposed over the last couple of years [25, 32, 43]. These involves real-world applications, such as discovering new chemical and molecular structures [10, 28], and constructing knowledge graphs.

When we learn the underlying distribution of complex data such as images, learning interpretable representations of data that expose semantic meaning is very important. Such representations are useful not only for standard downstream tasks such as supervised learning and reinforcement learning, but also for tasks such as transfer learning and zero-shot learning where humans excel but machines struggle [29]. As yet, most research has focused on learning factors of variations in the data, commonly referred to as learning a disentangled representation, where the variables of the representation are highly independent. Examples of this include variables that only control the size of objects, or their color. For the instance in Fig. 1 (a), where a semantic factor controls the degree of smile in a human facial image.

However, in the promising domain of deep generative models for graph generation, disentangled enhancement has rarely been well explored yet, but could be highly beneficial for applications such as controlling the generation of protein structures, or designing Internet of Things (IoT). As shown in Fig. 1, we would love to generalize from an image situation to a graph situation, where the variables control specific factors related to node attributes, edge attributes, or joint-node-edge patterns in the graph. For example, Fig. 1 (a) shows the semantic factor (i.e. smile) in the images, which

can be regarded as special cases of graphs where nodes are pixels that are connected in a fixed topology. All the factors that control image formulation are effectively node-related. Fig. 1(b) shows the factors that control the formulation of a cyber network, which is an attributed graph where computers are nodes and their links are edges. Unlike images, there are three types of factors formulating the networks: (1) node-related factors that control some properties of node attributes but are independent of edge patterns (e.g., the CPU usage of each computer); (2) edge-related factors that only influence edge patterns but are independent of node patterns (e.g., geo-spatial distances between computers); and (3) node-edge-joint factors that jointly influence some properties from both nodes and edges (e.g., the node patterns of "downloaded data amount" and edge patterns "network traffic" which are inherently highly entangled and hence must be controlled by such factors). Thus, it is necessary to develop a generic model to discover and disentangle all three types of factors for the graph data. Though a few researchers have sought to apply the disentanglement learning to graphs [33, 36, 44], as yet they only have identified the latent factors that caused the edge between a node and its neighbors.

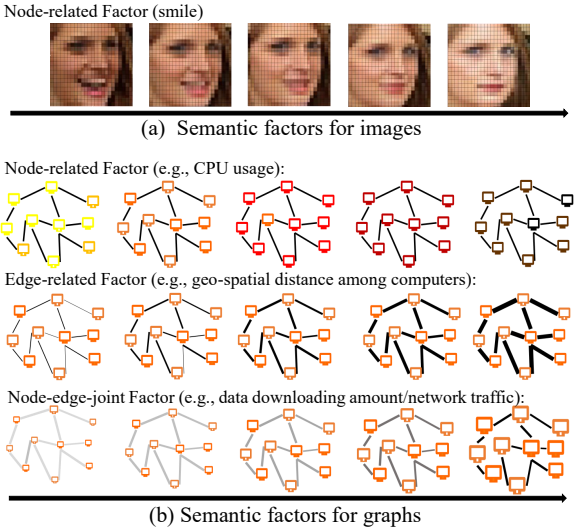

Figure 1: Two examples of disentanglement: (a) semantic factors of images, where each pixel is a node and each pixel is connected to its eight neighboring pixels, and (b) semantic factors of cyber networks, where each computer is a node and the link between each pair of computers is an edge (better seen in color).

In this paper, we focus on the generic problem of disentangled representation learning on attributed graphs, where the characteristics of graphs pose great challenges to disentanglement learning on graphs: 1) **Lack of node and edge joint deconvolution operations**. The formation process for real-world graphs, which is both complex and iterative, is based on the three types of factors depicted in Fig. 1. For example, edges are generated not only by the edge related factors, but also node-edge-joint related factors. There is no existing graph decoder that can simultaneously handle all three types of factors during the generation process. 2) **Complex disentanglement enhancement of multiple types of latent representation**. Although the three types of semantic factors mentioned in Fig. 1 are independent from each other, it is extremely difficult to enforce that. First, it is difficult to automatically categorize individual factors into these three types. Second, even they are categorized, the enhancement of such independency patterns still cannot be accomplished by the existing techniques which mostly focus on images without categorization capability. 3) **The dilemma between disentanglement and reconstruction quality for attributed graphs**. Disentangling the three types of factors and reconstructing both edges and nodes can require multiple trade-offs between reconstruction errors and disentanglement performance during the training process. For example, the objective of disentanglement of node-edge-joint factors can conflict with not only edge but also node reconstruction errors.

To the best of our knowledge, this is the first work that can address all the above challenges and provides a generic framework that incorporates multiple disentanglement enhancements for attributed graphs. We propose the new Node-Edge Disentangled Variational Auto-encoder (NED-VAE) model, a deep unsupervised generative approach for disentanglement learning on graphs that automatically discovers the independent latent factors in both edges and nodes. A novel objective for node-edge jointly disentanglement is derived and proposed based on the variational autoencoder (VAE) [24, 38]. A novel architecture is proposed consisting of three sub-encoders and two sub-decoders to model the complicated relationships between nodes and edges. We also propose a general framework of objectives that can include various extensions of the base NED-VAE to realize the group-wise and variable-wise disentanglement. The contributions of this work are summarized as follows:

- **A novel framework is proposed for the disentanglement of attributed graph generation**. A novel objective framework is derived for learning three factors that are exclusive to node patterns, exclusive to edge patterns, and those spanning node-edge-joint patterns.
- **A novel architecture is proposed for disentanglement learning on graphs.** Derived from the theoretical objective, an architecture is proposed with three sub-encoders (a node encoder, an edge encoder, and a node-edge co-encoder) to learn three types of representations, and two sub-decoders (a node-decoder and an edge decoder) to co-generate both nodes and edges.
- **Simultaneous group-wise and variable-wise disentanglement**. The proposed framework hierarchically disentangles attributed graph generation according to node, edge, and their joint factors. A set of varational auto-encoder-based models for attributed graphs have been proposed.
- **Comprehensive experiments have been conducted to validate the effectiveness of our proposed model and its extensions**. Qualitative and quantitative experiments on two synthetic and two real-world datasets demonstrate that NED-VAE and its extensive models are indeed capable of learning disentangled factors for different types of graphs.

2 RELATED WORKS

Disentangled Representation Learning. Disentangled representation learning has gained considerable attention, in particular in the field of image representation learning [2, 7, 22, 23]. The goal is to learn representations that separate out the underlying explanatory

factors responsible for variations in the data. Such representations have been shown to be relatively resilient to the complex variants involved [4], and can be used to enhance generalizability as well as improve robustness against adversarial attack [2]. The disentangled representations are inherently more interpretable, and can thus potentially facilitate debugging and auditing [11]. This has prompted a number of approaches that modify the VAE objective by adding, removing, or altering the weight of individual terms [2, 7, 14, 23, 27, 35, 49]. However, the best way of learning representations that disentangle the latent factors behind a graph remains largely unexplored.

