DEEP GRAPH TRANSFORMATION AND INTERPRETATION

by

Xiaojie Guo

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____	Dr. Liang Zhao, Dissertation Director
_____	Dr. Feitian Zhang, Dissertation Director
_____	Dr. Hemant Purohit, Committee Member
_____	Dr. Amarda Shehu, Committee Member
_____	Dr. Lingfei Wu, Committee Member
_____	Dr. Kai Zeng, Committee Member
_____	Dr. Deborah Goodings, Associate Dean
_____	Dr. Kenneth Ball, Dean, The Volgenau School of Engineering

Date: _____ Spring Semester 2021
George Mason University
Fairfax, VA

Deep Graph Transformation and Interpretation

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Xiaojie Guo
Master of Science
Soochow University, 2017
Bachelor of Science
Soochow University, 2014

Director: Dr. Liang Zhao, Dr. Feitian Zhang
Department of Information Science and Technology

Spring Semester 2021
George Mason University
Fairfax, VA

Acknowledgments

First, I would like to express my sincere gratitude to to my advisor, Dr. Liang Zhao, for his support during my Ph.D. study. I have been incredibly fortunate to benefit from his insightful and patient guidance on doing research. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life, especially the hardest moments of my PhD years. I will never forget his great efforts on refining and commenting my research papers and slides again and again. Without his guidance, I would not have made it.

I am very thankful to all my committee members, who have provided not only important guidance for the completion of my dissertation, but also for my research work during our collaborations. Thanks go to Dr. Lingfei Wu for offering his fantastic guidelines as my mentor during my internship at IBM Watson and his strong support in my career development. Thanks go to Dr. Amarda Shehu who provided extensive expertise in protein designing, that opened up new windows for my research. I am impressed and learnt a lot from her high efficiency and deep insights in scientific research. Thanks also go to Dr. Hemant Purohit and Dr. Kai Zeng for providing insightful comments and valuable suggestions from my preliminary proposal, through my research defense, and for my final defense. Their knowledge of the field suggested new research directions that I might not otherwise have recognized. Special thanks go to Dr. Feitian Zhang, who agreed to be my co-advisor and provided numerous help in my graduation and dissertation process.

I would like to express appreciation to my friends in and outside my Laboratory for our unforgettable collaborations: Qingzhe Li, Yuyang Gao, Junxiang Wang, Cuie Yang, Yuanqi Du, Shiyu Wang, Tanmoy Chowdhury, to name but a few-with whom I have shared unique moments throughout this time. Thanks for being around and made my Ph.D. journey enjoyable with many delighted memories. Finally, I would like to dedicate this dissertation with special thanks to my husband, my parents, my sister, and my parents-in-law. These ordinary and great people have provided me with enormous love, courage, comfort, support throughout this process.

Table of Contents

List of Tables

List of Figures

Abstract

DEEP GRAPH TRANSFORMATION AND INTERPRETATION

Xiaojie Guo, PhD

George Mason University, 2021

Dissertation Director: Dr. Liang Zhao

Graphs are universal representations of pairwise information in many domains, e.g., social networks and molecule structures. The problem of deep graph generation which is handled by deep generative models has attracted great attentions in the most recent years. However, it is usually desirable to guide the graph generation process by conditioning on additional information such as data from the different modality, which is also coined as conditional graph generation (i.e., graph transformation). Conditional graph generation can be seen in many real-world applications such as Internet of Things (IoT) confinement and chemical reactant prediction. In addition, the existing graph generation works neglect the entanglement of the encoded latent factors, rendering the generation process non-robust and hardly explainable. Therefore, the general goal of this research is how to develop the graph deep generative models for conditional and interpretable graph generation.

To solve the above problems, there are two main aims to be achieved: (1) The conditional graph generation aims to control the generation process on a specific input graph. One needs to not only learn the transformation mapping in the local information of a graph (i.e., neighborhood pattern of each node), but also in the global property of the whole graph (e.g., node degree distribution or graph density). It is also very important to deal with

more general graph transformation problem for various graph types, such as the multi-attributed graphs, the signed graphs and directed graphs. (2) The interpretation of the graph generation process is also imperative but unexplored. The complex formation process of graphs requires the model to have a sophisticated mechanism for inferring the latent factor that may cause an edge of a specific node and the global properties of the whole graph. This mechanism needs to be differentiable to support end-to-end training and be capable of conducting inductive learning to enable out-of-sample node processing in real-time for real-world deployment.

To achieve the above goals, we first present a novel framework for conditional deep graph topology generation with a graph-translation generative adversarial nets (GT-GAN). GT-GANs learn a conditional generative model, which is a graph translator that transforms an input graph to a target graph. For a more generalized problem, we propose a node-edge co-evolution framework for the multi-attributed graph transformation considering both the directed and sign graphs. Secondly, to interpret the generation process, we first propose a novel Variational Auto-encoder (VAE)-based graph generative model which can learn the disentangled latent representations as well as semantic factors for interpreting the generation process. In addition, to further precisely control the generation process, we propose a property controllable generative model for manipulating the generated graphs with desired properties.

This research spans multiple disciplines and promises to make general contributions in various domains such as deep learning, explainable AI, molecular modeling, and computational biology by putting forth a novel algorithm that can be applied to various real-world network transformation and generation problems, ranging from cyber network transformation to novel molecule structure generation.

Chapter 1: Introduction

Graphs are ubiquitous in the real world, representing objects and their relationships such as social networks, citation network, biology networks, and traffic networks, etc. Graphs are also known to have complicated structures that contain rich underlying values [5]. In the last decade, deep learning has been a "crown jewel" in the domain of artificial intelligence and deep learning, showing superior performance in images [6] and natural language processing [7], etc. The expressive power of deep learning to extract complex patterns underlying from data has been well recognized. As a result, how to utilize deep learning methods for graph data analysis has attracted considerable research attention in the past few years. Tremendous effort has been made towards this area, resulting in a rich literature of related papers and methods to deal with various kinds of graph problems, such as node classification, graph classification, link prediction, community detection and graph generation. The architectures adopted also vary greatly, with operations ranging from supervised to unsupervised, convolutional to recursive.

Graph generation is one of the main topics of graph problems. Graph generation entails modeling and generating real-world graphs, and it has applications in several domains, such as understanding interaction dynamics in social networks [8–10], link prediction [11, 12], and anomaly detection [13]. Owing to its many applications, the development of generative models for graphs has a rich history, resulting in famous models such as random graphs, small-world models, stochastic block models, and Bayesian network models, which all generate graphs based on apriori structural assumptions [14]. These traditional graph generation models [15–17] are engineered towards modeling a pre-selected family of graphs, such as random graphs [18], small world networks [19], and scale-free graphs [15]. However, they have limitations. First, due to their simplicity and hand-crafted nature, these random graph

models generally have limited capacity to model complex dependencies and are only capable of modeling a few statistical properties of graphs. For example, Erdos–Rényi graphs do not have the heavy-tailed degree distribution that is typical of many real-world networks. Second, the utilization of the apriori assumption limits these traditional techniques from exploring more applications in larger scale of domains, where the apriori knowledge of graphs is always not available.

Considering the limitations of the traditional graph generation techniques, a key open challenge in this area is developing methods that can directly learn generative models from an observed set of graphs. Developing generative models that can learn directly from data is an important step towards improving the fidelity of generated graphs, and paves a way for new kinds of applications, such as discovering new graph structures and completing evolving graphs. Recent advances in deep generative models, such as variational autoencoders (VAE) [20] and generative adversarial networks (GAN) [21], have made important progress towards generative modeling for complex domains, such as image and text data. Building on these approaches a number of deep learning models for generating graphs have been proposed [11, 22–24] and result in a new domain: deep generative models for graph generation.

However, there are some limitations or unexplored aspects of existing deep graph generative models on graphs. **(1) Inability to deal with the conditional graph generation problem**. More importantly, most of the existing graph generation models are unconditioned and thus ignore rich input graph information for generating a new graph. In many applications, it is crucial to guide the graph generation process by conditioning on an input graph, which can be cast as a graph transformation learning problem – transforming the input graph to the output graph. **(2) Inability to deal with the mutli-attributed graphs generation**. Multi-attributed graphs are commonly observed in the real-world where nodes and edges both have multiple attributes (i.e. labels, features, and properties). It is very important and challenging to conditionally co-generate both the node and edges attributes in the domain of graph generation. **(3) The generative models usually lack**

interpretation for the generated graphs and the generation process. When we learn the underlying distribution of complex graph data, learning interpretable representations of graphs 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 [25]. In the domain of computer vision, 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 representations are highly independent. Examples of this include variables that only control the size of objects, or their color. However, in the promising domain of graph generation, disentangled enhancement has rarely been well explored yet, but could be highly beneficial for domains such as controlling the generation of protein structures, or designing Internet of Things (IoT) applications. **(4) Lack of abilities to interpret and precisely control the data generation process with the desired properties, especially continuous-valued properties.** Given the interpretation of the generation process, sometimes it is important to utilize the information to manipulate and control the generated data with the desired values of properties. That is, it is important to explicitly explore the mathematical relationship between each property and each latent variable of the data. The first and second sub-problems can be categorized and result in a new research topic, namely conditional deep graph generation, where the task is to learn the underlying mapping from input graphs to target graphs regarding different kinds of graphs (i.e. graph topology, multi-attributed graphs, signed graphs and directed graphs) efficiently and effectively via an end-to-end model, which requires less aprior knowledge about the graphs. The third and forth sub-problems can be categorized into one new research topic, namely interpretation of graph generation and will be handled in two sub-tasks.

Thus, the goal of this research is to explore two novel and important problems in the domain of deep generative model for graph generation, that is, conditional deep graph generation and interpretable deep graph generation as well as their applications in many

real-world tasks. The details of each issue are provided in the following subsections.

1.1 Research Issues

This research aims to explore deep generative models for conditional and interpretable graph generation as well as their applications in many domains. For deep generative models for conditional graph generation, we explore two sub-problems, namely, deep graph topology transformation and multi-attributed graph transformation. For deep interpretable graph generation, we also explore two sub-problem, namely, disentangled deep graph generation and property controllable deep graph generation. The major research issues are stated as follows:

1.1.1 Conditional Deep Graph Topology Generation

Deep graph generation models have achieved great successes recently, among which however, are typically unconditioned generative models that have no control over the target graphs given an input graph. There are various kinds of graph transformation problems regarding the entries to be transformed. One important topic is graph topology transformation where the node set is fixed and the graph topology changes. In this dissertation, we propose a novel Graph-Translation-Generative-Adversarial-Network (GT-GAN) [26] that transforms the input graphs into their target output graphs. GT-GAN consists of a graph translator equipped with innovative graph convolution and deconvolution layers to learn the translation mapping considering both global and local features. A new conditional graph discriminator is proposed to classify the target graphs by conditioning on input graphs while training. Extensive experiments on multiple synthetic and real-world datasets demonstrate that the proposed GT-GAN significantly outperforms other baseline methods in terms of both effectiveness and scalability. For instance, GT-GAN performs at least 10 and 15 time faster than two existing unconditional generation models, i.e., GraphRNN and Random-VAE, respectively, when the size of the graph is around 50.

1.1.2 Conditional Deep Graph Generation for Multi-attributed Graphs

Generalized from image and language transformation, graph transformation aims to generate a graph in the target domain by conditioning an input graph in the source domain. This promising topic has attracted fast-increasing attention recently. Existing works are limited to either merely predicting the node attributes of graphs with fixed topology or predicting only the graph topology without considering node attributes (i.e., graph topology transformation), but cannot simultaneously predict both of them, due to substantial challenges: 1) difficulty in characterizing the interactive, iterative, and asynchronous translation process of both nodes and edges and 2) difficulty in discovering and maintaining the inherent consistency between the node and edge in predicted graphs. These challenges prevent a generic, end-to-end framework for joint node and edge attributes generation, which is a need for real-world applications such as malware confinement in IoT networks and structural-to-functional brain-network transformation. These real-world applications highly depend on hand-crafting and ad-hoc heuristic models, but cannot sufficiently utilize massive historical data. Here, we termed this generic problem "multi-attributed graph transformation" and developed a novel framework, namely, node-edge co-evolution deep graph translator (NEC-DGT), which integrates both node and edge transformation seamlessly. The proposed novel edge translation path is generic and is proven to be a generalization of the existing link prediction models. In addition, a spectral graph regularization based on a novel non-parametric graph Laplacian is proposed in order to learn and maintain the consistency of the predicted nodes and edges. Extensive experiments on both synthetic and real-world application data demonstrated the effectiveness of the proposed method.

1.1.3 Interpretable Deep Graph Generation for Multi-attributed Graphs

For interpretation of the graph generation, one way is to discover the semantic meaning of the representations that are used for graph generation. Disentangled representation learning has recently attracted significant amount of attentions, particularly in the field

of interpretable image representation learning. However, learning the disentangled representations behind a graph remains largely unexplored, especially for the multi-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 based on deep generative models for attributed graphs, namely, node-edge co-disentanglement variational auto-encoder (NED-VAE). 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. Moreover, within each type, individual-factor-wise disentanglement is further enhanced, which is shown to be a generalization of existing framework for images. Qualitative and quantitative experiments on both synthetic and real-world datasets demonstrate the effectiveness of the proposed model and its extensions.

1.1.4 Property Controllable Deep Graph Generation via Graph Manipulation

Deep generative models have made important progress towards modeling complex, high dimensional data. Their usefulness is nevertheless often limited by a lack of control over the generative process or a poor understanding of the latent representation. It is always desirable to manipulate the generated complex data with desired properties. To overcome these issues, attention is now focused on discovering latent variables correlated to the data properties and exploring the mathematical relationship between these properties and latent variables. In this dissertation, we present the Property-controllable VAE (PCVAE), where a new Bayesian model is proposed to inductively bias the latent representation using explicit data properties via novel group-wise and property-wise disentanglement terms. Each data property corresponds seamlessly to a latent variable, by enforcing invertible mutual

dependence between them. This allows us to move along the learned latent dimensions to control specific properties of the generated data with great precision. Quantitative and qualitative evaluations confirm that the PCVAE outperforms the existing models by up to 28% in capturing and 65% in manipulating the desired properties.

1.2 Contribution

The major contributions of the research presented here can be stated as follows:

Conditional Deep Graph Topology Generation

- **Development of an unsupervised framework:** We develop a generic framework called GT-GAN consisting of a novel graph translator and conditional graph discriminator for learning a conditional distribution of target graphs given the input graphs.
- **Proposal of a novel convolution-based graph encoder:** We propose a novel graph encoder consisting of "edge convolution" layers that extract various relations among nodes containing both local and global information, and "node convolution" layers that embed the node representations based on the extracted relations.
- **Proposal of a novel deconvolution-based graph decoder:** We propose a novel graph decoder consisting of the "edge deconvolution" and "node deconvolution" layers, which can decode the node representations first into the latent relations of the target graph and then generate the final target graph. The graph skip-connection is also utilized to map the learned latent relations between the input and target graphs.

Conditional Deep Graph Generation for Multi-attributed Graphs

- **The development of a new framework for multi-attributed graph transformation**. We formulate, for the first time, a multi-attributed graph transformation problem and propose a framework (i.e., NEC-DGT) to tackle this problem. The proposed framework is generic for different applications where both node and edge attributes can change after translation.

- **The proposal of novel and generic edge translation layers and blocks**. A new edge translation path is proposed to translate the edge attributes from the input domain to the output domain. Existing edge translation methods were proven to be special cases of ours, which can handle broad multi-attribute edges and nodes.

- **The proposal of a spectral-based regularization that ensures consistency of the predicted nodes and edges**. In order to discover and maintain the inherent relationships between predicted nodes and edges, a new non-parametric graph Laplacian regularization with a graph frequency regularization is proposed and leveraged.

Interpretable Deep Graph Generation for Multi-attributed Graphs

- **A novel framework is proposed for the disentanglement of attributed graph generation.** In order to jointly disentangle the nodes and edges, we derive a novel objective framework for learning three factors that are exclusive to node patterns, exclusive to edge patterns, and those spanning node-edge-joint patterns. This new framework is demonstrated to be a significant generalization over existing disentanglement frameworks for image generation.