Graph neural networks and graph generation. Recent work on graph neural networks (GNNs) [17, 41], especially graph convolutional networks [6, 21], is attracting considerable attention, because of their remarkable success in multiple domains such as natural language processing [8, 9], computer vision [42], software engineering [30] and traffic flow prediction [31]. Self-attention mechanisms and sub graph-level information have also been explored as ways to potentially improve the representation power of learned node embeddings [3, 15, 45]. Most of the existing GNN based graph generation methods are based on VAE [40, 43] and generative adversarial nets (GANs) [5], and others [32, 48]. For example, GraphRNN [48] builds an autoregressive generative model on these sequences utilizing LSTM model and has demonstrated good scalability; while GraphVAE [43] represents each graph in terms of its adjacent matrix and feature vector and utilizes the VAE model to learn the distribution of the graphs conditioned on a latent representation at the graph level. Graphite [18] and VGAE [26] encode the nodes of each graph into node-level embeddings and predict the links between each pair of nodes to generate a graph. Some conditional graph generation methods also provide powerful graph encoders and decoders for attributed graphs where both node and edge attributes are considered [19, 20].

3 PROBLEM FORMULATION

Define an input graph as $G(\mathcal{V}, \mathcal{E}, E, F)$, where $\mathcal{V}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of M edges. $\mathcal{E}$ contains all pairs of nodes, while the existence of each edge is reflected by one of its attributes. $E \in \mathbb{R}^{N\times N\times K}$ is the edge attributes tensor, where K is the dimension of the edge attributes. $F \in \mathbb{R}^{N\times D}$ refers to the node attribute matrix, where $F_i \in \mathbb{R}^{1\times D}$ is the node attributes of node i and D is the dimension of the node attribute vector. As shown in Fig. 1, three types of factors (i.e. node-related factors, edge-related factors and node-edge-joint related factors) are assumed to control the generation of the graph G.

The goal is to develop an unsupervised deep generative model that can learn the joint distribution of the graph G and three groups of generative latent variables $Z = (z_e \in \mathbb{R}^{L_1}, z_f \in \mathbb{R}^{L_2}, z_g \in \mathbb{R}^{L_3})$ (L_1, L_2, and L_3 are the number of variables in each group) to discover the three types of factors, such that the observed graph G can be generated as $p(G|Z) = P(E, F|z_e, z_f, z_g)$. There are three challenges must be overcome to achieve the above goal: (1) The lack of co-decoder based on co-deconvolution to jointly generate both the nodes attributes F and edges attributes E; (2) difficulty of enforcing independence among the variable groups z_e, z_f and z_g (group-wise disentanglement), rather than simply enforcing the disentanglement of the variables inside z_e, z_f and z_g (variable-wise disentanglement); and (3) the need to simultaneously solve multiple reconstruction-disentanglement conflicts in z_e and E, z_f and F, z_g and E, and z_g and F.

4 NODE-EDGE DISENTANGLEMENT VAE

In this Section, we first introduce the derived training objective and the architecture of the proposed Node-edge Disentanglement VAE (NED-VAE). Then we propose a generic objective framework as well as its derivation to further enforce the disentanglement of NED-VAE models with different purposes.

4.1 Objective and Architecture

In this section, we first derive the objective for learning disentanglement on graphs. Then, to solve the first challenge, we propose a new architecture, the NED-VAE, based on the derived objectives.

4.1.1 The objective for disentanglement on graphs. For a given observation $G = (E, F)$, we describe the inferred posterior configurations of the latent factors $Z = (z_e, z_f, z_g)$ using a probability distribution $q_\phi(z_e, z_f, z_g|E, F)$. Our aim is to ensure that the inferred latent factors $q_\phi(z_e, z_f, z_g|E, F)$ capture all three types of generative factors in a disentangled manner. In order to encourage this disentangling property in the inferred $q_\phi(z_e, z_f, z_g|E, F)$, we can introduce a constraint by trying to match it to a prior $p(z_e)$, $p(z_f)$ and $p(z_g)$ that both controls the capacity of the latent information bottleneck, and embodies the statistical independence mentioned above. This can be achieved if we set the prior to be an isotropic unit Gaussian, i.e. $p(Z) = p(z_e, z_f, z_g) = \mathcal{N}(0, 1)$, leading to the constrained optimisation problem in Eq. 1, where ϵ specifies the strength of the applied constraint:

$$\max_{\theta,\phi} \mathbb{E}_{G\sim D}[\mathbb{E}_{q_\phi(Z|G)} log p_\theta(E, F|z_e, z_f, z_g)] \\ s.t. D_{KL}(q_\phi(z_e, z_f, z_g|E, F)||p(z_e, z_f, z_g) < \epsilon. \quad (1)$$

Eq. 1 can be rewritten as a Lagrangian under the KKT conditions and, according to the complementary slackness KKT condition, we therefore arrive at the β-VAE [22] formulation, which takes the form of the familiar variational free energy objective function:

$$\mathcal{L}(\theta, \phi, G, Z, \beta) = \mathbb{E}_{q_\phi(Z|G)}[log p_\theta(E, F|z_e, z_f, z_g)] \\ - \beta D_{KL}(q_\phi(z_e, z_f, z_g|E, F)||p(z_e, z_f, z_g). \quad (2)$$

Based on the definitions of z_f, z_e, and z_g, namely that z_f only controls some properties of nodes, z_e only controls some properties of edges and z_g controls the properties of both, we obtain:

$$q_\phi(z_e, z_f, z_g|E, F) = q_\phi(z_f|F)q_\phi(z_e|E)q_\phi(z_g|E, F) \quad (3)$$

$$p_\theta(E, F|z_e, z_f, z_g) = p_\theta(F|z_f, z_g)p_\theta(E|z_e, z_g) \quad (4)$$

We can now rewrite the loss function as:

$$\mathcal{L}(\theta, \phi, G, Z, \beta) = \mathbb{E}_{q_\phi(Z|G)}[log p_\theta(F|z_f, z_g)p_\theta(E|z_e, z_g)] \\ - \beta D_{KL}(q_\phi(z_f|F)q_\phi(z_e|E)q_\phi(z_g|E, F)||p(z_e)p(z_f)p(z_g)) \\ = \mathbb{E}_{q_\phi(Z|G)}[log p_\theta(F|z_f, z_g)p_\theta(E|z_e, z_g)] - \beta D_{KL}(q_\phi(z_f|F)||p(z_f)) \\ - \beta D_{KL}(q_\phi(z_e|E)||p(z_e)) - \beta D_{KL}(q_\phi(z_g|E, F)||p(z_g)) \quad (5)$$

To maximize the above objective, a deep generative model is needed to model each of the components in this objective.

4.1.2 The architecture of the node-edge disentangled VAE. Derived from the above objective, the Node-Edge Disentangled VAE model (NED-VAE) is proposed based on a novel architecture. The architecture of the proposed model is shown in Fig. 2.

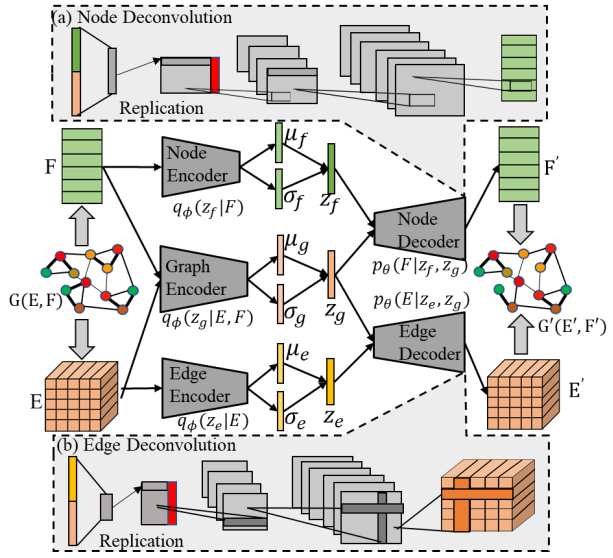

Figure 2: The architecture of the proposed NED-VAE consist of three sub-encoders to inference z_e, z_f and z_g, as well as two sub-decoders to reconstruct E and F simultaneously.