- **A novel architecture is proposed for disentanglement learning on graphs**. Derived from the theoretical objective of our framework, a novel architecture proposed for the representation learning of graphs consists of three sub-encoders (a node encoder, an edge encoder, and a node-edge co-encoder) to learn the three types of representations, along with two novel 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 variational auto-encoder-based models for attributed graphs have been proposed.

Property Controllable Deep Graph Generation via Graph Manipulation

- **A new Bayesian model and inference process is proposed**. A new Bayesian model that inductively biases the latent representation using explicit real data properties is proposed. A variational inference strategy and inference model have been customized to ensure effective Bayesian inference.

- **Two novel regularization terms are proposed to enforce the disentanglement of latent variables**. Group-wise and property-wise disentanglement terms are proposed to enhance the mutual independence among the desired property, their relevant and irrelevant latent variables.

- **A novel invertible property decoder is proposed**. The invertible mutual dependence between property-latent variable pair is achieved by enforcing an invertibility constraint over a residual-based decoder. The invertible property decoder allows the precisely property control on the generated complex data.

1.3 Proposal Organization

The remainder of the research proposal is as follows. Chapter 2 defines the deep graph transformation problem (i.e., conditional deep graph generation), and proposes a new GAN-based model for conditional graph topology generation problem and presents experiments results and discussions. Chapter 3 extends the conditional graph topology generation into conditional multi-attributed graph generation, and describes the proposed supervised framework for dealing with the multi-attributed graph transformation with a node-edge co-evolution process and designs a spectral-based regularization term to enforce the node-edge consistent, where the regularization term is also expanded to handle the sign graphs and directed graphs. Experiments results and discussions are also presented for this work. Chapter 4 introduces the new Node-Edge Disentangled Variational Auto-encoder (NED-VAE) model for interpretable graph generation. NED-VAE is a deep unsupervised generative approach for disentanglement learning on graphs that automatically discover the independent latent factors in both edges and nodes. In Chapter 5, a new model, Property-controllable VAE

(PCVAE) is introduced for interpretable graph generation, where a new Bayesian model is proposed to inductively bias the latent representation using explicit data properties via novel group-wise and property-wise disentanglement terms. Chapter 6 summarizes the work carried out, lists the associated publications, and suggests directions for future research.

Chapter 2: Deep Generative Models for Conditional Graph Topology Generation

2.1 Introduction

In recent years, deep learning on graphs has seen a surge of interest, especially for graph representation and recognition tasks such as node-level classification [27–31] and graph-level classification [32–34]. Because of the successes in graph neural networks, researchers have recently started to explore the use of deep generative models for graph synthesis on practical applications such as designing new chemical molecular structures [23, 35]. This has led to many of the recent advances in deep graph generative models, some of which are domain-dependent models [36, 37] for generating graphs with physical constrains, while others consider the generation of generic graphs [24, 38, 39].

However, there are two main drawbacks of existing deep graph generative models. First, one significant limitation of the previous approaches is that most of these models are only suitable for small graphs with 40 or fewer nodes, which is mainly due to their one-node-per-step generation manner. More importantly, most of the existing graph generation models are unconditioned and thus ignore rich input graph information for generating a new graph.

In many applications, it is crucial to guide the graph generation process by conditioning on an input graph, which can be cast as a graph translation learning problem — translating the input graph to the output graph. Such graph translation can be highly desirable for applications such as disaster management and rare event forecasting, where rare and abnormal graph patterns (e.g., traffic congestion and terrorism events) can be inferred prior to their occurrence even without historical data on the abnormal patterns for this specific graph (e.g., a road network or human contact network). For example, important trade

secrets stored in an enterprise's computer network are extremely hard to defend from theft-of-trade-secret behaviors, which involve highly sophisticated activities in terms of malicious authentication paths over cyber-networks[40]. Malicious detection involves learning from rich historical attack examples, which are not available for all the employee accounts. The generic distribution of theft behaviors from historical attacks on some accounts can be learned and used to synthesize a range of possible malicious authentication graphs for other accounts based on their regular authentication graphs.

For another example, in social networks where people are the nodes and their contacts are the edges, the contact graph among them varies dramatically across different situations. For instance, when people are organizing a riot, it is expected that the contact graph will become denser and several special "hubs" (e.g., key players) may appear. Hence, accurately predicting the contact network in a target situation is highly beneficial to situational awareness and resource allocation.

This type of problem is formulated as *deep graph transformation*, which aims at transforming an input graph in the source domain into the distribution of corresponding output graphs in the target domain based on deep graph neural networks. Few works have been proposed in the domain of graph transformation, however, with their own limitations. Some works propose to only handle some specific tasks without generality, such as molecule reaction prediction or molecule optimization in the domain of biology [39, 41]. Others utilize the sequential generating technique which has difficulty in preserving the global probability of graphs due to the lack of long-term dependency in the sequence [42]. Thus, there are still critical challenges that hurdle the further scientific exploration of the deep graph translation domain: 1) **How to learn one-to-more mapping between the input graph and the target graphs**. Different from the plain graph generation problem, a conditional graph synthesis task is to learn a distribution of target graphs conditioning on the input graph, which aims to capture the underlying implicit properties of the graphs, such as their scale-free characteristic. 2) **How to jointly learn both local and global information for translation**. One needs to learn the translation mapping not only in the local information

(i.e., neighborhood pattern of each node), but also in the global property of the whole graph (e.g., node degree distribution or graph density). 3) **Difficulty in developing a decoder to utilize the extracted patterns of multiple levels in the encoder**. In the deep graph translation process, where multiple levels' graph information is learned and propagated through a top-down and bottom-up path, it is hard to decide which level's extracted patterns in the encoder should be preserved for the generation of target graphs.

To address the aforementioned challenges, we present a novel neural network architecture: Graph-Translation-Generative-Advers arial-Nets (GT-GAN). We first propose a conditional graph GAN architecture that consists of an encoder-decoder translator and a conditional graph discriminator to learn the one-to-more mapping (a conditional distribution) for graph translation. To jointly embed the local and global information, we present a novel graph encoder including both the edge and the node convolution layers. In addition, we further propose a novel graph U-net with graph skips and dedicated graph deconvolution layers including both the edge and the node deconvolution layers. This graph U-net with skips enables the model to learn multiple levels' information in the graph encoder as well as select to utilize valuable multiple levels' information to generate the target graphs in graph decoder. Finally, GT-GAN is scalable with at most quadratic computation and memory consumption in terms of the number of nodes in a graph, making it suitable for at least modest-scale graphs (with hundreds of nodes, compared to the tens of nodes in most of the existing graph generative models).

For this work, we highlight our main contributions as follows:

- We develop a generic framework called GT-GAN consisting of a novel graph translator and conditional graph discriminator for learning a conditional distribution of target graphs given the input graphs.
- We propose a novel graph encoder consisting of "edge convolution" layers that extract various relations among nodes containing both local and global information, and "node convolution" layers that embed the node representations based on the extracted relations.

- We propose a novel graph decoder consisting of the "edge deconvolution" and "node deconvolution" layers, which can decode the node representations first into the latent relations of the target graph and then generate the final target graph. The graph skip-connection is also utilized to map the learned latent relations between the input and target graphs.

- We conduct extensive experiments on both synthetic and real-world datasets on six performance metrics to demonstrate the effectiveness and efficiency of the proposed model.

2.2 Related Works

Graph Neural Networks. The recent surge of research into graph neural networks (GNN) can be generally divided into two categories: graph recurrent networks and graph convolutional networks. Graph recurrent networks originate from early work by [43, 44] and are based on recursive neural networks that have been extended by modern deep learning techniques such as gated recurrent units [27]. The other category, graph convolutional networks, originates from spectral graph convolutional neural networks [45], which were extended by Defferrard et al. [46] using fast localized convolutions, and further approximated by an efficient architecture for a semi-supervised setting proposed by Kipf et al. [28]. Self-attention mechanism and subgraph-level information were also explored later to further improve the representation power of learned node embedding [29, 47, 48]. GNNs are mostly utilized for first learning the latent representation of the nodes or graphs and then apply the learnt representation into many downstream tasks such as node classification, graph classification and link prediction.

Graph Generation. Most of the existing graph generation methods for general graphs has been proposed in the last two years and is based on variational auto-encoders (VAE) [23, 38] and generative adversarial nets (GANs) [49], and others [24, 35]. The current deep

graph generation methods can be categorized into two main branches: (1) Sequential generating [24, 35]: this generates the nodes and edges in a sequential way, one after another. Sequential generating performs the local decisions made in the preceding one in an efficient way with time complexity of only $O(N)$, but it has difficulty in preserving the long-term dependency. Thus, some global properties (e.g., scale-free property) of the graph are hard to include. (2) One-shot generating [23, 38]: this refers to building a probabilistic graph model based on the matrix representation that can generate all nodes and edges in one shot. One-shot generating methods have the capacity of modeling the global property of a graph by generating and refining the whole graph (i.e. nodes and edges) synchronously through several iterations, but most of them are limited to small graphs (i.e. the size of node set is less than 20) since the time complexity is not less than $O(N_4)$. In this chapter, we adopt the one-shot generation process considering that the global and local information are both critical to be captured in the graph translation task. Furthermore, our one-shot generation process are validated to enjoy less complexity (i.e., $O(N2)$) compared to the existing one-shot generation methods. Most of these approaches generate nodes and edges sequentially to form a whole graph, leading to the issues of being sensitive to the generation order and being very time-consuming for large graphs.

Data-Transformation-Involved Graphs. A variety of graph-to-sequence models have been proposed to cope with different tasks including machine translation [50, 51], semantic parsing [52–54], question generation [55], and health status prediction [56]. The sequence-to-graph algorithms are generally popular with those working on NLP methods, including generating AMR structures [57] and dependency graphs [58, 59]. A few very recent attempts have also been made to develop graph-to-graph translation models. Jin et al. [60] proposed a domain-specific graph translation model to deal with the molecular optimization task by utilizing the domain knowledge: a junction tree and molecule graph. Do et al. [41] dealt with the chemical reaction product prediction problem by predicting the reaction sequences based on the input graph of molecules. Sun et al. [42] proposed an RNN-based model for encoding and decoding the directed acyclic graph (converted from regular

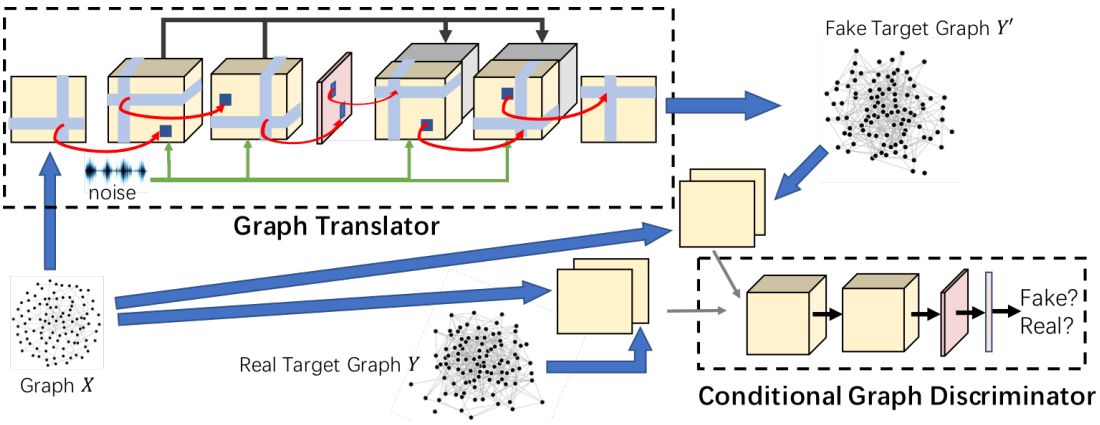


Figure 2.1: GT-GANs consisting of a graph translator and a conditional graph discriminator. A novel graph encoder and decoder are designed for the graph translation problem.

graphs). Guo et al. [61] proposed the NEC-DGT model to deal with the transformation between multi-attributed graphs. However, these methods are trained following the encoder-decoder architecture in a supervised setting instead of learning a distribution of graphs. More importantly, they are difficult to scale to even modest-scale graphs due to their one-node-per-step generation manner.

2.3 The Overall Architecture of GT-GAN

In this section, we first present our formulation of the graph translation problem. We then propose our new GT-GAN model for graph translation and discuss each component in detail in the subsequent sections.

2.3.1 Problem Formulation for Deep Graph Translation

Our goal is to learn an end-to-end translation mapping from an *input graph* to a *target graph*. Let an input graph $G_X = (\mathcal{V}, \mathcal{E}, A, S)$ such that $\mathcal{V}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $A \in \mathbb{R}^{N \times N}$ is an adjacency matrix (binary or weighted), where G_X can be weighted or unweighted, directed or undirected. Let $S \in \mathbb{R}^{N \times F}$ be a node feature matrix

with each row representing a node feature vector S_i. Denote $e_{i,j} \in \mathcal{E}$ as an edge from the node $v_i \in \mathcal{V}$ to $v_j \in \mathcal{V}$; $A_{i,j} \in A$ therefore denotes the corresponding weight of the edge $e_{i,j}$. Similarly, we define a *target graph* $G_Y = (\mathcal{V}', \mathcal{E}', A', S')$ that shares the same node sets and node features with G_X but with different topology and connection weights. Formally, *graph translation* is learning a translator from an input graph $G_X \in \mathcal{G}_X$ with a random noise U to generate a target graph $G_Y \in \mathcal{G}_Y$, where $\mathcal{G}_X$ and $\mathcal{G}_Y$ denote the domains of the input and target graphs, respectively. The translation mapping is denoted as $\mathcal{T} : U, G_X \rightarrow G_Y$.

Note that since our aim is to learn a conditional distribution of the target graphs given an input graph, we can cast the graph translation problem as a conditional graph generation problem, where an input graph can be mapped into any target graph that may have different topologies yet follow the same distribution. In contrast, graph generation, which is designed to learn a distribution of graphs and generate a new graph sample based on this distribution, typically uses a variational autoencoder framework for graph generation. Therefore, the previous graph generation frameworks such as graphVAE [23] and GraphRNN [35] do not directly fit into the "translation" setting.

The Proposed GT-GAN Framework. Fig. 2.1 shows our proposed generic GAN framework for graph translation that consists of a graph translator $\mathcal{T}$ and a conditional graph discriminator $\mathcal{D}$. In this figure, we assume the node feature has only one dimension for simplicity. Since our task is to train a conditional generator with "one-to-many mapping" instead of a deterministic one, the noise U is introduced by the dropout function [62] in each convolution and deconvolution layer, as shown (in green lines) in Fig. 2.1. Our graph translator $\mathcal{T}$ is trained to produce target graphs that cannot be distinguished from "real" ones by our conditional graph discriminator $\mathcal{D}$. Specifically, the generated target graph $G_{Y'} = \mathcal{T}(G_X, U)$ cannot be distinguished from the real one, G_Y, based on the current input graph G_X. $\mathcal{T}$ and $\mathcal{D}$ undergo an adversarial training process based on input and target graphs by solving the following loss function:

$$\mathcal{L}(\mathcal{T}, \mathcal{D}) = \mathbb{E}_{G_X, G_Y}[\log \mathcal{D}(G_Y|G_X)] + \mathbb{E}_{G_X, U}[\log(1 - \mathcal{D}(\mathcal{T}(G_X, U)|G_X))], \tag{2.1}$$

where $\mathcal{T}$ tries to minimize this objective while an adversarial $\mathcal{D}$ tries to maximize it, i.e. $\mathcal{T}^* = \arg\min_{\mathcal{T}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{T}, \mathcal{D})$. We mix the GAN loss with the L1 loss to enforce sparsity similarity, which is also found to be useful in the image translation problem [63],

$$\mathcal{L}_{l1}(\mathcal{T}) = \mathbb{E}_{A, A', U}[\|A' - T(G_X, U)\|_1], \tag{2.2}$$

where $T(G_X, U)$ refers to the adjacent matrix of the generated graph. The training process is a trade-off between $\mathcal{L}_{l1}$ and $\mathcal{L}(\mathcal{T}, \mathcal{D})$, which jointly enforces $\mathcal{T}(G_X, U)$ and G_Y to follow a similar, but not necessarily identical, topological pattern. Specifically, $\mathcal{L}_{l1}$ makes $\mathcal{T}(G_X, U)$ share the same rough outline of sparsity pattern as G_Y, while $\mathcal{L}(\mathcal{T}, \mathcal{D})$ allows $\mathcal{T}(G_X, U)$ to vary to some degree. Thus, the optimal objective $\mathcal{T}^*$ of the translator, which generates graphs that are as "real" as possible, is defined as

$$\mathcal{T}^* = \arg\min_{\mathcal{T}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{T}, \mathcal{D}) + \mathcal{L}_{l1}(\mathcal{D}). \tag{2.3}$$

The graph translator $\mathcal{T}$ is an encoder-decoder architecture, where we propose a new graph encoder to obtain the node representations of the input graph and propose the graph deconvolution with skips to generate the target graph, as shown in Fig. 2.1, which we elaborate in the following sections.

2.3.2 Graph Encoder

The graph encoder aims to learn the representations of nodes based on the node features and graph topology of the input graph. One of the crucial challenges is to learn both local and global information in the graph embedding. For instance, when learning translation between two scale-free graphs, one needs to translate both the local information (i.e., n-hop

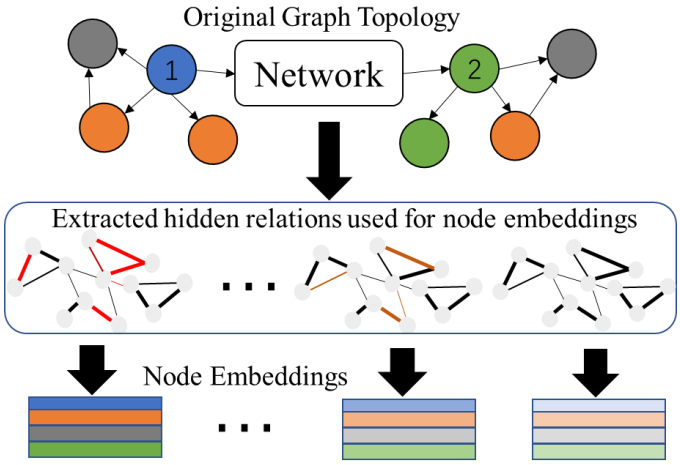


Figure 2.2: The process of proposed graph convolution: the latent edge representations are first extracted based via the edge convolution and the node latent representation are generated based via the node convolution.

neighborhood of each node) and the scale-free property (i.e., node degree distributions of the whole graph) from an input graph to a target graph.

The Proposed Graph Convolution. To learn the local information, the proposed encoder learns each node representation based on its n-hop neighbors. To learn the global information, it learns each node representation by looking for more "virtual neighbors" regarding the latent relations from the aspect of the whole graph. As shown in Fig. 2.2, though Nodes 1 and 2 are located far away in the original network, they have similarities in some properties, such as neighborhood structure and node degree. Thus, these nodes are treated as virtually connected, which is called "virtual neighbors", and they have "hidden relations." We first propose "edge convolution" layers to learn a group of hidden relations from the topology of the input graph, which can include both the n-hop connection relations and those that can deliver structural similarity among nodes. And then the "node convolution" layer is used to embed each node representation by aggregating its "virtual neighbors" that relate to each hidden relation. Fig. 2.3 illustrates the details of these matrix operations involving graph convolution.

Edge Convolution. In each "edge convolution" layer, each node pair's hidden relations

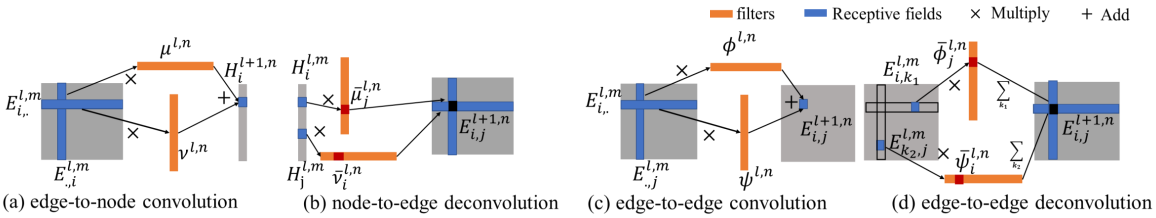


Figure 2.3: Matrix operations for graph convolution and graph deconvolution. In convolution operations, we need to utilize row filter to convolute "incoming" edges and column filter for "outgoing" edges. However, in deconvolution operations, we have to utilize the transposed filters, namely the column filter to decode for "incoming" edges and row filter to decode for "outgoing" edges.

are computed by its adjacent edges or the extracted hidden relations from the last layer. In the directed graph, each node has incoming edge(s) and outgoing edge(s). Thus, there are two learnable parametric vectors ϕ and ψ as convolution filters for two directions to convolute the adjacent edges/relations for each node pair, as shown in Fig 2.3(c). The relation $E_{i,j}^{l+1,n}$ in the nth relation mode of the $(l+1)$th layer is learned by the outgoing edges/relations of node v_i and the incoming edges/relations of node v_j:

$$E_{i,j}^{l+1,n} = \sum_{m=1}^{R_{l+1}} (\sigma(\sum_{k_1=1}^{N} E_{i,k_1}^{l,m} \phi_{k_1}^{l,n}) + \sigma(\sum_{k_2=1}^{N} E_{k_2,j}^{l,m} \psi_{k_2}^{l,n})), \tag{2.4}$$

where $E_{i,j}^{1,1} \equiv A_{i,j}$, $\phi^{l,n} \in \mathbb{R}^{N\times 1}$ refers to the filter vector to be learned, and $\phi_{k_1}^{l,n}$ refers to the element of $\phi^{l,n}$ that is related to node v_{k_1}. R_l refers to the number of hidden relation that will be extracted for the $(l+1)$th layer of the graph encoder. For the indirected graphs, only one direction's convolution is needed.

Node Convolution. After learning the various hidden relations, the "node convolution" layer is used to learn each node's representations by aggregating its "virtual neighbors" in terms of each kind of relation. There are also two vectors of covolution filter $\mu_{l,n}$ and $\nu_{l,n}$ for two directions, as shown by the rectangles in orange in Fig. 2.3. The nth feature vector

of node representation tensor $\bar{H}_i^{l+1,n} \in \mathbb{R}^{1\times F}$ for node v_i is computed as

$$\bar{H}_i^{l+1,n} = \sum_{m=1}^{R_l} (\sigma(\sum_{k_1=1}^{N} E_{i,k_1}^{l,m} \mu_{k_1}^{l,n} S_{k_1}) + \sigma(\sum_{k_2=1}^{N} E_{k_2,i}^{l,m} \nu_{k_2}^{l,n} S_{k_2})), \tag{2.5}$$

where $\bar{H}_i^{l+1} \in \mathbb{R}^{R_{l+1}\times F}$ and R_{l+1} refers to the number of feature vectors in the "node convolution" layer. Here, $\mu^{l,n}, \nu^{l,n} \in \mathbb{R}^{N\times 1}$ refers to the filter vectors for the two directions to be learned, and $\mu_{k_1}^{l,n}$ refers to the element of $\mu^{l,n}$ that is related to node v_{k_1}. $\bar{H}_i^{l+1}$ is then flattened and transformed into a node representation vector $H_i^{l+1} \in \mathbb{R}^{1\times C}$ by a fully connected layer. C is the length of the node representation. Note that our graph encoder is designed for a directed graph, and it is easily generalized to an undirected graph, where the weight vector is shared by both directions. In this proposed architecture, the number of "node convolution layers" is one.

2.3.3 Graph Decoder

The decoder aims to generate the edges of the target graph by taking the extracted latent information of the input graph. It is straightforward to directly use the embedded node representation of the last layer to generate the target graph. However, the extracted information from each layer in the encoder could also be useful for generating the target graph. Thus, we consider all possible information learned in the encoder to be fed into a graph decoder.

The Proposed Graph Deconvolution. Motivated by these observations, we propose a graph U-Net consisting of graph skips and dedicated graph deconvolution layers. The graph deconvolution decodes the single node (or edge) information to yield its incoming and outgoing adjacent edges as a mirrored graph convolution process. In addition, several skips are implemented to map the learned information of each layer in the encoder to mirror the corresponding layers in the decoder. The proposed graph deconvolution technique incorporates both "node deconvolution" and "edge deconvolution" layers.

Similar graph U-Net was proposed by Gao et al [64]. The key difference is that their U-Net is barely a graph embedding method because it uses the old graph topology from the pooling part to embed nodes during the unpooling part. However, our Graph U-Net can not only learn node embedding in the graph encoder but also generate the new graph's topology in the graph decoder, which is necessary for the graph translation problem.

Node Deconvolution. First, the "node deconvolution" layer is used to generate the various hidden relations of the target graph mentioned above based on the learned latent node representations. As shown in Fig. 2.3(b), "node deconvolution" is a reversed process of "node" convolution. It is assumed that each node can influence its relations to other nodes. Then the relation $E_{i,j}^{l+1,n}$ between node v_i and node v_j in the nth relation mode of the $(l+1)$th "node" deconvolution layer in the decoder can be computed as follows:

$$E_{i,j}^{l+1,n} = \sum\nolimits_{m=1}^{C} (\sigma(H_i^{l,m} \bar{\mu}_j^{l,n}) + \sigma(H_j^{l,m} \bar{\nu}_i^{l,n})), \tag{2.6}$$

where $\sigma(H_i^{l,m} \bar{\mu}_j^{l,n})$ means the deconvolution contribution of node v_i to its relation with node v_j made by the mth element of its node representations, and $\bar{\mu}_j^{l,n}$ represents the element of the deconvolution filter vector $\bar{\mu}^{l,n} \in \mathbb{R}^{N \times 1}$ that is related to node v_j.

Edge Deconvolution. We can now recursively apply our proposed "edge deconvolution" layer to decode the latent relation between each pair of nodes from the upper layer to those of the lower layer. As a reversed way of doing "edge" convolution, the relation of each pair of nodes in the lth layer can make contribution to generating itself and its adjacent relations in the $(l+1)$th layer, as shown in Fig. 2.3 (d). Thus, the relation $E_{i,j}^{l,n}$ between node v_i and node v_j in the $(l+1)$th layer is computed as follows:

$$E_{i,j}^{l+1,n} = \sum\nolimits_{m=1}^{R'_l} (\sigma(\bar{\phi}_j^{l,n} \sum_{k_1=1}^{N} E_{i,k_1}^{l,m} S_{k_1}) + \sigma(\bar{\psi}_i^{l,n} \sum_{k_2=1}^{N} E_{k_2,j}^{l,m} S_{k_1})), \tag{2.7}$$

where $\bar{\phi}_j^{l,n} \sum_{k_1=1}^{N} E_{i,k_1}^{l,m}$ is interpreted as the decoded contribution of node i to its relations

with node v_j, and $\bar{\phi}_j^{l,n}$ refers to the element of deconvolution filter vector that is related to node v_j. R'_l refers to the number of relation modes extracted by the lth layer in the graph decoder. The output of the last "edge" deconvolution layer denotes the edges of the target graph.

Skips for Graph Deconvolution. Based on the graph deconvolution above, it is possible to utilize skips to link the extracted latent relation sets of each layer in the graph encoder with those in the graph decoder. Specifically, the output of the lth "edge deconvolution" layer with R_l channels in the decoder is concatenated with the output of the lth "edge convolution" layer with R'_l channels in the encoder to form joint $R_l + R'_l$ channels, which are then input into the $(l+1)th$ deconvolution layer.

2.3.4 Conditional Graph Discriminator

The graph discriminator must distinguish between the "translated" target graph and the "real" ones based on the input graphs, as this helps to train the generator in an adversarial way. Technically, this requires the discriminator to accept two graphs simultaneously as inputs (a target graph and an input graph or a generated graph and an input graph) and classify the two graphs as either related or not. Thus, we propose a *conditional graph discriminator* (CGD) that leverages the same graph convolution layers in the translator for the graph classification, as shown in Fig. 2.1. Specifically, the input and target graphs are both ingested by the CGD and stacked into an $N \times N \times 2$ tensor, which can be considered a 2-channel input. After obtaining the node representations, the graph-level embedding is computed by summing these node embeddings. Finally, a softmax layer is implemented to distinguish the input graph-pair from the real graph or generated graph.

2.3.5 Computational Complexity Analysis

The graph encoder and decoder share the same time complexity. Without loss of generality, we assume all the hidden layers have the same number of feature maps as M. P is the

length of the fully connected layer in the CGD. The worst-case total complexity of GT-GAN (i.e., the dense graph) is now $O(9N^2M^2+3N^2M^2+N^2MP)$, where the first, second, and third terms represent "edge convolutions," "node convolutions," and fully connected layers in the graph discriminator, respectively. Similarly, the total memory consumption for GT-GAN is $O((9NM^2+9N^2M)+(3NM^2+3NM)+(N^2MP+P))$. In practice, many graphs are likely to be sparse; thus, using sparse matrix-vector operations [35] further reduces the computational and memory costs to $O(N)$, which paves the way toward modest scale graphs with hundreds or thousands of nodes, compared to most of the existing graph generation methods, which often have $O(N^3)$ or even $O(N^4)$ computational costs.

2.3.6 Indirect Evaluation Metrics

To further evaluate the generated graphs, we propose a novel indirect evaluation schema. It is straightforward to directly evaluate whether the generated graphs follow the same underlying patterns as the target graphs by simply calculating some predefined graph property metrics, such as graph kernel similarity and degree distribution similarity. However, sometimes the underlying patterns are more complex to define and measure. Thus, we design an indirect evaluation schema inspired from a real-world classification problem: label imbalance issues. For example, we may want to build a classifier to determine whether an authentication graph of a user is malicious (positive) or normal (negative), but this user has few malicious records. For this difficult task, the graphs (i.e., malicious graphs) generated by GT-GAN, which has been trained on other users' records, can be utilized as positive samples to train the classifier.

Specifically, when evaluating, the test set is further split evenly into two subsets. The first subset is used to train a graph classifier, as proposed by Nikolent et al. [65], using only the normal graphs plus the generated malicious graphs. The second subset, which contains both the normal and real malicious graphs, can then be used to validate the trained classifier. In addition, a "gold standard" classifier trained on both normal and *real* malicious graphs acts as the "best-possible performer" and is used to evaluate all the different generative

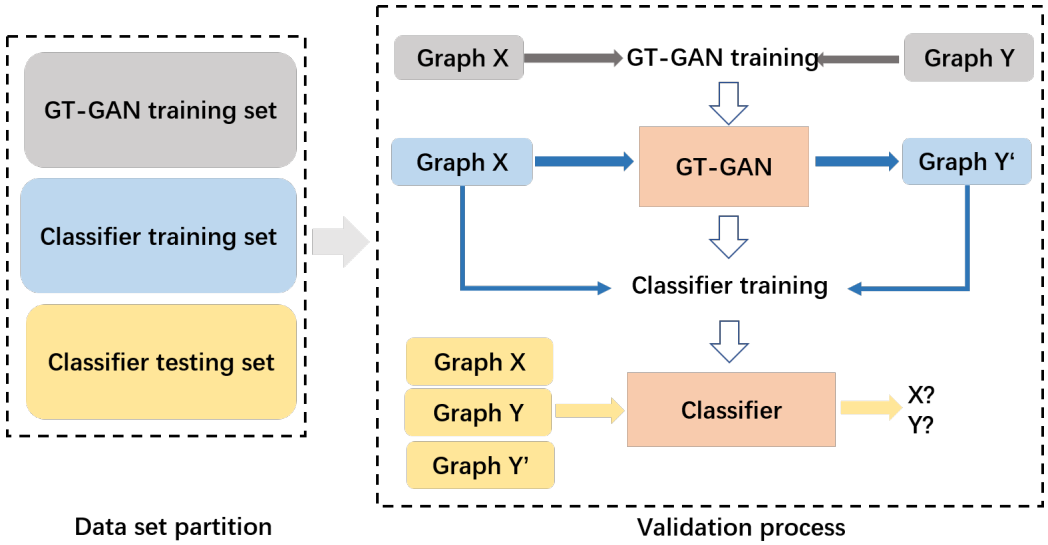


Figure 2.4: Flow chart of validation: Graphs X and Y denote the real input and target graphs. Graph Y' denote to the generated target graphs given the input condition X. Graphs X and Y' are used for training the classifier. The classification results on graphs X and Y are evaluation metrics of GT-GAN. The classification results on graphs X and Y' are "gold standards".

models to judge how "real" the graphs they generate are. The detailed evaluation flowchart is provided in Figure. 2.4

2.4 Experiments

This section reports the results of extensive experiments and ablation studies carried out to test the performance of GT-GAN on two synthetic and two real-world datasets. All experiments were conducted on a 64-bit machine with an NVIDIA GPU (GTX 1070, 1683 MHz, 8 GB GDDR5).