The overall framework is based on the traditional VAE, where the encoder learns the mean and standard deviation of the latent representation of the input and the decoder decodes the sampled latent representation vector to reconstruct the input. Unlike the structure of traditional VAE, the proposed framework has three encoders, each of which models the distributions $q_\phi(z_f|F)$, $q_\phi(z_g|E, F)$ or $q_\phi(z_e|E)$; and two novel decoders to model $p_\theta(F|z_g, z_f)$ and $p_\theta(E|z_g, z_e)$, that jointly generate the node and edge attributes based on the three types of latent representations. Each type of representations is sampled by their own inferenced mean and standard derivation. For example, the representation vectors z_f are sampled as $z_e = \mu_f + \sigma_f \odot \epsilon$, where ϵ follows a standard normal distribution. This architecture also partially solves the second challenge described above because it enforces the disentanglement between the two groups of variables z_e and z_f by separating their inference process. The details of each components are described as follows.

Node, edge and graph encoder. The *node encoder* consists of several traditional convolution layers to extract latent features from node attribute matrix F; and two paths of fully connected layers to get the mean μ_f and standard derivation vectors σ_f of the node representation distribution. The *edge encoder* consists of several edge convolution layers proposed by Guo et al. [19] to extract edge representations from the edge attribute tensor E; and edge embedding layers to get the node-level representation; and fully connected layers to yield the mean μ_e and standard derivation σ_e vectors of the edge representation distribution. The *graph encoder* consists of several graph convolution layers proposed in [25] to get node-level representations; and fully connected layers to aggregate the learned node representations into a graph-level representation that can be separately mapped into the mean μ_g and standard derivation σ_g vectors of the graph representation distribution [1].

Node decoder. The proposed node decoder aims to generate the node attribute matrix F' based on the sampled node representations z_f and graph representations z_g, which ensures the node attribute generation process is controlled by both the node-related factors and the node-edge-joint related factors. As shown in Fig. 2 (a), the node decoder consists of several traditional deconvolution layers and fully connected layers as a reversed process of the node encoder. First, the input node representation z_f and graph representation z_g are concatenated together and mapped into several fully connected layers to decode the vector into multiple feature vectors. Next, we aim to convert each feature vector into a feature matrix, where each row should refer to an individual node, by replicating each feature vectors N times. Moreover, to ensure the diversity and randomness of the nodes in each graph, a node assignment vector $S \in \mathbb{R}^N$ (shown as a red rectangle in Fig. 2 (a) is sampled following the normal distribution and is concatenated with each feature matrix. Thirdly, once the feature matrix has been obtained, one-dimensional filters are used to deconvolute each row of the feature matrix into the attribute vectors for each node, completing the reconstruction of the input node attribute matrix F.

Edge Decoder. The proposed edge decoder aims to generate the reconstructed edge attribute E based on the sampled node representations z_e and graph representations z_g, ensuring that the edge attributes generation is controlled by both the edge-related and node-edge-joint related factors. The proposed edge decoder consists of several edge deconvolution layers and fully connected layers as a reversed process of the edge encoder. The input is the concatenation of both the edge representation z_e and the graph representation z_g. First, the input vector is mapped into a node-level feature vector through a fully connected layer and is converted into a matrix by being replicated. The same node assignment vector S is also concatenated to this feature matrix. The hidden edge feature matrices are then generated by the edge-node deconvolution layer [19] by decoding each of the node-level representations, where the principle is that each node's representation can make contributions to the generation of its related edges features (contributions are shown as dark grey rectangles in Fig. 2 (b)). Thirdly, the edge-attribute tensor E is generated through the edge-edge deconvolution layer, where the principle is that each hidden edge feature can contribute to the generation of its adjacent edges.

4.2 Framework of node-edge co-disentanglement

To solve the second and third challenges, we propose a generic objective framework to further enforce the disentanglement of NED-VAE models with different purposes. In Section 4.2.1, the basic overall framework with four terms are introduced, namely two conditional distribution terms of the graphs (denoted as ①), the latent representations term (denoted as ②), the marginal distribution term for the graphs (denoted as ③), and the inferred prior distributions (denoted as ④). In Section 4.2.2, we move on to further enforce the disentanglement among variable groups by generalizing the term ④ to introduce a novel node-edge-total-correlation term

[1] Operation details of the encoders can be found in https://github.com/xguo7/NED-VAE.

(denoted as Ⓐ) for group-wise disentanglement and a variable-wise disentanglement term (denoted as Ⓒ). Next, in Section 4.2.3, we further enforce the disentanglement inside the three types of latent representations, generalizing the term Ⓒ to introduce three variable-total-correlation terms (denoted as $Ⓐ_f$, $Ⓑ_f$, and $Ⓒ_f$). Furthermore, based on the framework, six extensions of the base NED-VAE models are proposed that enforce different terms, as shown in Table 1.

Table 1: Summary of objectives of the extensions of NED-VAE model. ($Ⓒ^*$ refers to the sum of $Ⓒ_e$, $Ⓒ_f$ and $Ⓒ_g$; $②^a_e$ can be changed to $②^a_f$ or $②^a_g$)

NED-VAE	①+③+β(②+④)
NED-IPVAE-I	①+③+②+λ④
NED-IPVAE-II	①+③+λ④
NED-HCVAE	①+③+②+γⒶ
NED-TCVAE	①+③+②+Ⓒ+βⒶ
NED-VTCVAE	①+③+②+$Ⓒ^*$+βⒶ+$\gamma_1 Ⓐ_f$+$\gamma_2 Ⓐ_e$+$\gamma_3 Ⓐ_g$
NED-AnchorVAE	①+③+②+④-$\lambda ②^a_e$

4.2.1 Overall graph disentanglement framework. As proved by Esmaeili et al. [14], the VAE objective can be equivalently defined as a KL divergence between the generative model $p_\theta(x,z)$ and inference model $q_\phi(z,x) = q_\phi(z|x)q(x)$. Inspired by this and given that $p(z_1,z_2,z_3) = p(z_1)p(z_2)p(z_3)$, in conjunction with Eq. 4, the NED-VAE objective for the graph data can be defined as:

$$
\begin{aligned}
&- D_{KL}(p_\theta(z_e,z_f,z_g,E,F)||q_\phi(E,F,z_e,z_f,z_g)) \\
&= \mathbb{E}_{q_\phi(Z,G)}[log\frac{p_\theta(E,F,z_e,z_f,z_g)}{p_\theta(E,F)p(z_e,z_f,z_g)} + log\frac{q(E,F)q_\phi(z_e,z_f,z_g)}{q_\phi(E,F,z_e,z_f,z_g)} \\
&+ log\frac{p_\theta(E,F)}{q(E,F)} + log\frac{p(z_e,z_f,z_g)}{q_\phi(z_e,z_f,z_g)}] \\
&= \mathbb{E}_{q_\phi(Z,G)}[log\frac{p_\theta(E,F|z_e,z_f,z_g)}{p_\theta(E,F)} - log\frac{q_\phi(z_e,z_f,z_g|E,F)}{q_\phi(z_e,z_f,z_g)}] \\
&- KL(q(E,F)||p_\theta(E,F)) - KL(q_\phi(z_e,z_f,z_g)||p(z_e,z_f,z_g)) \\
&= \mathbb{E}_{q_\phi(Z,G)}[\underbrace{log\frac{p_\theta(F|z_f,z_g)p_\theta(E|z_e,z_g)}{p_\theta(E,F)}}_{①} \\
&\underbrace{- log\frac{q_\phi(z_e|E)q_\phi(z_f|F)q_\phi(z_g|E,F)}{q_\phi(z_e)q_\phi z_f)q_\phi(z_g)}}_{②}] \\
&\underbrace{-KL(q(E,F)||p_\theta(E,F))}_{③} \underbrace{- KL(q_\phi(z_e,z_f,z_g)||p(z_f)p(z_e)p(z_g))}_{④} \quad (6)
\end{aligned}
$$