2.4.1 Datasets

In this section, we describe the experimental settings and rules for generating synthetic input-target graph pairs, as well as the process of collecting the real-world graphs for each dataset.

Synthetic Datasets. Synthetic datasets were used to validate the performance of the proposed GT-GAN. Each input graph is generated as a directed scale-free network, whose degree distribution follows the power-law property [66]. A node will be selected as the target node with a probability proportional to its in-degree, which will be linked to a new source node with probability of 0.41. Similarly, a node will be selected as a source node with probability proportional to its out-degree, which will be linked to a new target node with a probability of 0.54. Then, in the target graph, each weight between two connected nodes will be added to m, where m could be any value larger than 1. Thus, both the input and target graphs are scale-free graphs. Each group has five subsets with different graph sizes (number of nodes): 10, 20, 50, 100, and 150. Each subset consists of 5,000 input-target graph pairs: 2,500 pairs were used for training and the remaining 2,500 for testing.

User Authentication Dataset. This dataset includes the authentication activities of 97 users on their accessible computers and servers in an enterprise computer network [67]. Each user account generates a log file recording the computer accessing history, which could be formulated as a directed weighted graph called an authentication graph, where nodes represent computers and the directed edges' weights represent authentication activities with certain frequencies. Here, each authentication graph records user behavior for thirty minutes. The goal of this application was to forecast future potential malicious authentication graphs given the user's normal authentication graph. In total, there are 78 pairs of graphs (malicious and normal behavior) of graph size 50 and 315 pairs of graphs of graph size 300 from 97 users in two subsets. We performed a 2-fold cross-validation and 3-fold cross-validation, respectively, for the two subsets.

Internet of Things (IoT) Dataset. This application focused on IoT network malware confinement prediction (predicting optimal network operation given a compromised one), which is also tested in the work proposed by Guo et al. [61]. There are three subsets of graph pairs with different sizes (20, 40, and 60), where the nodes represent devices and the node attributes indicate the compromised status of the nodes. The weights of the edges represent the distance between two devices. There are 334 pairs of input (compromised

IoT) and target graphs (optimal IoT) in each subset, and each is divided into two parts for the 2-fold cross-validation.

Real-world HCP Dataset: The human connectome project (HCP) dataset is used for evaluating the signed graph translation model. Brain network prediction, such as prediction of functional connectivity based on structural connectivity, is a very critical task in neuroscience. The goal is to learn the mapping from the resting-state functional connectivity into task-specific functional connectivity in the human brain. In this dataset, the source and the target graphs respectively reflect the structural connectivity (SC) and the functional connectivity (FC) of the same subject's brain network. In particular, both types of connectivity are processed from the magnetic resonance imaging (MRI) data obtained from the HCP dataset [68]. By following the preprocessing procedure in [69], the SC data is constructed by applying probabilistic tracking on the diffusion MRI data using the Probtrackx tool from the FMRIB Software Library [70] with 68 predefined regions of interest (ROIs). Then, the edge attributes of FC are defined as the Pearson's correlation between two ROIs' blood oxygen level-dependent time obtained from the resting-state functional MRI data, where both "full" normalized temporal correlation and partial regularized temporal[1] between every node time-series and every other are considered. The node attributes refer to the index of each node by a one-hot vector. FC data are collected regarding 5 different human brain states, namely, resting, emotion, gambling, language and motoring. Thus, there are 5 subusets for both full normalized and partial regularized dataset. In total, each subset has 823 pairs of SC and FC samples.

2.4.2 Baseline Methods

We compare our GT-GAN against five state-of-the-art graph generation methods: 1) GraphRNN [35] is a new graph generation method based on sequential generation with the LSTM model; 2) GraphVAE [23] is a probability-based graph generation method for small graphs; 3) GraphGMG [24] is a framework based upon graph neural networks for small single graphs;

[1]To estimate the partial correlation coefficients, a small amount of L2 regularization is applied

4) RandomVAE [38] was described in related works; and 5) S-Generator is the part of our full model GT-GAN that essentially is a graph translator with L1 loss but no discriminator. We propose this S-Generator model in order to evaluate the necessity of the proposed GT-GAN framework to learn the one-to-many mappings. All the comparison methods were trained on the malicious graphs without conditioning on the input graphs due to the models' inherent capability limitations. The datasets were assigned to each comparison model for the experiment based on their scalability in terms of graph size.

2.4.3 Evaluation Results on Synthetic Datasets

Statistics-based Evaluation. To evaluate the similarity between the generated and real target graphs, we selected four performance metrics: 1) one metric is distance between a generated and real graph in terms of Closeness centrality (C-dist) [71] and another is similarity score based on the graph kernels of the Weisfeiler Lehman kernel(wl-sim) [72]; 2) two metrics are used to evaluate the node degree distribution correlation between the generated and real target graphs by Jensen-Shannon distances (JS) and the Wasserstein Distances (WD). As shown in Table 2.1 (left), our GT-GAN consistently outperforms all other baselines by a large margin, especially when the graph size becomes large (i.e., it outperforms the other methods by 34.6% when the size is 150). S-Generator is generally the second best method in terms of these six evaluation metrics, highlighting the effectiveness of our proposed graph encoder and decoder.

Classifier-based Evaluation. To further evaluate the generated graphs, we propose a novel classifier-based evaluation schema. We assume that the classifier that is capable of recognizing the real target graphs can also be succeed in recognizing the generated target graphs. Table 2.2 shows the average results of graph classifiers: Precision, Recall, AUC, and F1-measure for different methods. The 'gold standard' is a classifier trained on real input and target graphs, which is used to compared with those trained on the graphs generated by different models. For small graphs (e.g., fewer than 10), the power-law property of scale-free networks is less obvious compared to larger size graphs, which may explain why the tasks on

Table 2.1: Statistics-based evaluation results for the scale-free graphs

Size	Methods	JS	WD	C-dist	wl-sim
10	GraphRNN	0.47	1.64	0.5319	0.2470
	GraphVAE	0.67	2.85	0.6664	0.3723
	GraphGMG	0.43	1.69	0.4763	0.3701
	S-Generator	**0.35**	0.80	0.2465	0.4185
	GT-GAN	**0.35**	**0.77**	**0.2379**	**0.4195**
20	GraphRNN	0.50	1.44	0.6087	0.2652
	S-Generator	0.36	0.67	**0.1367**	0.4665
	GT-GAN	**0.35**	**0.66**	0.1894	**0.4681**
50	GraphRNN	0.49	0.94	0.6129	0.2613
	S-Generator	**0.31**	0.90	0.2100	0.3400
	GT-GAN	0.43	**0.89**	**0.2087**	**0.4078**
100	GraphRNN	0.48	0.90	0.6519	0.2713
	S-Generator	**0.14**	**0.30**	**0.1501**	0.3522
	GT-GAN	0.15	0.31	0.2087	**0.4078**
150	GraphRNN	0.42	0.95	0.6266	0.2891
	S-Generator	0.08	0.29	**0.1101**	0.3493
	GT-GAN	**0.07**	**0.27**	0.2105	**0.3926**

smaller scale-free graphs are more difficult. However, when the size of graphs increases, GT-GAN becomes closer to the performance of the "Gold Standard" with average differences of 10% and 9% on F1 accordingly on two sub-datasets, and it significantly outperforms the other methods by large margins up to 51% and 19% on F1, respectively.

To verify whether GT-GAN indeed discovers the underlying ground-truth graph distributions of the target graphs, we draw the node degree distribution curve for three pairs of generated and real target graphs by GT-GAN, as an example shown in Fig. 2.5. In addition, Fig. 2.6 shows 18 examples of the node degree distribution curve in generated and real target graphs for scale free dataset from size 50 to 150. As shown in Fig. 2.6, the node degree distribution of the generated graphs closely follow the node degree distribution of the real target graphs. It is also interesting to observe that the larger the size of the graphs, the more similar of the generated graphs to the real target graphs. This is because the power-law property of the node degree distribution is more obvious and thus is more easier to capture when there are more nodes in the graphs.

Table 2.2: classifier-based evaluation results for the scale-free graphs

Size	Method	P	R	AUC	F1
10	GraphRNN	0.31	0.11	0.49	0.16
	GraphVAE	0.75	0.23	**0.65**	0.35
	GraphGMG	0.42	0.12	0.49	0.18
	S-Generator	0.46	**0.83**	0.43	0.59
	GT-GAN	**1.00**	0.50	0.52	**0.67**
	Gold Standard	*1.00*	*0.74*	*0.82*	*0.77*
20	GraphRNN	0.67	0.12	0.50	0.21
	S-Generator	0.50	**1.00**	0.50	0.67
	GT-GAN	**1.00**	0.50	0.50	**0.67**
	Gold Standard	*1.00*	*0.67*	*0.72*	*0.71*
50	RandomVAE	0.89	0.67	0.84	0.76
	GraphRNN	0.52	0.53	0.70	0.52
	S-Generator	0.50	**1.00**	0.37	0.67
	GT-GAN	**0.93**	0.82	**0.94**	**0.87**
	Gold Standard	*0.94*	*0.90*	*0.97*	*0.91*
100	GraphRNN	0.61	0.65	0.67	0.60
	S-Generator	0.50	**1.00**	0.50	0.67
	GT-GAN	**0.72**	0.69	**0.68**	**0.70**
	Gold Standard	*0.99*	*0.61*	*0.81*	*0.75*
150	GraphRNN	0.73	**0.92**	0.92	0.81
	S-Generator	0.90	0.50	0.50	0.67
	GT-GAN	**0.94**	0.79	**0.96**	**0.86**
	Gold Standard	*0.99*	*0.93*	*0.96*	*0.95*

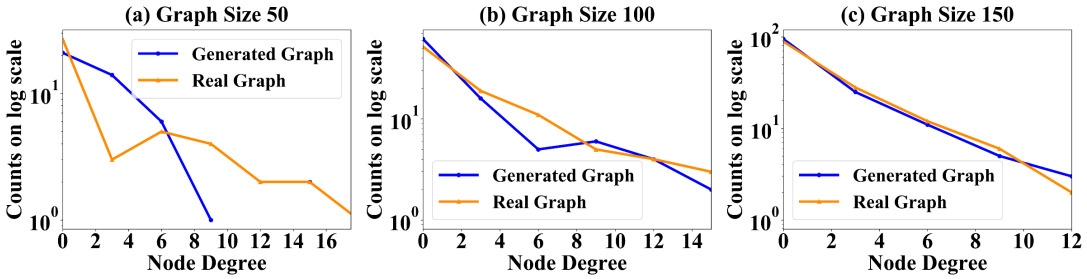


Figure 2.5: Examples of node degree distributions of generated and target graphs for scale-free graphs

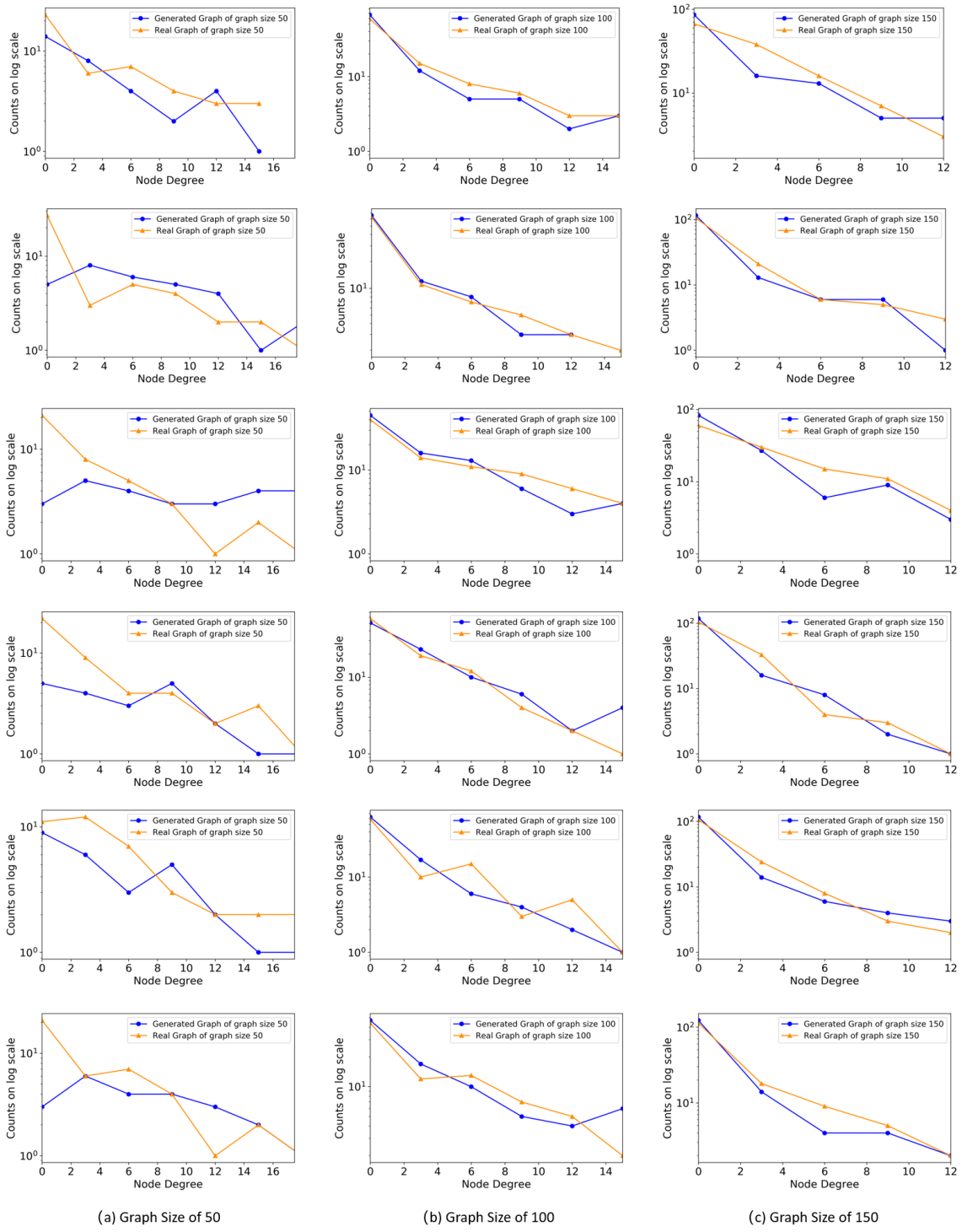


Figure 2.6: Examples of node degree distribution for generated graphs and real graphs for different graph size

2.4.4 Evaluation Results on User Authentication Datasets

Classifier-based Evaluation. As shown in Table 2.3, classifiers trained by the graphs generated by GT-GAN can classify normal and hacked behaviors effectively with AUC above 0.78, which is well above the 0.5 obtained using a random model. GT-GAN significantly outperforms other methods by around 25%, 16%, 24.5%, and 22.1%, respectively, on the four metrics: precision (P), recall (R), AUC, and F1-score for the trained classifier. GT-GAN performs consistently better than other methods when the graph size increases from 50 to 300.

Table 2.3: Classifier-based Results for user authentication datasets

Size	Method	P	R	AUC	F1-score
50	GraphRNN	0.34	0.36	0.50	0.36
	S-Generator	0.72	0.61	0.74	0.66
	GT-GAN	**0.79**	**0.68**	**0.78**	**0.73**
	Gold Standard	*0.97*	*0.97*	*0.97*	*0.97*
300	S-Generator	0.77	0.58	0.62	0.66
	GT-GAN	**0.84**	**0.66**	**0.79**	**0.74**
	Gold Standard	*0.98*	*0.96*	*0.97*	*0.97*

Statistics-based Evaluation. In addition, GT-GAN clearly outperformed the S-Genertor in statistics-based evaluation setting. For example, there are three direct evaluation metrics (i.e., En-dist, C-dist and wl-sim) mentioned above that are also tested, and the results can be found in Table 2.4. The proposed GT-GAN has the best performance in all three aspects both in small scale graphs and large scale graphs. Specifically, for the graphs with size 50, GT-GAN significantly outperforms S-Generator by around 2.2%, 31.2%, and 3.2%, respectively, on the three metrics. For the graphs with size 300, GT-GAN significantly outperforms S-Generator by around 35.2%, 16.7%, and 0.2%, respectively, on the three metrics. This confirms that using a translator alone to learn a deterministic output given an input graph is not sufficient to capture the generic distribution of the target graphs.

Case study on user authentication dataset. Fig. 2.7 shows two examples of users

Table 2.4: Statistics-based evaluation for user authentication datasets

Size	Method	C-dist	wl-sim
50	GraphRNN	0.7456	0.6265
	S-Generator	0.4414	0.9137
	GT-GAN	**0.1924**	**0.9439**
300	S-Generator	0.0702	0.9846
	GT-GAN	**0.0681**	**0.9864**

with a regular activity graph, real malicious activity graph, and malicious activity graph generated by our GT-GAN, from left to right. Only those of edges with a difference among them are drawn for legibility. For User 049, it can be seen that the hacker performed attacks on Computer 192, which has been successfully simulated by our GT-GAN. In addition, GT-GAN also correctly identified that Computer 192 is the end node (i.e., with only incoming edges) in this attack. This is because GT-GAN can learn not only the global hacking patterns (i.e., graph density, modularity) but also the local properties for specific nodes (i.e., computers). GT-GAN even successfully predicted that the hacker connect from Computers 0 and 1, with Computers 7 and 14 as false alarms. For User 006, the red team attackers make more connections on Node 192 compared to user's regular activity, as marked in the red rectangle. GT-GAN leans how to choose the Node 192 and it also generated more connections in Node 36.

2.4.5 Evaluation Results on IoT dataset

Table 2.5 compared the performance of GT-GAN and other comparison methods for the IoT dataset by examining the edges of the generated and real target graphs for four metrics: coefficient of determination score(R2) and Accuray(ACC) for the correct existence of edges among all the pairs of nodes, as well as two statistic-based metrics mentioned above. The results show that GT-GAN performed almost the best for all three subsets. Due to the L1-loss required to maintain topology pattern similarity, GT-GAN outperformed the comparison methods with around 8%, 26%, and 40% superiority in ACC for the three subsets

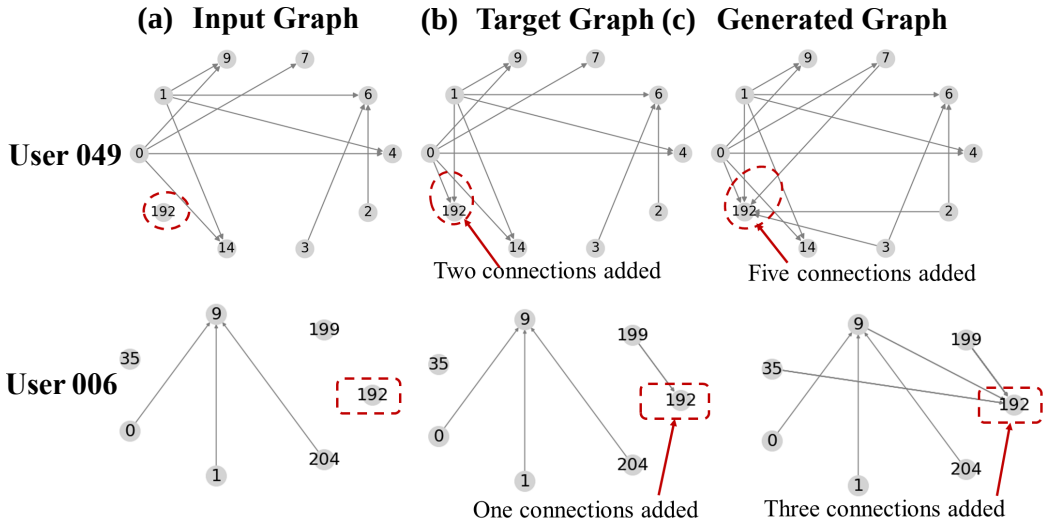


Figure 2.7: Regular graphs, malicious graphs, and generated graphs of User 049 and User 006 for author authentication graph synthesis task

with the size 20, 40 and 60, respectively. As the graph size increases, the properties of the graphs are clearer and the superiority of the proposed GT-GAN model is more obvious. For example, the proposed GT-GAN model outperforms the GraphRNN and GraphVAE by 67.6% and 21.5% on average on the metrics of C-dist and wl-sim, respectively, when the graph size is 40, and by 75.4% and 12.2% when the graph size is 60.

Table 2.5: Results for the IOT datasets

Size	Method	R2	Acc(%)	C-dist	wl-sim
20	GraphRNN	0.16	83.97	0.6913	0.3334
	GraphVAE	0.39	81.19	**0.3283**	0.3471
	GT-GAN	**0.67**	**92.00**	0.4653	**0.3536**
40	GraphRNN	0.44	70.54	0.6908	0.3333
	GraphVAE	**0.73**	66.60	0.4683	0.2603
	GT-GAN	0.69	**93.94**	**0.3378**	**0.3758**
60	GraphRNN	0.52	61.07	0.6914	0.3333
	GraphVAE	0.00	50.64	0.3896	0.3510
	GT-GAN	**0.62**	**94.63**	**0.3051**	**0.3899**

2.4.6 Evaluation Results for HCP Datasets

We consider four classic brain network prediction methods and one graph-based method that use SC to predict FC [73, 74] as the comparison methods for this experiment in the domain of brain network science. Abdelnour et al. [74] considered the graph spectral transformation kernels by assuming that SC and FC share the identical eigenvectors on their Laplacians. Another two methods directly considers the graph translation between SC and FC. The goal is to predict the FC given the SC when the brain is doing different tasks. The Pearson coefficient used as metric by comparing the predicted graphs with the empirical target graphs, which is widely used in the domain of brain network [75, 76]. As shown in Table 3.5, the proposed GT-GAN achieves the highest Pearson coefficient on all the five sub-datasets regarding both the full correlation and L2 correlation, with superiority of about 52.38% and 32.5% averagely. The great superiority of the proposed GT-GAN over the typical methods in the domain of brain networks validates the power of deep learning-based models in handling complex real-world applications.

Table 2.6: Pearson correlation between the predicted graph and empirical graph on HCP datasets (Res for Resting, Emo for Emotion, Gam for Gambling, Lang for Language)

Method	Full Normalized Network				L2 Regularized Network			
	Res	Emo	Gam	Lang	Res	Emo	Gam	Lang
Ganlan et al. [73]	0.23	0.14	0.14	0.14	0.41	0.42	0.44	0.44
Abdelnour et al. [77]	0.23	0.14	0.14	0.15	0.41	0.42	0.43	0.44
Meier et al. [78]	0.26	0.16	0.15	0.16	0.43	0.44	0.46	0.46
Abdelnour et al. [74]	0.23	0.14	0.14	0.15	0.40	0.41	0.43	0.43
Guo et al. [61]	0.13	0.34	0.04	0.17	0.43	0.43	0.46	0.45
GT-GAN	**0.45**	**0.34**	**0.34**	**0.35**	**0.66**	**0.62**	**0.63**	**0.64**

2.4.7 Model Ablation Study

To further validate the superiority of the proposed graph convolution and deconvolution layers, an ablation experiment was conducted. The graph encoder was replaced by GCN [28],

Table 2.7: Ablation study of graph encoder and decoder in GT-GAN (Scale-III denotes to the scale free datasets with graph size of 50; Auth-I denotes to the user authentication dataset with graph size of 50; IoT-III donotes to the IoT dataset with graph size of 60).

Dataset	Method	JS	WD	C-dist	wl-sim
Scale-III	GCN+decoder	0.18	18.84	0.6751	0.4031
	DCNN+decoder	0.65	0.77	0.6745	0.4032
	Unet+decoder	0.69	5.77	0.6496	0.4040
	Encoder+VGAE	0.31	43.78	0.2559	0.4003
	GT-GAN	**0.15**	**0.31**	**0.2087**	**0.4078**
		P	F1	C-dist	wl-sim
Auth-I	GCN+decoder	0.31	0.33	0.7494	0.6632
	DCNN+decoder	0.59	0.57	0.3349	0.6851
	Unet+decoder	0.41	0.49	0.6859	0.9239
	Encoder+VGAE	0.49	0.47	0.3129	0.6111
	GT-GAN	**0.79**	**0.73**	**0.1924**	**0.9439**
		R2	Acc(%)	C-dist	wl-sim
IoT-III	GCN+decoder	0.46	92.69	0.4349	0.3304
	DCNN+decoder	0.52	93.26	0.3217	0.3292
	Unet+decoder	0.45	92.46	**0.2771**	0.3310
	Encoder+VGAE	0.12	88.14	0.4876	0.3333
	GT-GAN	**0.62**	**94.63**	0.3051	**0.3899**

DCNN [79] and Graph U-NET [64]. The graph decoder was replaced by the decoder in VGAE [11]. There were thus three method combinations for comparison.

Table. 2.7 shows parts of the results in the ablation study of the proposed encoder and decoder on part of the scale-free graphs with size 50 (Scale-III), user authentication with graph size 50 (Auth). The encoder of GT-GAN outperformed both the GCN- and DCNN-based encoders by a large margin on these datasets, especially for the real-world datasets, where the edges of the graphs can have a very complex meaning. For example, on Auth-I, GT-GAN performed 41% and 38% better on average than other encoders in terms of precision and F1-scores, respectively. Similar results can also be found for the proposed decoder, which demonstrates that the proposed decoder in GT-GAN was both effective and irreplaceable for graph generation.

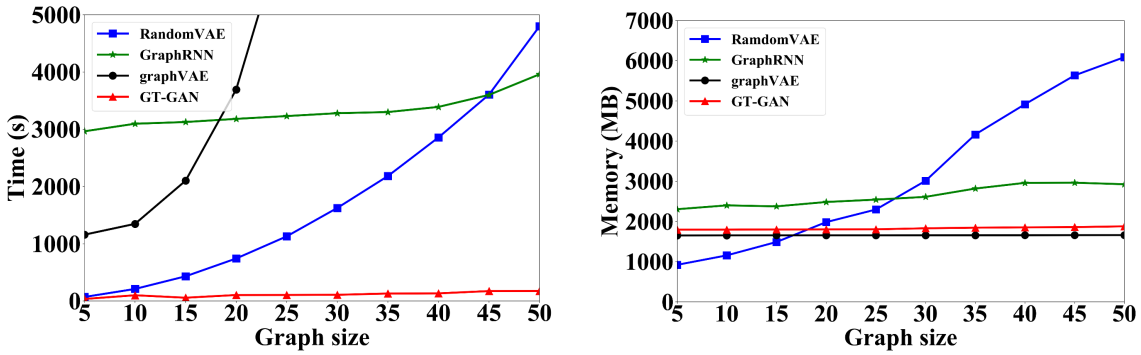


Figure 2.8: Scalability plots for memory (upper) and time cost (bottom) of GT-GAN, RandomVAE, GraphVAE, and GraphRNN

2.4.8 Model Scalability Analysis

We compare the scalability of GT-GAN against three graph generation methods as shown in Fig. 2.8. Our GT-GAN model significantly outperforms other state-of-the-art baselines in terms of both computational time and memory consumption. As the graph size increases up to 50, both the computational time and memory consumption of GT-GAN remain almost constant. In contrast, the runtime and memory consumption of RandomVAE and the runtime of GraphVAE increase super-linearly as the graph size increases, making it hard to scale even to a graph size of 50. Though the runtime and memory consumption of GraphRNN also increase slightly as the graph size increases, our GT-GAN achieves around a ten-time speedup while requiring almost half of the memory as compared to GraphRNN. Moreover, graphRNN requires a larger base runing time (i.e.,3000s) for even a graph with 5 nodes. This is because graphRNN generates the graph in a sequential style, where nodes and edges are generated one by one.

2.5 Conclusion

This chapter focuses on a new problem: conditional deep graph topology generation, which has been never explored before. To deal with it, we propose a novel GT-GAN model

that transforms an input graph to a target graph. To learn both global and local mapping between graphs, a new graph encoder-decoder model is proposed while preserving the graph patterns in various scales. Extensive experiments were conducted on synthetic and real-world datasets to compare with the state-of-the-art graph generation models. Experimental results show that our GT-GAN can discover the ground-truth translation rules, and significantly outperforms other baselines in terms of both effectiveness and scalability. Considering that both nodes and topology are important in formalizing a graph, the next step is to explore the conditional graph generation for multi-attributed graphs where both nodes and edges can change during transformation process.

Chapter 3: Conditional Deep Multi-attributed Graph Generation with Node-Edge Co-evolution

3.1 Introduction

As introduced in the last chapter, many problems regarding structured predictions are encountered in the process of "translating" an input data (e.g., images, texts) into a corresponding output data, which is to learn a translation mapping from the input domain to the target domain. For example, many problems in image processing and computer vision can be seen as a "translation" from an input image into a corresponding output image. Similar applications can also be found in language translation [52, 53], where sentences (sequences of words) in one language are translated into corresponding sentences in another language. Such generic translation problem, which is important yet has been extremely difficult in nature, has attracted rapidly-increasing attention in recent years. The conventional data translation problem typically considers the data under special topology. For example, an image is a type of grid where each pixel is a node and each node has connections to its spatial neighbors. Texts are typically considered as sequences where each node is a word and an edge exists between two contextual words. Both grids and sequences are special types of graphs. In many practical applications, it is required to work on data with more flexible structures than grids and sequences, and hence more powerful translation techniques are required in order to handle more generic graph-structured data. This has been widely applied into many applications, e.g. predicting future states of a system in the physical domain based on the fixed relations (e.g. gravitational forces) among nodes [80] and the traffic speed forecasting on the road networks [81, 82]. Though they can work on generic graph-structured data, they assume that the graphs from the input domain and target domain share the same graph topology but cannot model or predict the change of the graph

topology.

To address the above issues where the topology can change during translation, deep learning-based graph translation problem has debuted in the very recent years. This problem is promising and critical to the domains where the variations of the graph topology are possible and frequent such as social network and cyber-network. For example, in social networks where people are the nodes and their contacts are the edges, the contact graph among them vary dramatically across different situations. For example, when the people are organizing a riot, it is expected that the contact graph to become denser and several special "hubs" (e.g., key players) may appear. Hence, accurately predicting the contact network in a target situation is highly beneficial to situational awareness and resource allocation. Existing topology translation models [26,42] predict the graph topology (i.e., edges) in a target domain based on that in an input domain. They focus on predicting the graph topology but assume that the node attributes value are fixed or do not exist.

Therefore, existing works either predict node attributes upon fixed topology or predict edge attributes upon fixed node attributes, as introduced in the last chapter. However, in many applications, both node attributes and edge attributes can change. In this work, such generic problem is named as *multi-attributed graph translation*, with important real-world applications ranging from biological structural to functional network translation [74] to network intervention research [83]. For example, the process of malware confinement[1] over IoT (Internet of Things) is typically a graph translation problem as shown in Fig. 4.1. It takes the initial status of IoT as input, and predicts the target graph which is ideally the optimal status of the network with modified connections (i.e., edges) and devices (i.e., nodes) state that helps to limit malware propagation and maintain network throughput. Epidemic controlling can also be considered as a multi-attributed graph translation problem, which is to estimate how the initial disease contact network (i.e., multi-attributed edges) and the human health stage (i.e., multi-attribute nodes) are jointly changed after the specific

[1] A device infected in an IoT network can propagate to other nodes connected to it, leading to contaminating the whole network, such as MiraiBot attack. As such, it is non-trivial to confine the malware to limit the infection and also equally important to maintain overall network connectivity and performance.