Specifically, Terms ③ and ④ enforce consistency between the marginal distributions over $G = (E,F)$ and $Z = (z_e,z_f,z_g)$. Minimizing the KL divergence in Term ③ maximizes the marginal likelihood $\mathbb{E}_{q(E,F)}logp_\theta(E,F)$; maximizing Term ④ which is named as inferred priors term enforces the distance between $q_\phi(z_e,z_f,z_g)$ and $p(z_e,z_f,z_g)$. Terms ① and ② enforce consistency between the conditional distributions. Specifically, Term ① maximizes the correlation for each Z that generates each G^n; when $Z \sim q_\phi(Z|G^n)$ is sampled, the likelihood $p_\theta(G^n|Z)$ should be higher than the marginal likelihood $p_\theta(G^n)$. Meanwhile Term ② regularizes Term ① by minimizing the mutual information $I(Z,G)$ in the inference model.

Since Term ② actually represents the mutual information between the latent z_e, z_f, z_g and the graphs G, this will lead to poor reconstructions when enforcing disentanglement with high values of β in the proposed NED-VAE [37]. Thus, to solve the trade-off problems between the disentanglement of z_e, z_f, z_g and G, we propose to either enforce Term ④ alone or enforce it with high weights. Accordingly, we can refer to the model enforcing only Term ④ as (Node-edge disentangled Inferred Priors VAE) NED-IPVAE-I, and the model enforcing both ② and ④ with different weights as NED-IPVAE-II, as shown in Table 1.

4.2.2 Generalization of the Inferred Priors Term ④. Next, to further address the second challenge and enforce the disentanglement among groups of variables z_e, z_f and z_g, we further generalize the Term ④ by decomposing it and introduce the Node-edge Total Correlation term (Ⓐ in Table. 1). Specifically, Term ④ can be decomposed into sub components Ⓐ, Ⓑ and Ⓒ, as the followings (Here, we use Z to denote (z_e,z_f,z_g) for clarity):

$$
\begin{aligned}
④ &\rightarrow -\mathbb{E}_{q_\phi(Z)}[log\frac{q_\phi(Z)}{q_\phi(z_e)q_\phi(z_f)q_\phi(z_g)} + log\frac{q_\phi(z_e)q_\phi(z_f)q_\phi(z_g)}{p(z_e)p(z_f)p(z_g)} \\
&+ log\frac{p(z_e)p(z_f)p(z_g)}{p(Z)}] \\
&= -E_{q_\phi(z)}[\underbrace{log\frac{q_\phi(Z)}{q_\phi(z_e)q_\phi(z_f)q_\phi(z_g)}}_{Ⓐ} + \underbrace{log\frac{p(z_e)p(z_f)p(z_g)}{p(Z)}}_{Ⓑ}] \\
&\underbrace{- D_{KL}(q_\phi(z_e)||p(z_e)) - D_{KL}(q_\phi(z_f)||p(z_f)) - D_{KL}(q_\phi(z_g)||p(z_g))}_{Ⓒ}
\end{aligned}
$$

We refer to Term Ⓐ as the "Node-Edge Total Correlation" term since it measures the dependence between the three types of latent of graphs z_e, z_f and z_g (group-wise disentanglement). The penalty for this term forces the model to find statistically independent factors for the nodes, the edges and their combinations. A heavier penalty on this term induces better separately and disentangled learning for the graph format data. We refer to Term Ⓒ as the "variable-disentangelment" term which enforces the disentanglement of the variables inside each latent group. This allows us to propose variant model which only penalizes Terms Ⓐ and Ⓒ, shown as the Node-edge Disentangled Total Correlation VAE (NED-TCVAE) in Table 1. In some application cases where only the group-wise disentanglement is needed, and the variable-wise disentanglement in z_e, z_f and z_f is not required. This kind of disentanglement can be referred to as a "Half Correlation Disentanglement" of the graphs, where the penalty for Term Ⓒ is ignored, leading to another variant model NED-HCVAE, as defined in Table.1.

When calculating Term Ⓐ, we utilize the Naïve Monte Carlo approximation based on a mini-batch of samples to underestimate $q_\phi(Z)$, $q_\phi(z_e)$, $q_\phi(z_f)$, and $q_\phi(z_g)$, as described in work proposed by Chen et al. [7].

4.2.3 Generalization of variable-wise disentanglement Ⓒ. To further enforce the variable-wise disentanglement, we generalize Term

Ⓒ by decomposing it to obtain the "Variable Total Correlation"(VTC) terms to largely enforce the variable-wise disentanglement in z_e, z_f and z_g respectively. The following shows the decomposition of $D_{KL}(q_\phi(z_f)||p(z_f))$ in Term Ⓒ as an example:

$$
\begin{aligned}
& - D_{KL}(q_\phi(z_f)||p(z_f)) \\
& = -\mathbb{E}_{q_\phi(z_f)}[log\frac{q_\phi(z_f)}{\prod_d q_\phi(z_f^d)} + log\frac{\prod_d q_\phi(z_f^d)}{\prod_d p(z_f^d)} + log\frac{\prod_d p(z_f^d)}{p(z_f)}] \\
& = -\mathbb{E}_{q_\phi(z_f)} \underbrace{[log\frac{q_\phi(z_f)}{\prod_d q_\phi(z_f^d)}}_{Ⓐ_f} \underbrace{- log\frac{p(z_f)}{\prod_d p(z_f^d)}]}_{Ⓑ_f} - \sum_d \underbrace{D_{KL}(q_\phi(z_f^d)||p(z_f^d))}_{Ⓒ_f}
\end{aligned}
$$

Here, Term $Ⓐ_f$ (referred to as the "Node Total Correlation"(TC)) is the most important term as it helps the model to identify the statistically independent factors in the representation z_f, as proved by Watanabe [46]. Similarly, when decomposing the latent z_e and z_g, we obtain their respective TC terms $Ⓐ_e$ and $Ⓐ_g$. The relevant variant model, labelled $NED-VTCVAE$ in Table. 1, can flexibly enforce both the group-wise disentanglement and the variable-wise disentanglement with pre-defined weights.

4.2.4 Generalization of conditional distribution Term ②. In some cases, we are really only concerned with node attributes or edge attributes, so we need only control either the nodes or edges when generating the graph. Thus, to learn the types of factors involved, we can anchor a single group of latent variable (e.g., z_e), to yield higher mutual information with the observation graphs G.

First, if we decompose Term ② in Eq. 6, we have:

$$
\begin{aligned}
& - log\frac{q_\phi(z_e|E)q_\phi(z_f|F)q_\phi(z_g|E,F)}{q_\phi(z_e)q_\phi(z_f)q_\phi(z_g)} \\
& = \underbrace{log\frac{q_\phi(z_e)}{q_\phi(z_e|E)}}_{②_e^a} + \underbrace{log\frac{q_\phi(z_f)}{q_\phi(z_f|F)}}_{②_f^a} + \underbrace{log\frac{q_\phi(z_g)}{q_\phi(z_g|E,F)}}_{②_g^a}.
\end{aligned} \tag{7}
$$

Since each of the three above terms actually represents mutual information between observations and latent representations, because $-log\frac{q_\phi(z_e)}{q_\phi(z_e|E)} = -log\frac{q_\phi(z_e)q_\phi(E)}{q_\phi(z_e,E)} = log\frac{q_\phi(z_e,E)}{q_\phi(z_e)q_\phi(E)} = I(z_e,E)$. Thus, enforcing them can help ensure the mutual information between each types of latent representations and observed graphs. The extensive model that enforces either of the three terms is named as NED-AnchorVAE in Table 1.

5 EXPERIMENT

This section reports the results of both qualitative and quantitative experiments that are carried out to test the performance of NED-VAE and its extensions on two synthetic and one real-world datasets. All experiments are conducted on a 64-bit machine with an NVIDIA GPU (GTX 1070, 1683 MHz, 16 GB GDDR5) [2].

[2]The code of the model and additional experiment results and details are available at: https://github.com/xguo7/NED-VAE.

5.1 Dataset

5.1.1 Erdos-Renyi Graphs. Erdos-Renyi (ER) graphs are generated based on three types of factor. One is an edge-related factor a that refers to the probability of edge creation in a graph following the rule specified in [13]; the second is a node-related factor b which is the mean of a Gaussian random distribution (the standard is set to 0.1), based on which node attribute $F_{i,1}$ is generated; and the third is a node-edge-joint related factor c defining the function: $F_{i,2} = degree(i) + 10 * c$ (where c is a positive integers chosen from 1 to 10), based on which the second node attribute $F_{i,2}$ is generated. Here, $degree(i)$ refers to the degree of Node i. The dimension of the node attribute and edge attribute is 2 and 1 respectively. A total of 25,000 ER graphs are used for training and 12,500 for testing.

5.1.2 Watts Strogatz Graphs. Watts Strogatz (WR) graphs are also generated based on three types of factor. One is an edge-related factor a that indicates the number of nearest neighbours that each node is joined to in a ring topology [47]; the second is a node-related factor b that refers to the mean of a Gaussian distribution (the standard is set as 0.01) based on which node attribute is generated; and the third factor is a node-edge-joint related factor c that not only defines the probability of rewiring each edge for graph topology but also defines the second node attribute as:$F_{i,2} = degree(i) + 10 * c$. The dimension of the node and edge attribute is 2 and 1 respectively. A total 25,000 WR graphs used for training and 12,500 for testing.

5.1.3 Protein Structure Dataset. Protein structures can be formulated as graph structured data where each amino acid is a node and the geo-spatial distances between them are edges. The molecule simulation process is provided in Appendix. There are two factors involved in generating the contact maps and nodes attributes: simulation time (T) and ionic concentration (C), both of which are edge-related factors. Here, 38 values are used for the ionic concentration (C) and 2,000 values are used for the simulation time (T) to generate the dataset, producing 38,000 samples for training and 38,000 samples for testing.

5.2 Comparison Methods

Since graphVAE [43] is the only existing method that fits the requirement of graph disentanglement (i.e, not only learning the representations of graphs but also generate both edge and node attributes), it is utilized as one comparison method. In addition, to validate the necessities of inferring three types of representations separately, a baseline model called GDVAE is used, which has only one graph encoder for inferring an overall graph representation vector. The proposed model NED-VAE as well as the extensions (except NED-AnchorVAE) in Table 1 are all tested and compared.

5.3 Evaluation Metrics

5.3.1 Qualitative Metrics. As it is important to be able to measure the level of disentanglement achieved by different models, we search to qualitatively demonstrate that our proposed NED-VAE model and its extensions consistently discover more latent factors and disentangles them in a cleaner fashion than the previous models. By learning a latent code representation of a graph, we assume that each variable in the latent code corresponds to a certain factor or property that is used to generate the graphs' edge

and node attributes. Thus, by changing the value of one variable continuously and fixing the remaining variables, we can visualize the corresponding change in the generated graphs.

5.3.2 Quantitative Metrics. We used four quantitative metrics to evaluate the disentanglement of the proposed models. $\beta-M$ [22] and $F-M$ [23] measure disentanglement by examining the accuracy of a classifier that predicts the index of a fixed factor of variation; and the modularity score (mod) [39] measures whether each dimension of z depends on at most a factor describing the maximum variation using their mutual information. Finally, disentanglement metric, DCI metric [12] computes the entropy of the distribution obtained by normalizing the importance of each dimension of the learned representation for predicting the value of a factor of variation. All the implementation details are the same as those in the work proposed by Locatello et al. [34].

5.4 Results for ER dataset

5.4.1 Qualitative Evaluation. For ER graphs visualization, the color of nodes is used to represent the value of the node-related factor b, and graph topology is used to represent the value of the edge-related factor a, and the size of the node is used to represent the value of the edge-node-combined factor c. The values of the latent variables range in $[0, 10]$ and some segments of the generated graphs is shown in Fig. 3. All of the proposed node-edge disentangelment models (NED-) shows the best capabilities in discovering and disentangling all the three types of factors than the graphVAE and the baseline GVAE. For example, the node-related factor travels well with the obvious color ranging, while the discovered node-related factor by graphVAE is not disentangled well because it has some influence on the edges. This is highly due to the powerful co-decoder in the generation of both nodes.

Table 2: Comparison of disentanglement scores of the proposed NED-VAE and its extensions for three datasets.

Dataset	method	β-M(%)	F-M(%)	DCI	Mod
ER	GraphVAE	79.20	33.30	0.33	0.75
	GDVAE	79.20	33.34	0.33	0.74
	NED-VAE	97.20	86.70	0.62	0.95
	NED-IPVAE-I	99.71	**98.84**	**0.73**	0.92
	NED-IPVAE-II	**99.90**	98.70	0.71	0.93
	NED-TCVAE	99.70	88.00	0.64	0.92
	NED-VTCVAE	94.00	59.10	0.63	**0.97**
WS	GraphVAE	73.10	37.87	0.13	0.49
	GDVAE	73.06	37.86	0.13	**0.62**
	NED-VAE	**100.00**	64.96	0.16	0.52
	NED-IPVAE-I	99.30	91.23	0.16	0.50
	NED-IPVAE-II	**100.00**	**97.82**	0.16	0.50
	NED-TCVAE	94.91	64.70	0.16	0.50
	NED-VTCVAE	94.50	49.33	**0.17**	0.51
Protein	GraphVAE	54.00	50.00	0.20	0.61
	GDVAE	54.00	50.00	0.21	0.60
	NED-VAE	**63.42**	61.67	**0.31**	**0.69**
	NED-IPVAE-I	60.46	55.20	**0.31**	0.67
	NED-IPVAE-II	60.00	**64.00**	0.28	0.67
	NED-TCVAE	57.63	50.25	0.25	0.68
	NED-VTCVAE	58.40	50.00	0.24	0.67

5.4.2 Quantitative Evaluation. Four quantitative evaluation metrics are tested on different models and compared in Table 2. The proposed node-edge disentanglement models all shows greater superority than graphVAE and baseline GDVAE. Specifically, NED-IPVAE-II achieves the score of 99.90% in $\beta - M$, outperforming comparison methods by 20% and other proposed extensions by 2.5%. NED-IPVAE-I achieves 98.84% score in $F - M$, outperforming comparison methods by 66.28% and other proposed extensions by 16.9%.The great superiority of the two NED-IP-VAE models is mainly due to their great penalty on the inferred prior term in the objective, which balances the trade-off between the reconstruction error and the disentanglement.