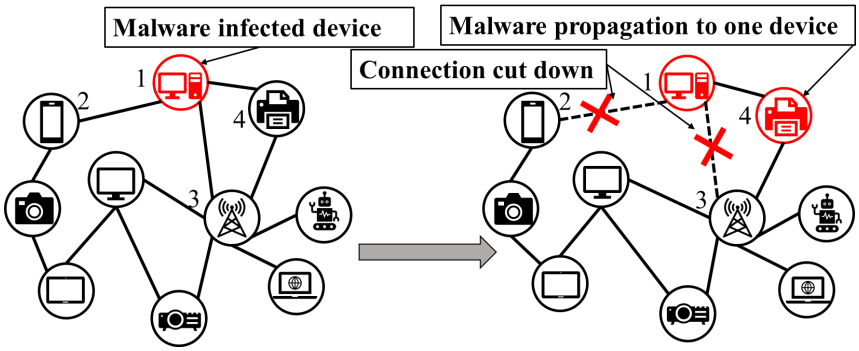


Figure 3.1: Given the network at time t (shown in the left graph), malware confinement is conducted to predict the most optimal status at time $t + \gamma$ shown in the right, where Devices 2 and 3 are protected by cutting the links (edges) to the compromised Device 1, while the Device 4 is propagated by malware without cutting link.

interventions. Since multi-attributed graph translation problem is highly sophisticated, there is no generic framework yet, but only ad-hoc methods for few specific domains, which heavily rely on intensive hand-crafting and domain-specific mechanistic models that could be extremely time- and resource- consuming to run in large scale. Hence, a generic, efficient, and end-to-end framework for general multi-attributed graph translation problems is highly in demand. Such framework needs to be able to comprehensively learn the translation mapping, remedy human bias by enjoying the large historical data, and achieve efficient prediction.

In this chapter, we focus on the generic problem of multi-attributed graph transformation, which cannot be handled by the existing methods because of the following challenges: 1) **Translation of node and edge attributes are mutually dependent**. The translation of edge attributes should not only consider edges, but also the node attributes. For example, in Fig. 4.1, two links are cut down since their linked Device 1 is compromised, which exemplifies the interplay between nodes and edges. Similarly, node translation also needs to jointly consider both nodes and edges, e.g., Device 4 is infected due to its link to Device 1. All the above issues need to be jointly considered but no existing works can

handle. 2) **Asynchronous and iterative changes of node and edge attributes during graph translation**. The multi-attributed graph translation process may involve a series of iterative changes in both edge and node attributes. For example in Fig.4.1, the translation could take several steps since the malware propagation is an iterative process from one device to the others. The links to a device may be cut (i.e., edge changes) right after it is compromised (i.e, node attribute change). These orders and dependencies of how node and edge attributes change during the translation are very important, yet difficult to be learned. 3) **Difficulty in discovering and enforcing the correct consistency between node attributes and graph spectra**. Although the predicted node and edge attributes are two different outputs, they should be highly dependent on each other instead of being irrelevant. For example, as shown in Fig. 4.1, the reason why Devices 2 and 3 on the right graph are not compromised is that they have no links with the compromised Device 1 anymore. It is highly challenging to learn and maintain the consistency of node and edge attributes, which are very sophisticated and domain-specific patterns.

To the best of our knowledge, this is the first work that addresses all the above challenges and provides a generic framework for the multi-attributed graph translation problem. We propose an Node-Edge Co-evolving Deep Graph Translator (NEC-DGT) with novel architecture and components for joint node and edge translation. Multi-block network with novel interactive node and edge translation paths are developed to translate both node and edge attributes, while skip-connection is utilized among different blocks to allow the non-synchronicity of changes in node and edge attributes. A novel spectral graph regularization is designed to ensure the consistency of nodes and edges in generated graphs. The contributions of this work are summarized as follows:

- **The development of a new framework for multi-attributed graph translation**. We formulate, for the first time, a multi-attributed graph translation problem and propose the NEC-DGT to tackle this problem. The proposed framework is generic for different applications where both node and edge attributes can be transformed.
- **The proposal of novel and generic edge translation layers and blocks**. A

new edge translation path is proposed to translate the edge attributes from the input domain to the output domain. Existing edge translation methods were proven to be special cases of ours, which can handle broad multi-attribute edges and nodes.

- **The proposal of a spectral-based regularization that ensures consistency of the predicted nodes and edges**. In order to discover and maintain the inherent relationships between predicted nodes and edges, a new non-parametric graph Laplacian regularization with a graph frequency regularization is proposed and leveraged.

- **The conduct of extensive experiments to validate the effectiveness and efficiency of the proposed model**. Extensive experiments on four synthetic and four real-world datasets demonstrated that NEC-DGT is capable of generating graphs close to ground-truth target graphs and significantly outperforms other comparison graph generative models.

3.2 Related Works

Traditional graph transformation. Traditional graph transformation target on handling a specified relationship between the input and output graphs, including graph morphism, conceptual graph projecting and reasoning, and graph matching [84,85]. Specifically, graph morphism aims to rewrite the input graph into its isomorphic graph based on predefined transformation rules [86,87]. For many applications, one requires more than graphs labeled over a finite alphabet. Hence attributed graph transformation methods are proposed to deal with graphs that are attributed with elements of given data types (e.g., integer, string, boolean) [88–90]. They can perform computations (e.g., add two integers) of attributes and define guards that restrict the applicability of rules (e.g., apply the rule only if a certain attribute is above some threshold). Conceptual graph projection and reasoning only handles the conceptual graph which is a knowledge representation scheme in formalizing semantic networks. These kinds of graphs are also attributed graphs with two types of nodes: concepts (which represent objects, entities or ideas) and relation nodes, which represent

relations between the concepts [91]. Reasoning based on the conceptual graphs can be regarded as a translation problem where the output graph is generated based on some predefined canonical formation rules, such as equivalence, specialization, and generalization. Concept Projection in ontology is used to find out whether two ontologies are semantically compatible or similar.

Though the above problem is about translating an input graph into a target graph, these traditional graph transformation requires the pre-knoledge of the relationship between the input and target graph (e.g., isomorphism or semantically compatible), based on which a set of rules and operations (e.g., projection) are hand-crafted. However, In real-world graphs, the relationship between two graphs can be various and complex and is hard to define and observe. And the graphs are also not limited to semantic graph or logical graphs. Thus, to model and learn the relationship between the input and output graphs, we resorts to deep learning methods on graphs via observing from large amounts of data. The most significance of the proposed deep learning-based graph transformation model is that given any large amounts of input-target graphs where the relationship and translation rules are unknown, our proposed data-driven model could automatically learn and model this translation process via optimizing an objective function.

Graph neural networks learning. In recent years, there has been a surge in research focusing on graph neural networks, which are generally divided into two categories: Graph Recurrent Networks [43,92,93] and Graph Convolutional Networks [32,46,65,94–98]. Graph Recurrent Networks originates from the early works of graph neural networks proposed by Gori et al. [43] and Scarselli et al. [92] based on recursive neural networks. Another line of research is to generalize convolutional neural networks from grids (e.g., images) to generic graphs. Bruna et al.[99] first introduced the spectral graph convolutional neural networks, and then it was extended in [46] using fast localized convolutions, which is further approximated for an efficient architecture for a semi-supervised setting [97]. The above-mentioned methods all focus on the general problem of graph representation learning, which aims to embed the graphs into the low-dimension spaces. They provide the fundamental theories

and operations for many downstream tasks such as node classification, graph classification, and graph generation.

Deep Generative Models for Graph generation. Deep generative models for graph generation aim to learn the distribution of a set of observed graphs via a deep generative model. Most of the existing GNN based graph generation for general graphs have been proposed in the last two years and are based on VAE [23, 38] and generative adversarial nets (GANs) [49], among others [24, 35]. Most of these approaches generate nodes and edges sequentially to form a whole graph, leading to the issues of being sensitive to the generation order and very time-consuming for large graphs. Differently, GraphRNN [35] builds an autoregressive generative model on these sequences with LSTM model and has demonstrated its good scalability. However, the above-mentioned models all deal with the unconditional graph generation instead of generating a target graph conditioning on an input graph, as our problem requires in this chapter.

Graph structured data transformation. The existing Graph structured data translation either deal with the node attributes prediction or translate the graph topology. Node attributes prediction aims at predicting the node attributes given the fixed graph topology [80–82, 100]. Li et al. [81] propose a Diffusion Convolution Recurrent Neural Network (DCRNN) for traffic forecasting which incorporates both spatial and temporal dependency in the traffic flow. Yu et al. [82] formulated the node attributes prediction problem of graphs based on the complete convolution structures. Graph topology translation considers the change of graph topology from one domain distributions to another. Guo et al. [26] proposed and tackled graph topology translation problem by proposing a generative model consisting of a graph translator with graph convolution and deconvolution layers and a new conditional graph discriminator, as introduced in Chapter 2. Sun et al. [42] proposed a graphRNN based model which generates a graph's topology based on another graph. Beyond the above two separate tasks, our proposed model can jointly predict the node attributes and topology via modeling the complex interaction between them during the translation process.

3.3 Problem Formulation

This chapter focuses on predicting a *target multi-attributed graph* based on an *input multi-attributed graph* by learning the graph translation mapping between them. The following provides the notations and mathematical problem formulation.

Table 3.1: Important notations and descriptions

Notations	Descriptions
$G(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0)$	Input graph with node set $\mathcal{V}_0$, edge set $\mathcal{E}_0$, edge attributes tensor E_0 and node attributes matrix F_0
$G(\mathcal{V}', \mathcal{E}', E', F')$	Target graph with node set $\mathcal{V}'$, edge set $\mathcal{E}'$, edge attributes tensor E' and node attributes matrix F'
C	Contextual information vector
N	Number of nodes
M	Number of edges
D	Dimension of node attributes
K	Dimension of edge attributes
c	Dimension of contextual information vector
S	Number of translation blocks

Define an input graph as $G(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0)$ where $\mathcal{V}_0$ is the set of N nodes, and $\mathcal{E}_0 \subseteq \mathcal{V}_0 \times \mathcal{V}_0$ is the set of M edges. $e_{i,j} \in \mathcal{E}_0$ is an edge connecting nodes $i \in \mathcal{V}_0$ and $j \in \mathcal{V}_0$. $\mathcal{E}_0$ contains all pairs of nodes while the existence of $e_{i,j}$ is reflected by its attributes. $E_0 \in \mathbb{R}^{N \times N \times K}$ is the edge attributes tensor, where $E_{0,i,j} \in \mathbb{R}^{1 \times K}$ denotes the edge attributes of edge $e_{i,j}$ and K is the dimension of edge attributes. $F_0 \in \mathbb{R}^{N \times D}$ refers to the node attribute matrix, where $F_{0,i} \in \mathbb{R}^{1 \times D}$ is the node attributes of node i and D is the dimension of the node attributes. Similarly, we define the target graph as $G(\mathcal{V}', \mathcal{E}', E', F')$. Note that the target and input graphs are different both in their node attributes as well as edge attributes. Moreover, vector C provides some contextual information on the translation process. Therefore, multi-attributed graph translation is defined as learning a mapping: $\mathcal{T} : G(\mathcal{V}_0, \mathcal{E}_0, E_0, F_0); C \rightarrow G(\mathcal{V}', \mathcal{E}', E', F')$.

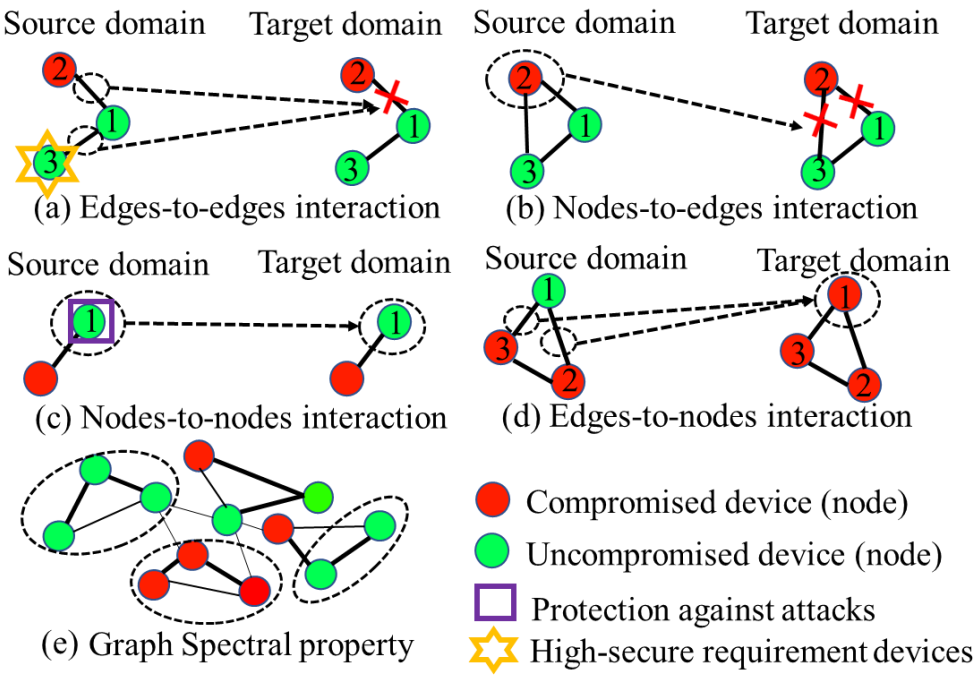


Figure 3.2: Five types of interactions during graph translation in the example of malware confinement. Node attributes are indications of malware attacks of IoT devices and edges represent the connections between devices.

For example, considering the malware confinement case where the nodes refer to IoT devices and the edges reflect the communication links between two devices. The node attributes include the malware-infection status and the properties of that device (i.e., specification and anti-virus software features). A single IoT device (i.e., node) that is compromised has the potential to spread malware infection across the network, eventually compromising the network or even ceasing the network functionality. In contrast, in order to avoid malware spreading as well as maintain the performance of the network, the network connectivity (i.e., graph topology) should be modified through *malware confinement*, thus to change the device status (i.e., node attributes) accordingly. Hence, malware confinement can be considered as predicting the optimal topology as well as the corresponding node and edge attributes of the target graph, where both malware prevention and device performance are maximized.

Multi-attributed graph translation problem requires to highlight several unique considerations as depicted in Fig. 3.2: 1) **Edges-to-edges interaction**: In target domain, the edge attributes $E'_{i,j}$ of an edge $e_{i,j}$ can be influenced by its incident edges' attributes $E_{0,i,k}$ and $E_{0,k,j}$ in input domain. For example, in Fig. 3.2 (a), if Devices 1 and 3 must be prevented from infection, then the edges between the compromised Device 1 and Device 2 need to be cut, due to the paths among them in input domain. 2) **Nodes-to-edges interaction**: In target domain, the attributes $E'_{i,j}$ of edge $e_{i,j}$ can be influenced by its incident nodes' attributes $F_{0,i}$ and $F_{0,j}$ in the input domain. As shown in Fig. 3.2 (b), if Device 2 is compromised in input domain, then in target domain, only its connections to Devices 1 and 3 need to be removed but the connection between Devices 1 and 3 can be retained because they are not compromised. 3) **Nodes-to-nodes interaction**: For a given node i, its attribute $F_{0,i}$ in input domain may directly influence its attribute F'_i in target domain. As shown in Fig. 3.2 (c), Device 3 with effective anti-virus protection (e.g. firewall) may not be easily compromised in target domain. 4) **Edges-to-nodes interaction**: For a given node i, its related edge attributes $E_{0,i,j}$ in input domain may affect its attributes F'_i in target domain. As shown in Fig. 3.2 (d), Device 1 which has more connections with compromised devices in input domain is more likely to be infected in target domain. 5) **Spectral Graph Property**: There exist relationships between nodes and edges in one graph as reflected by the graph spectrum. These relationships are claimed to have some persistent or consistent patterns across input and target domains, which have also been verified in many real-world applications such as brain networks [74]. For example, as shown in Fig. 3.2 (e), the devices that are densely connected as a sub-community tend to be in the same node status, which is a shared pattern for relationships between nodes and edges in different domains.

Multi-attributed graph translation should consider all the above properties, which cannot be comprehensively handled by existing methods because: 1) Lack of a generic framework to simultaneously characterize and automatically infer all of the above node-edge

interactions during translation process. 2) Difficulty in automatically discovering and characterizing the inherent spectral relationship between the nodes and edges in each graph, and ensuring consistent spectral patterns in graphs across input and target domains. 3) All the above interactions could be imposed repeatedly, alternately, and asynchronously during the translation process. It is difficult to discover and characterize such important yet sophisticated process.