5.5 Results for WR dataset

5.5.1 Qualitative Evaluation. For WR graphs, we utilize the color of node icon to reflect the node-related factor b; and the number of neighboring rings in the graph topology to reflect the edge-related factor a; and the density of graph edges as well as the size of node icon to reflect the edge-node-joint related factor c. The values of the latent variables range in $[0, 10]$ and some segment of the generated graphs to visualize, as shown in Fig. 4. All of the proposed node-edge disentanglement models (NED-) successfully discovers and disentangle at least two of all the three types of factors, while graphVAE fails in discovering both edge-related and node-edge-joint related factors, and GDVAE fails in discovering the node-edge-joint related factors. This validates the necessities of the three types of factor disentanglement and superiority of the proposed architecture which separates the inference of node-related, edge-related and node-edge-related representations.

5.5.2 Quantitative Evaluation. Four quantitative evaluation metrics are tested on WS dataset on different models and compared in Table 2. The proposed node-edge disentanglement models all shows greater superority than graphVAE and baseline GDVAE. Specifically, NED-VAE and NED-IPVAE-I both achieve 100% in $\beta - M$, outperforming comparison methods by 26.9% and other proposed extensions by 3.9%. NED-IPVAE-II achieves 97.8% score in $F - M$, outperforming comparison methods by 60.3% and other proposed extensions by 30.8%.

5.6 Results for Protein Structure dataset

5.6.1 Qualitative Evaluation. We evaluate the control of the factor of simulation time (T) to the generation of edges by visualizing the contact map of the proteins. The value of the relevant latent variables ranges in $[0, 10]$ and some segment of the generated contact maps are shown in Fig. 5. All of the proposed models are capable of finding T factor, while graphVAE shows bad performance in a very slight variation of structure. In addition, qualitative evaluation on protein dataset is also meaningful in analyzing how the proteins folds (reflected in contact maps) as the time flies.

5.6.2 Quantitative Evaluation. Four quantitative evaluation metrics are also tested on protein dataset on different models and compared in Table 2. The proposed node-edge disentanglement models, especiallt NED-VAE all shows greater superiority than graphVAE and baseline GDVAE. Specifically, NED-VAE outperforms the comparison methods by 14.9%, 32.2%, and 13.1% on metrics of $\beta - M$, DCI and Modularity respectively; and outperforms other proposed extensions by 6.8%, 17.2%, and 2.8% on metrics of $\beta - M$, DCI and Modularity respectively.This proves that the proposed NED-VAE still have superiority even when there is only edge-related factor.

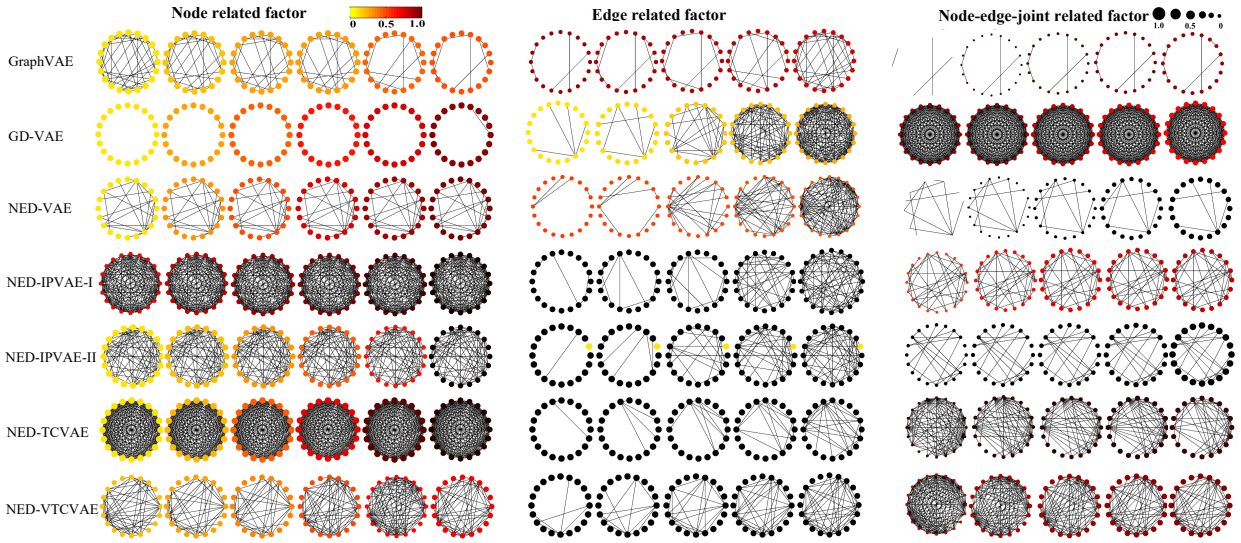

Figure 3: Generated Graphs from different models when the related latent variables range from 0 to 10 for ER graphs: (a) node-related factor which is reflected by color of node icon; (b) edge-related factor which is reflected by edge density (c) node-edge-joint related factor which is reflected by the size of node icon.

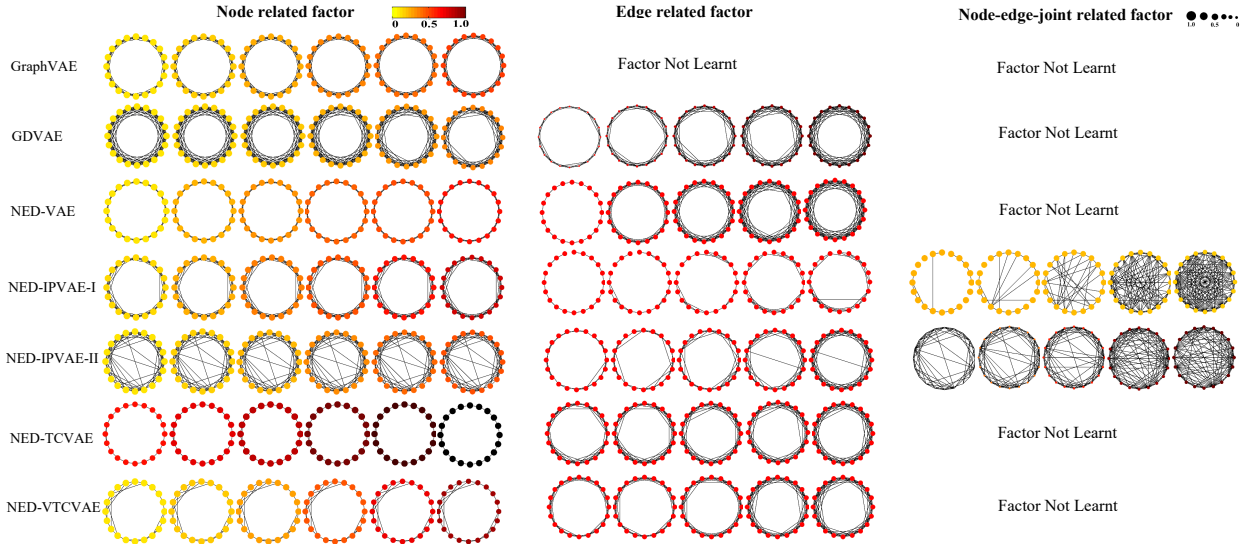

Figure 4: Generated Graphs from different graph disentangled models when the related latent variable value ranges in $[0, 10]$ for WS graphs: (a) node-related factor, which is reflected by color of node icon; (b) edge-related factor which is reflected by number of rings in topology and (c) node-edge-joint related factor which is reflected by edge density and the size of node icon.