3.4 The Proposed Method: NEC-DGT

In this section, we propose the Node-Edge Co-evolving Deep Graph Translator (NEC-DGT) to model the multi-attributed graph translation process. First, an introduction of the overall architecture and the loss functions is given. Then, the elaborations of three modules on edge translation, node translation, and graph spectral regularization are presented.

3.4.1 Overall architecture

Multi-block asynchronous translation architecture. The proposed NEC-DGT learns the distribution of graphs in the target domain conditioning on the input graphs and contextual information. However, such a translation process from input graph to the final target graph may experience a series of interactions of different types among edges and nodes. Also, such a sophisticated process is hidden and needs to be learned by a sufficiently flexible and powerful model. To address this, we propose the NEC-DGT as shown in Fig. 4.2. Specifically, the node and edge attributes of input graphs are inputted into the model and the model output the generated target graphs' node attributes and edge attributes after several blocks. The skip-connection architecture (black dotted lines in Fig. 4.2) implemented across different blocks aims to deal with the asynchrony property of different blocks, which ensures that the final translated results fully utilize various combinations of blocks' information. To train the deep neural network to generate the target graph $G(E', F')$ conditioning on the input graph $G(E_0, F_0)$ and contextual information C, we minimize the loss function

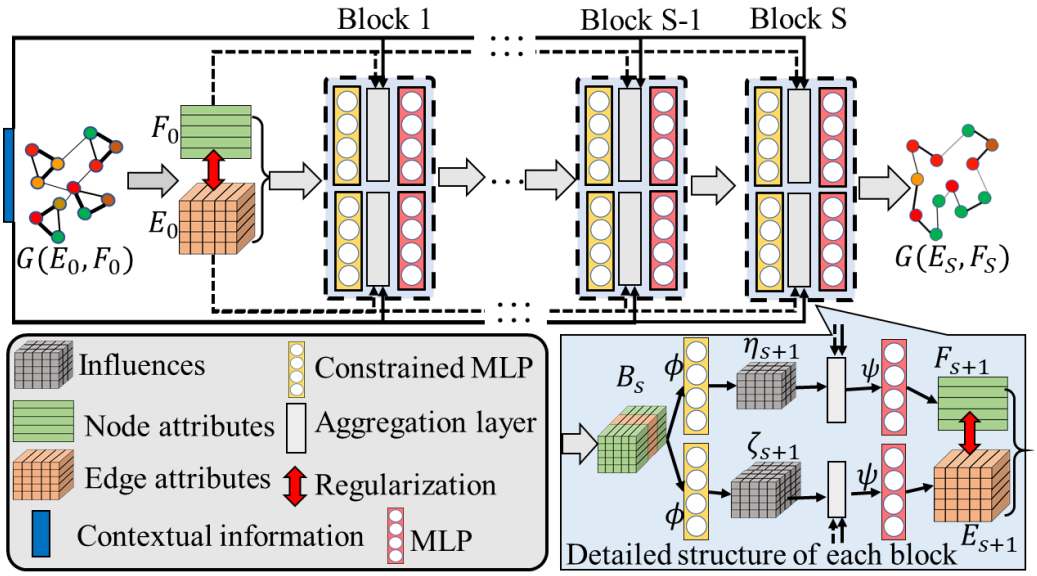


Figure 3.3: The proposed NEC-DGT consists of multiple blocks. Each block has edge and node translation paths which are co-evolved and combined by a graph regularization during training process.

as follows:

$$\mathcal{L}_{\mathcal{T}} = \mathcal{L}(\mathcal{T}(G(E_0, F_0), C), G(E', F')) \tag{3.1}$$

where the nodes set $\mathcal{V}_0$ and $\mathcal{V}'$ as well as edges set $\mathcal{E}_0$ and $\mathcal{E}'$ can be reflected in F_0 and F', as well as E_0 and E'.

Node and edge translation paths. To jointly tackle various interactions among nodes and edges, respective translation paths are proposed for each block. In node translation path (in upper part of detailed structure in Fig. 4.2), node attributes are generated considering the "nodes-to-nodes" and "edges-to-nodes" interactions. In edge translation path (in lower part of detailed structure in Fig. 4.2), edge attributes are generated following the "edges-to-edges" and "node-to-edges" interactions.

Spectral graph regularization. To discover and characterize the inherent relationship between nodes and edges of each graph, the frequency domain properties of the graph is learned, based on which the interactions between node and edge attributes are jointly

regularized upon non-parametric graph Laplacian. Moreover, to maintain consistent spectral properties throughout the translation process, we enforce the shared patterns among the generated nodes and edges in different blocks by regularizing their relevant parameters in the frequency domain. The regularization of the graphs is formalized as follows:

$$\mathcal{R}(G(E,F)) = \sum\nolimits_{s=0}^{S} \mathcal{R}_\theta(G(E_s,F_s)) + \mathcal{R}(\theta) \tag{3.2}$$

where S refers to the number of blocks, and θ refers to the overall parameters in the spectral graph regularization. E_s and F_s refer to the generated edge attributes tensor and node attributes matrix in the sth block. Thus $G(E_S, F_S)$ is the generated target graph. Then the final loss function can be summarized as follows:

$$\widetilde{\mathcal{L}} = \mathcal{L}(\mathcal{T}(G(E_0,F_0),C), G(E',F')) + \beta\mathcal{R}(G(E,F)) \tag{3.3}$$

where β is the trade-off between the $\mathcal{L}_\mathcal{T}$ and spectral graph regularization. The model is trained by minimizing the mean squared error of E_S with E', and F_S with F', enforced by the regularization. Optimization methods (e.g. Stochastic gradient descent (SGD) and Adam) based on Back-propagation technique can be utilized to optimize the whole model.

Subsequently, the details of a single translation block are introduced: edge translation path in Section 3.4.2, node translation path in Section 3.4.3 and graph spectral-based regularization in Section 3.4.4.

3.4.2 Edge Translation Path

Edge translation path aims to model the nodes-to-edges and edges-to-edges interactions, where edge attributes in the target domain can be influenced by both nodes and edges in the input domain. Therefore, we propose to first jointly embed both node and edge information into influence vectors and then decode it to generate edges attributes. Specifically, the edge translation path of each block contains two functions, *influence-on-edge function* which

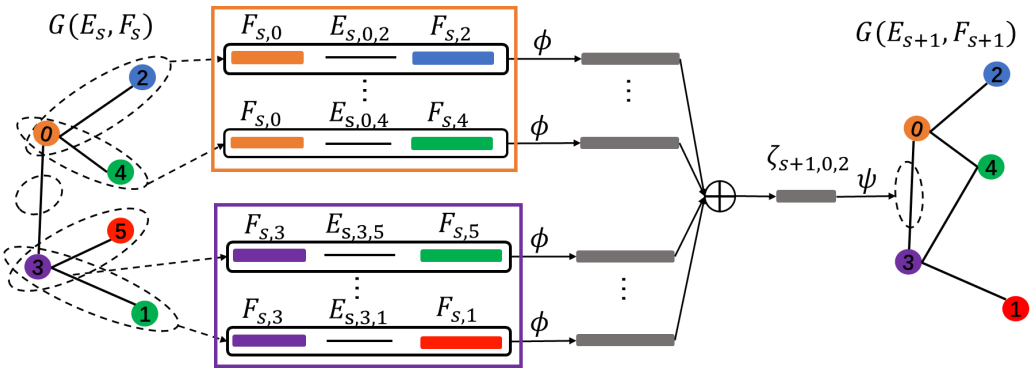


Figure 3.4: Details of edge translation path for one edge (i.e. $e_{0,3}$) in a single block.

encodes each pair of edge and node attributes into the influence for generating edges, and the *edge updating function* which aggregates all the influences related to each edge into an integrated influence and decodes this integrated influence to generate each edge' attributes. Fig. 3.4 shows the operation of the two functions in a single block by translating the current input of graph $G(E_s, F_s)$ to output graph $G(E_{s+1}, F_{s+1})$.

Influence-on-edge layers. As shown in Fig. 3.4, the input graph $G(E_s, F_s)$ is first organized in unit of several pairs of node and edge attributes. For each pair of nodes v and u, we concatenate their edge attributes $E_{s,u,v}$ and their node attributes: $F_{s,u}$ and $F_{s,v}$ as: $B_{s,u,v} = [F_{s,u}, E_{s,u,v}, F_{s,v}]$ (as circled in black rectangles in Fig. 3.4). Then $B_{s,u,v} \in \mathbb{R}^{1\times(2D+K)}$ is inputted into the influence-on-edge function: a constrained MLP (Multilayer Perceptron) ϕ which is used to calculate the influence $\phi(B_{s,u,v}) \in \mathbb{R}^{1\times q}$ from the pair of the nodes u and v. q refers to the dimension of the final influence on edges. ϕ for edge translation path is expressed as follows:

$$\begin{aligned} \phi(X; W_E, b_E) =& \sigma_M(...(\sigma_0(X \cdot W_E^{(0)} + b_E^{(0)})...W_E^{(M)} + b_E^{(M)}) \\ s.t., & W_{E,1:D}^{(0)} \equiv W_{E,(D+K):(2D+K)}^{(0)} \end{aligned} \tag{3.4}$$

where W_E and b_E are weights and bias for ϕ in edge translation path. M refers to the

number of layers of ϕ and $\{\sigma_0, ...\sigma_M\}$ refers to the activation functions. For undirected graph, we add a weight constraint $W^{(0)}_{E,1:D} \equiv W^{(0)}_{E,(D+K):(2D+K)}$ to ensure that the influence of $B_{s,u,v}$ is the same as the influence of $B_{s,v,u}$, which means that the first D rows (related to the attributes of node u) and the last D rows (related to the attributes of node v) of $W^{(0)}_E$ are shared. The influence on edges of each pair is computed through the same function with the same weights. Thus the NEC-DGT can handle various size of graphs.

Edge updating layers. After calculating the influence of each pair of nodes and edge, the next step is to assigning each pairs' influences to its related edge to get the integrated influence for each edge (as shown of $\bigoplus$ operation in Fig. 3.4). This is because each edge is generated depending on both its two related nodes and its incident edges (like the pairs circled in the orange rectangle and purple rectangle related to node 0 and node 3 respectively in Fig. 3.4). Here we define the integrated influence on one edge attribute $E_{s+1,i,j}$ as: $\zeta_{s+1,i,j} \in \mathbb{R}^{1\times q}$, which is computed as follows:

$$\begin{aligned}\zeta_{s+1,i,j} = &\sum\nolimits_{k_1\in N(i)} \phi(B_{s,i,k_1}; W_E, b_E) + \\ &\sum\nolimits_{k_2\in N(j)} \phi(B_{s,k_2,j}; W_E, b_E)\end{aligned} \tag{3.5}$$

where $N(i)$ refers to the neighbor nodes of node i. Then the edge attributes $E_{s+1,i,j}$ is generated by $\psi([E_{0,i,j}, \zeta_{s+1,i,j}, C])$, where $E_{0,i,j}$ refers to the input edge attributes of edge $e_{i,j}$. C refers to the contextual information for the translation. The function ψ is implemented by an MLP.

Relationship with other edge convolution networks. Edge convolution network is the most typical method to handle the edge embedding in graphs, which was first introduced as BrainNetCNN [95] and later explored in many studies [26,101,102]. Our edge translation path is a highly flexible and generic mechanism to handle multi-attributed nodes and edges. Several existing edge convolution layers and their variants can be considered as special cases

of our method, as demonstrated in the following theorem. The proof process is available at Appendix A.1.

Theorem 1. *The influence-on-edge function ϕ in edge translation path of NEC-DGT is a generalization of conventional edge convolution networks.*

3.4.3 Node Translation Path

Node translation aims to learn the "nodes-to-nodes" and "edges-to-nodes" interactions, where translation of one node's attributes depends on the edge attributes related to this node and its own attributes. The node translation path of each block contains two functions, *influence-on-node function* which learns the influence from each pair of nodes, and *node updating function* which generates the new node attributes by aggregating all the influences from pairs containing this node. Fig. 3.5 shows how to translate a node in a single block.

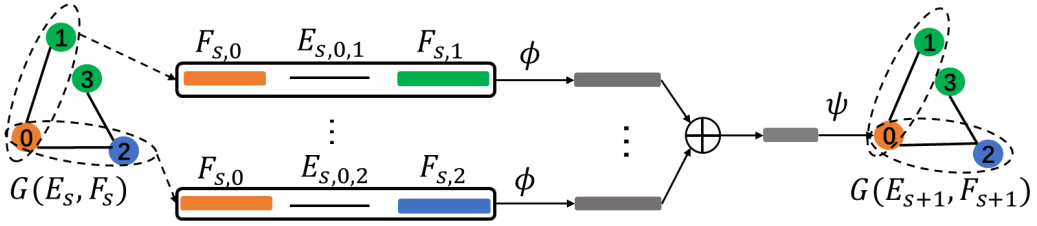


Figure 3.5: Details of node translation path for one node (i.e., node 0) in a single block.

Influence-on-node layers. As shown in Fig. 3.5, the input graph $G(E_s, F_s)$ is first organized in the unit of pairs of nodes, where each pair is $B_{s,u,v} \in \mathbb{R}^{1\times(2D+K)}$ which is similar to the edge translation path (as circled in the black rectangle in Fig. 3.5). Then $B_{s,u,v}$ is inputted into the influence-on-node function, which is implemented by contrained MLP ϕ as Equation (3.4), to compute the influence $\phi(B_{s,u,v}; W_F, b_F) \in \mathbb{R}^{1\times h}$ to nodes (as shown in the grey bar after ϕ in Fig. 3.5), where h is the dimension of the influence on nodes.

Node updating layers. After computing the influences of each node pair, the next step is to generate node attributes. For node i, an assignment step is required to aggregate all the influences from pairs containing node i (as shown of $\bigoplus$ operation in Fig. 3.5). Thus, all the influences for node i are aggregated and input into the updating function, which is implemented by a MLP model ψ to calculate the attributes of node i as: $F_{s+1,i} = \psi([F_{0,i}, \sum_{j \in N(i)} \phi(B_{s,i,j}; W_F, b_F), C])$.

3.4.4 Graph spectral-based regularization

Based on the edge and node translation path introduced above, we can generate node and edge attributes, respectively. However, since these generated node and edge attributes are predicted separately in different paths, their patterns may not be consistent and harmonic. To ensure the consistency of the edge and node patterns mentioned in Section 3.3, we propose a novel adaptive regularization based on non-parametric graph Laplacian, and a graph frequency regularization.

Non-parametric Graph Laplacian Regularization. First, we recall the property of the multi-attributed graphs where node information can be smoothed over the graph via some form of explicit graph-based regularization, namely, by the well-known graph Laplacian regularization term [97]: $F_s^{(d)^T} L_s^{(k)} F_s^{(d)} = \sum_{i,j \in \mathcal{V}} E_{s,i,j}^{(k)} \left\| F_{s,i}^{(d)} - F_{s,j}^{(d)} \right\|^2$, where $F_s^{(d)} \in \mathbb{R}^{N \times 1}$ is the node attribute vector for the dth node attribute and $E_s^{(k)} \in \mathbb{R}^{N \times N}$ is the edge attribute matrix for kth attribute generated in the sth block. $L_s^{(k)} = D_s^{(k)} - E_s^{(k)}$ denotes the graph Laplacian for the kth edge attributes matrix. The degree matrix $D_s^{(k)} \in \mathbb{R}^{N \times N}$ is computed as: $D_{s,i,i}^{(k)} = \sum_{j \in N(i)} E_{s,i,j}^{(k)}$.