6 CONCLUSION

We have introduced NED-VAE, a novel and the first method for disentangling on attributed graphs as far as we know. Moreover, we propose a generic framework of objectives including various derived disentanglement penalties to solve different issues in dealing with graph structured data, such as group-wise and variable-wise disentanglement; multiple trade-off issues between reconstructed edges and nodes, and edge-related, node-related, and node-edge-joint related latent. Finally, we have performed an experimental evaluation of disentangling qualitatively and quantitatively for the proposed NED-VAE and its extensions. The comparison with graphVAE and a baseline model validates the effectiveness of the graph disentanglement architecture and the necessities of separately learning three types of latent representations.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation Grant #1755850, #1841520, #1907805, #1763233, a Jeffress Memorial Trust Award, NVIDIA GPU Grant, and Design Knowledge Company (subcontract number: 10827.002.120.04). Y. Ye's work was partially supported by the NSF Grants IIS-2027127, IIS-1951504, CNS-1940859,

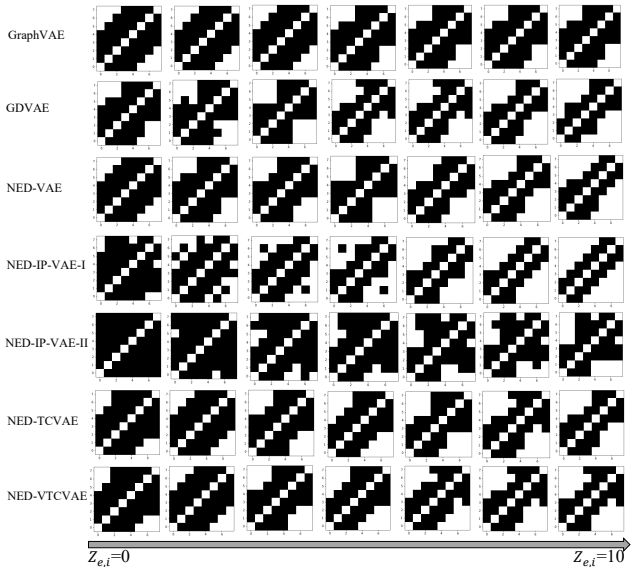

Figure 5: Generated contact maps from different models when one edge-related latent variable ranges from 0 to 10 in protein dataset: more blank spaces indicates higher degree of protein folding

CNS-1946327, CNS-1814825, OAC-1940855, and the NIJ 2018-75-CX-0032. This material is additionally based upon work supported by (while serving at) the NSF. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Romany NN Abskharon, Gabriele Giachin, and et al. 2014. Probing the N-terminal β-sheet conversion in the crystal structure of the human prion protein bound to a nanobody. *Journal of the American Chemical Society* 136, 3 (2014), 937–944.
[2] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2017. Deep variational information bottleneck. *ICLR* (2017).
[3] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. 2019. Unsupervised inductive graph-level representation learning via graph-graph proximity. In *IJCAI*. 1988–1994.
[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.
[5] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. 2018. NetGAN: Generating Graphs via Random Walks. In *ICML*. 609–618.
[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *ICLR* (2013).
[7] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. 2018. Isolating sources of disentanglement in variational autoencoders. In *NeurIPS*. 2610–2620.
[8] Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2020. Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension. *IJCAI* (2020).
[9] Yu Chen, Lingfei Wu, and Mohammed J Zaki. 2020. Reinforcement learning based graph-to-sequence model for natural question generation. *ICLR* (2020).
[10] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. 2018. Syntax-directed variational autoencoder for structured data. *ICLR* (2018).
[11] Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. (2017).
[12] Cian Eastwood and Christopher KI Williams. 2018. A framework for the quantitative evaluation of disentangled representations. (2018).
[13] Paul Erdős and Alfréd Rényi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.
[14] Babak Esmaeili, Hao Wu, Sarthak Jain, and et al. 2019. Structured Disentangled Representations. In *AISTATS*. 2525–2534.
[15] Yuyang Gao, Lingfei Wu, Houman Homayoun, and Liang Zhao. 2019. Dyngraph2seq: Dynamic-graph-to-sequence interpretable learning for health stage prediction in online health forums. *ICDM* (2019).
[16] Ian Goodfellow, Jean Pouget-Abadie, and et al. 2014. Generative adversarial nets. In *NeurIPS*. 2672–2680.
[17] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *IJCNN*, Vol. 2. IEEE, 729–734.
[18] Aditya Grover, Aaron Zweig, and Stefano Ermon. 2019. Graphite: Iterative Generative Modeling of Graphs. In *ICML*. 2434–2444.
[19] Xiaojie Guo, Lingfei Wu, and Liang Zhao. 2018. Deep graph translation. *arXiv preprint arXiv:1805.09980* (2018).
[20] Xiaojie Guo, Liang Zhao, and et al. 2019. Deep Multi-attributed Graph Translation with Node-Edge Co-evolution. In *ICDM*.
[21] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. (2015).
[22] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *ICLR* 2, 5 (2017), 6.
[23] Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. *ICML* (2018).
[24] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[25] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *ICLR* (2016).
[26] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. (2016).
[27] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. 2018. Variational inference of disentangled latent concepts from unlabeled observations. *ICLR* (2018).
[28] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *ICML*. JMLR. org, 1945–1954.
[29] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences* 40 (2017).
[30] Alexander LeClair, Sakib Haque, Linfgei Wu, and Collin McMillan. 2020. Improved code summarization via a graph neural network. *MSR* (2020).
[31] Qingzhe Li, Amir Alipour-Fanid, Martin Slawski, Yanfang Ye, Lingfei Wu, Kai Zeng, and Liang Zhao. 2019. Large-scale Cost-aware Classification Using Feature Computational Dependency Graph. *IEEE Transactions on Knowledge and Data Engineering* (2019).
[32] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018. Learning deep generative models of graphs. (2018).
[33] Yanbei Liu, Xiao Wang, Shu Wu, and Zhitao Xiao. 2019. Independence Promoted Graph Disentangled Networks. (2019).
[34] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2019. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *ICML*. 4114–4124.
[35] Romain Lopez, Jeffrey Regier, Michael I Jordan, and Nir Yosef. 2018. Information constraints on auto-encoding variational bayes. In *NeurIPS*. 6114–6125.
[36] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. In *NeurIPS*. 5712–5723.
[37] Alireza Makhzani and Brendan J Frey. 2017. Pixelgan autoencoders. In *NeurIPS*. 1975–1985.
[38] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML-Volume 32*. II–1278.
[39] Karl Ridgeway and Michael C Mozer. 2018. Learning deep disentangled embeddings with the f-statistic loss. In *NeurIPS*. 185–194.
[40] Bidisha Samanta, Abir De, Niloy Ganguly, and Manuel Gomez-Rodriguez. 2018. Designing random graph models using variational autoencoders with applications to chemical design. (2018).
[41] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.
[42] Kai Shen, Lingfei Wu, Fangli Xu, Siliang Tang, Jun Xiao, and Yueting Zhuang. 2020. Hierarchical Attention Based Spatial-Temporal Graph-to-Sequence Learning for Grounded Video Description. *IJCAI* (2020).
[43] Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN*. Springer, 412–422.
[44] Niklas Stoehr, Marc Brockschmidt, and et al. 2019. Disentangling Interpretable Generative Parameters of Random and Real-World Graphs. (2019).
[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. (2017).
[46] Satosi Watanabe. 1960. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development* 4, 1 (1960), 66–82.
[47] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440.
[48] Jiaxuan You, Rex Ying, and et al. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *ICML*. 5708–5717.
[49] Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2019. Infovae: Information maximizing variational autoencoders. *AAAI* (2019).

A EXPERIMENTAL DETAILS

We use $[0, 1]$ normalised data as targets for the mean of a Bernoulli distribution, using negative cross-entropy for calculating $logp(E|z_e, z_g)$ and Adam optimiser with learning rate 2×10^{-4}. We use the same encoder/decoder architecture for NED-VAE and all of its extensions, as shown in Tables 3. The architecture details of comparison methods graphVAE and baseline GDVAE are shown in Table 5 and Table 4 respectively. We use leaky ReLU (lReLU) non-linearity as the activation function in all models. We train for 500 iterations on two synthetic dataset and 1000 iterations on protein dataset. We use a batch size of 1000 for all data sets. The β in NED-VAE is set to 10. The λ in NED-IPVAE-II and NED-IPVAE-I are set to 10. The β in NED-TCVAE is set to 10. The β, γ_1, γ_2, γ_3 are all set to 10 in NED-VTCVAE.

B DETAILS ABOUT THE SIMULATION PROCESS FOR PROTEIN DATASET

In each of the simulations, we simulate the dynamic folding process of a protein peptide with a sequence AGAAAAGA. This sequence is selected because of the compelling evidence that indicates that an evolutionary N-terminal conserved motif AGAAAAGA plays an important role in causing the cellular prion protein to change to a misfold form and lead to prion diseases [1]. A prion is a type of protein that can trigger normal proteins in the brain to fold abnormally. Prion diseases can affect both humans and animals. The most common form of prion disease that affects humans is Creutzfeldt-Jakob disease, others include mad cow disease, Kuru disease and so on. Indeed, if we refer to the protein data bank (PDB) of all the known protein structures, AGAAAAGA sequence shows very different secondary structures within different protein molecules. Thus it is meaningful to the chemical domain on understanding how its structure change with different environmental conditions. We start by testing how simulation time (T) and ionic concentration (C) play roles in the folding. The initial structure of this 8 amino-acid sequence is a straight chain as the fully unfolded state. We solve the structure in a fully-periodic rectangular water box with a dimension of 50A in each of the (x, y, z) direction. NaCl is added by substituting water molecules to reach a certain concentration that varies from 0 to 3 mol/L. We run 38 simulations for different NaCl concentrations and simulate the folding of the peptide in NPT (constant number of the atom, constant room pressure and temperature 1 atm and 300 K, respectively) ensemble for 20,000,000 dynamic time steps (2 fs time step, 40 ns in total time). We save the fully atomistic structure for every 20 ps, which allows us to further extract the (x, y, z) coordinates of the C_α atoms for each frame, as well as the distance matrix that defines how far is C_α_i from $C\_\alpha_j(i, j = 1...8)$ for that frame. This distance matrix $(D, 8 \times 8)$ is converted to a contact map matrix $(M, 8 \times 8)$ by taking $M_{ij} = 1$ value for a distance smaller than $8A(D_{ij} < 8A)$ and taking $M_{ij} = 0$ value for a distance larger than or equal to $8A(D_{ij} >= 8A)$.

C DETAILS OF NODE, EDGE AND GRAPH ENCODER

The **node encoder** consists of several traditional convolution layers and fully connected layers. First, the node attribute matrix F is convoluted by convolution filters to learn the inherent patterns from all the nodes' attributes. Second, two paths of fully connected layers are used to get the mean μ_f and standard derivation vectors σ_f of the node representations distribution.

The **edge encoder** consists of several edge convolution layers proposed in [19] and fully connected layers. First, the edge attribute tensor E is convoluted by convolution filters in two directions (outgoing and incoming) to learn the hidden relations. Second, the extracted hidden relations are convoluted into a node-level representation and mapped into two paths of fully connected layers to yield the mean μ_e and standard derivation σ_e vectors of the edge representations distribution.

The **graph encoder** consists of several graph convolution layers [25] and fully connected layers. First, the node-level representations are embedded by graph convolution layers. Second, fully connected layers are used to aggregate the learned node representations into a graph-level representation that can be separately mapped into the mean μ_g and standard derivation σ_g vectors of the graph representations.

Table 3: Encoders and decoders architectures (Each layers is expressed in the format as *<filter_size><layer type><Num_channel><Activation function><stride size>*. *FC* refers to the fully connected layers). *c-deconv* and *c-conv* refers to the cross edge deconvolution and convolution respectively.

Node Encoder	Edge encoder	Graph encoder	Node decoder	Edge decoder
Input: $F \in \mathbb{R}^{20\times 2}$	Input: $E \in \mathbb{R}^{20\times 20}$	Input: E,F	Input:$z_n \in \mathbb{R}^3$,$z_g \in \mathbb{R}^3$	Input:$z_e \in \mathbb{R}^3$,$z_g \in \mathbb{R}^3$
2×2 conv.10 ReLU. stride 1	20×1 c-conv.10 ReLU. stride 1	Graph-conv.10 ReLU	FC.320	FC.6
2×2 conv.8 ReLU. stride 1	20×1 c-conv.8 ReLU. stride 1	Graph-conv.8 ReLU	2×2 deconv.8 ReLU. stride 1	20×1 deconv.6 ReLU. stride 1
FC.3	20×1 conv.6 ReLU. stride 1	FC.6	2×2 deconv.10 ReLU. stride 1	20×1 c-deconv.8 ReLU. stride 1
	FC.3	FC.3		20×1 c-deconv.10 ReLU. stride 1

Table 4: Encoders and decoders architectures of Baseline GDVAE (Each layers is expressed in the format as *<filter_size><layer type><Num_channel><Activation function><stride size>*. *FC* refers to the fully connected layers).

Graph encoder	Node decoder	Edge decoder
Input: $F \in \mathbb{R}^{20\times 2}$,$E \in \mathbb{R}^{20\times 20}$	Input$z_g \in \mathbb{R}^9$	Input:$z_g \in \mathbb{R}^9$
Graph-conv.10 ReLU	FC.320	FC.6
Graph-conv.8 ReLU	2×2 deconv.8 ReLU. stride 1	20×1 deconv.6 ReLU. stride 1
FC.6	2×2 deconv.10 ReLU. stride 1	20×1 cross-deconv.8 ReLU. stride 1
FC.3		20×1 cross-deconv.10 ReLU. stride 1

Table 5: Encoders and decoders architectures of graphVAE (Each layers is expressed in the format as *<filter_size><layer type><Num_channel><Activation function><stride size>*. *FC* refers to the fully connected layers)..

Graph encoder	Node decoder	Edge decoder
Input: $F \in \mathbb{R}^{20\times 2}$,$E \in \mathbb{R}^{20\times 20}$	Input$z_g \in \mathbb{R}^9$	Input:$z_g \in \mathbb{R}^9$
Graph-conv.10 ReLU	FC.8	FC.6
Graph-conv.8 ReLU	FC.10	FC.8
FC.9	FC.20×3	FC.20×20