However, the above traditional graph Laplacian can only impose an absolute smoothness regularization over all the nodes by forcing the neighbor nodes to have similar attribute values, which is often over-restrictive for many situations such as in signed networks and telecommunications networks. In the real world, the correlation among the nodes is much more complicated than purely "smoothness" but should be a mixed pattern of different

types of relations. To address this, we propose an end-to-end framework of non-parametric graph Laplacian which can automatically learn such node correlation patterns inherent in specific types of graphs, with rigorous foundations on spectral graph theory. In essence, we propose the non-parametric graph Laplacian based on the parameter θ as: $g_\theta(\hat{L}_s^{(k)})$. $\hat{L}_s^{(k)}$ is the normalized Laplacian computed as $\hat{L}_s^{(k)} = D_s^{(k)^{-\frac{1}{2}}} L_s^{(k)} D_s^{(k)^{-\frac{1}{2}}}$ and can be diagonalized by the Fourier basis $U_s^{(k)} \in \mathbb{R}^{N\times N}$, such that $\hat{L}_s^{(k)} = U_s^{(k)} \Lambda_s^{(k)} U_s^{(k)^T}$ where $\Lambda_s^{(k)} \in \mathbb{R}^{N\times N}$ is a diagonal matrix storing the graph frequencies. For example, $\Lambda_{s,1}^{(k)}$ is the frequency value of the first Fourier basis $U_{s,1}^{(k)}$. Then we got $g_\theta(\hat{L}_s^{(k)}) = g_\theta(U_s^{(k)} \Lambda_s^{(k)} U_s^{(k)^T}) = U_s^{(k)} g_\theta(\Lambda_s^{(k)}) U_s^{(k)^T}$. Therefore, we have the regularization as follows:

$$\mathcal{R}_\theta(G(E_s, F_s)) = \sum\nolimits_{k=1}^{K} \sum\nolimits_{d=1}^{D} F_s^{(d)^T} U_s^{(k)} g_\theta(\Lambda_s^{(k)}) U_s^{(k)^T} F_s^{(d)} \tag{3.6}$$

where $g_\theta(\Lambda_s^{(k)})$ is a non-paramteric Laplacian eigenvalues that will be introduced subsequently.

Scalable approximation. $g_\theta(\Lambda_s^{(k)})$ is a non-parametric vector whose parameters are all free; It can be defined as: $g_\theta(\Lambda_s^{(k)}) = diag(\theta_s^{(k)})$, where the parameter $\theta_s^{(k)} \in \mathbb{R}^N$ is a vector of Fourier coefficients for a graph. However, optimizing the parametric eigenvalues has the learning complexity of $O(N)$, the dimensionality of the graphs, which is not scalable for large graphs. To reduce the learning complexity of $O(N)$ to $O(\mathbf{1})$, we propose approximating $g_\theta(\Lambda_s^{(k)})$ by a normalized truncated expansion in terms of Chebyshev polynomials [103]. The Chebyshev polynomial $T_p(x)$ of order p may be computed by the stable recurrence relation $T_p(x) = 2xT_{p-1}(x) - T_{p-2}(x)$ with $T_1 = 1$ and $T_2 = x$. The eigenvalues of the approximated Laplacian filter can thus be parametric as the truncated expansion:

$$g_\theta(\Lambda_s^{(k)}) = \sum\nolimits_{p=1}^{P} \theta_{s,p}^{(k)} T_p(\widetilde{\Lambda}_s^{(k)}) / \sum\nolimits_{p=1}^{P} \theta_{s,p}^{(k)} \tag{3.7}$$

for P orders, where $T_p(\widetilde{\Lambda}_s^{(k)}) \in \mathbb{R}^{N\times N}$is the Chebyshev polynomial of order p evaluated at $\widetilde{\Lambda}_s^{(k)} = 2\Lambda_s^{(k)}/\Lambda_{s,max}^{(k)} - I$, a diagonal matrix of scaled eigenvalues that lie in $[-1, 1]$. The $\Lambda_{s,max}$ refers to the largest element in $\Lambda_s^{(k)}$. $\theta \in \mathbb{R}^{S\times P\times K}$ denotes the parameter tensor for all S blocks. $\theta_{s,p}^{(k)}$ is the pth element of Chebyshev coefficients vector $\theta_s^{(k)} \in \mathbb{R}^P$ for the kth edge attribute. Each $\theta_{s,p}^{(k)}$ is normalized by dividing the sum of all the coefficients in $\theta_s^{(k)}$ to avoid the situation where $\theta_s^{(k)}$ is trained as zero. Thus, the laplacian computation can then be written as $g_\theta(\widetilde{L}_s^{(k)}) = \sum_{p=1}^{P} \theta_{s,p}^{(k)} T_p(\widetilde{L}_s^{(k)}) / \sum_{p=1}^{P} \theta_{s,p}^{(k)}$, where $T_p(\widetilde{L}_s^{(k)}) \in \mathbb{R}^{N\times N}$ is the Chebyshev polynomial of order p evaluated at the scaled Laplacian $\widetilde{L}_s^{(k)} = 2\hat{L}_s^{(k)}/\Lambda_{s,max}^{(k)} - I$. For efficient computation, we further approximate $\Lambda_{s,max}^{(k)} \approx 1.5$, as we can expect that the neural network parameters θ will adapt to this change in scale during training.

Graph frequency regularization. To ensure that the spectral graph patterns are consistent throughout the translation process across different blocks, we utilize a graph frequency regularization to not only maintain the similarity but also allow the exclusive properties of each block's patterns to be reserved to some degree. Specifically, regarding all the frequency pattern basis of form $\widetilde{L}$, some are important in modeling the relationships between nodes and graphs while some are not, resulting in the sparsity pattern of θ. Thus, inspired by the multi-task learning, we learn the consistent sparsity pattern of θ_s by using the $L_{2,1}$ norm as regularization:

$$\mathcal{R}(\theta) = \sum\nolimits_{k=1}^{K} \sum\nolimits_{s=1}^{S} \sqrt{\sum\nolimits_{p=1}^{P} \theta_{s,p}^{(k)2}} \tag{3.8}$$

3.4.5 Extensions of Graph Spectral-based regularization

In this section, we extend our framework of multi-attributed graph translation into broadly signed graph and directed graphs, which makes our model more general to various graph types. The node and edge translation paths do not need to change since the trainable parameters can be adaptive in dealing with various edge and node attributes. However,

apart from the two translation paths, the spectral-based regularization term cannot be utilized without change for signed or directed graphs. This is incurred by several technical challenges: the Laplacian matrix for both sign and directed graph are different from the undirected graphs. The way to calculate the Laplacian matrix which should be symmetric and positive semi-definite needs to be customized for both signed and direct graphs. Thus, the way to approximate the Laplacian needs also changes accordingly. Thus, we explore how to fit our model into the signed and directed graph translation respectively, by proposing the signed spectral-based graph regularization and directed spectral-based graph regularization.

Extension to Signed Graphs

Signed graphs are defined as the weighted graphs in which negative and positive entries are allowed, the intuition being that a negative weight indicates dissimilarity or distance. Thus, for a signed graph, the degree matrix may contain zero or negative entries (thus $D_s^{(k)^{-\frac{1}{2}}}$ may not exist), and the Laplacian $L_s^{(k)}$ may no longer be positive semi-definite. However, the spectral-based graph regularization requires the Laplacian to be positive semi-definite for representing the relations between the node and edge attributes.

Thus, we adopt a definition of the signed graph Laplacaian [104]. First, the degree matrix is computed as $\bar{D}_{s,i,i}^{(k)} = \sum_{j \in N(i)} \left| E_{s,i,j}^{(k)} \right|$, and then the normalized version of the Laplacian for the signed graph is $L_s^{(k)} = I - \bar{D}_s^{(k)^{-\frac{1}{2}}} E_s^{(k)} \bar{D}_s^{(k)^{-\frac{1}{2}}}$. In this way, the signed graph Laplacian can be positive semi-definite and decomposed with eigenvalues and eigenvectors. Thus, we can fit our proposed non-parametric regularization and scalable approximation into the signed graph as

$$\mathcal{R}_\theta(G(E_s, F_s)) = \sum_{k=1}^{K} \sum_{d=1}^{D} F_s^{(d)^T} \sum_{p=1}^{P} \theta_{s,p}^{(k)} T_p(\widetilde{L}_s^{(k)}) F_s^{(d)} / \sum_{p=1}^{P} \theta_{s,p}^{(k)}, \tag{3.9}$$

where $\widetilde{L}_s^{(k)} = 2L_s^{(k)}/\Lambda_{s,max}^{(k)} - I$. Thus, the signed spectral-based graph regularization can enjoy a constant time complexity $O(1)$.

Extension to Directed Graphs

For directed graphs, the major problem is that the edge attribute matrix is not symmetric, and the way to calculate the Laplacian matrix described above cannot semantically enforce the relations between edges and nodes. Here, we calculate the Laplacian of the directed graph as a Hermitian matrix by using the transition probability matrix [105].

First, we use $E_{s,i,j}$ to denote the sth edge attributes of the edge from the ith node to the jth node. Thus, for a weighted directed graph with edge weights $E_{s,i,j} > 0$, each element of a general transition probability matrix P_s regarding the sth attribute can be defined as

$$P_{s,i,j} = \frac{E_{s,i,j}}{\sum_k E_{s,i,k}}. \tag{3.10}$$

The Perron-Frobenius Theorem [106] states that an irreducible matrix with nonnegative entries has a unique (left) eigenvector with all entries positive. Let ρ_s denote the eigenvalue of the positive eigenvector of P_s. Namely, the transition probability matrix P_s of a strongly connected directed graph has a unique left eigenvector ϕ_s with $\phi_{s,i} > 0$ for all nodes i, and $\phi_s P_s = \rho_s \phi_s$. We can normalize and choose ϕ_s to satisfy $\sum_i \phi_{s,i} = 1$. Now the ϕ_s is called the Perron vector of P_s. Then the Laplacian of a directed graph is defined by

$$L_s = I - \frac{\Phi_s^{\frac{1}{2}} P_s \Phi_s^{\frac{-1}{2}} + \Phi_s^{\frac{-1}{2}} P_s^* \Phi_s^{\frac{1}{2}}}{2}, \tag{3.11}$$

where Φ_s is a diagonal matrix with entries $\Phi_{s,i,i} = \phi_{s,i}$, and P_s^* denotes the conjugated transpose of P_s. Next, we use the non-parametric approximation to define the spectral-based regularization. As mentioned in Section 3.4.4, we need to approximate the eigenvalues $g_\theta(\Lambda_s)$ of the original graph Laplacian. This means the decomposition of P_s to calculate Φ_s and the decomposition of L_s to calculate U_s (U_s is the eigenvector for L_s, as mentioned in Section 3.4.4) are both needed. However, it will lead to intensive computation, especially for large graphs. Thus, to avoid decomposition, the non-parametric method is used to

approximate $g_\theta(L_s)$ for directed graphs as

$$g_\theta(L_s) = g_\theta(I - \frac{\Phi_s^{\frac{1}{2}} P_s \Phi_s^{\frac{-1}{2}} + \Phi_s^{\frac{-1}{2}} P_s^* \Phi_s^{\frac{1}{2}}}{2}) \tag{3.12}$$

$$= (I - \frac{g_\theta(\Phi_s)^{\frac{1}{2}} P_s g_\theta(\Phi_s)^{\frac{-1}{2}} + g_\theta(\Phi_s)^{\frac{-1}{2}} P_s^* g_\theta(\Phi_s)^{\frac{1}{2}}}{2}), \tag{3.13}$$

It is easy to find that approximating L_s is equal to approximating Φ_s. It should be noted that this operation also reduces the computational effort that is spent on getting Φ_s by decomposing the transition matrix P_s, because otherwise we need to use a polynomial-time algorithm to calculate the exact Φ_s computationally since there is no closed form solution for Φ_s [107]. Now we approximate Φ_s by learning $g_\theta(\Phi_s) = diag(\theta_1, \cdots, \theta_N)$ with the complexity $O(N)$.

3.4.6 Complexity Analysis

The proposed NEC-DGT requires $O(N^2)$ operations in time complexity and $O(N^2)$ space complexity in terms of number of nodes in the graph. It is more scalable than most of the graph generation methods. For example, GraphVAE [23] requires $O(N^4)$ operations in the worst case and Li et al [24] uses graph neural networks to perform a form of "message passing" with $O(MN^2)$ operations to generate a graph.

3.5 Experiments

In this section, we present both the quantitative and qualitative experiment results on NEC-DGT as well as the comparison models. All experiments are conducted on a 64-bit machine with Nvidia GPU (GTX 1070, 1683 MHz, 8 GB GDDR5). The model is trained by ADAM optimization algorithm.

3.5.1 Datasets

We performed experiments on four synthetic datasets and four real-world datasets with different graph sizes and characteristics. All the dataset contain input-target pairs.

Synthetic dataset: Four datsets are generated based on different types of graphs and translation rules. The input graphs of the first three datasets (named as Syn-I, Syn-II, and Syn-III) are Erdos-Renyi (E-R) graphs generated by the Erdos Renyi model [18] with the edge probability of 0.2 and graph size of 20, 40, and 60 respectively. The target graph topology is the 2-hop connection of the input graph, where each edge in the target graph refers to the 2-hop reachability in the input graph (e.g. if node i is 2-hop reachable to node j in the input graph, then they are connected in the target graph). The input graphs of the fourth dataset (named as Syn-IV) are Barabási-Albert (B-A) graphs generated by the Barabási-Albert model [108] with 20 nodes, where each node is connected to 1 existing node. In Syn-IV, topology of target graph is the 3-hop connection of the input graph. For all the four datasets, the edge attributes $E_{s,i,j} \in [0, 1]$ denotes the existence of the edge. For both input and target graphs, the node attributes are continuous values computed following the polynomial function: $f(x) : y = ax^2 + bx + c(a = 0, b = 1, c = 5)$, where x is the node degree and $f(x)$ is the node attribute. Each dataset is divided into two subsets, each of which has 250 pairs of graphs. Validation is conducted where one subset is used for training and another for testing, and then exchange them for another validation. The average result of the two validations is regarded as the final result.

Malware confinement dataset: Malware dataset are used for measuring the performance of NEC-DGT for malware confinement prediction. There are three sets of IoT nodes at different amount (20, 40 and 60) encompassing temperature sensors connected with Intel ATLASEDGE Board and Beagle Boards (BeagleBone Blue), communicating via Bluetooth protocol. Benign and malware activities are executed on these devices to generate the initial attacked networks as the input graphs. Benign activities include MiBench [109] and SPEC2006 [110], Linux system programs, and word processor. The nodes represent devices

and node attribute is a binary value referring to whether the device is compromised or not. Edge represents the connection of two devices and the edge attribute is a continuous value reflecting the distance of two devices. The real target graphs are generated by the classical malware confinement methods: stochastic controlling with malware detection [83, 111, 112]. We collected 334 pairs of input and target graphs with different contextual parameters (infection rate, recovery rate, and decay rate) for each of the three datasets. Each dataset is divided into two subsets: one has 200 pairs and another has 134 pairs. The validation is conducted in the same way as the synthetic dataset.

Molecule reaction dataset: We apply our NEC-DGT to one of the fundamental problems in organic chemistry, thus predicting the product (target graph) of chemical reaction given the reactant (input graph). Each molecular graph consists of atoms as nodes and bond as edges. The input molecule graph has multiple connected components since there are multiple molecules comprising the reactants. The reactions used for training are atom-mapped so that each atom in the product graph has a unique corresponding atom in the reactants. We used reactions from USPTO granted patents, collected by Lowe [113]. we obtained a set of 5,000 reactions (reactant-product pair) and divided them into 2,500 and 2,500 for training and testing. Atom (node) features include its elemental identity, degree of connectivity, number of attached hydrogen atoms, implicit valence, and aromaticity. Bond (edge) features include bond type (single, double, triple, or aromatic), and whether it is connected.

Real-world HCP Dataset: The human connectome project (HCP) dataset is used for evaluating the signed graph translation model. Brain network prediction, such as prediction of functional connectivity based on structural connectivity, is a very critical task in neuroscience. The goal is to learn the mapping from the resting-state functional connectivity into task-specific functional connectivity in the human brain. In this dataset, the source and the target graphs respectively reflect the structural connectivity (SC) and the functional connectivity (FC) of the same subject's brain network. In particular, both types of connectivity are processed from the magnetic resonance imaging (MRI) data obtained from

the HCP dataset [68]. By following the preprocessing procedure in [69], the SC data is constructed by applying probabilistic tracking on the diffusion MRI data using the Probtrackx tool from the FMRIB Software Library [70] with 68 predefined regions of interest (ROIs). Then, the edge attributes of FC are defined as the Pearson's correlation between two ROIs' blood oxygen level-dependent time obtained from the resting-state functional MRI data. The node attributes refer to the index of each node by a one-hot vector. Since the edges of FC can be either positive or negative, signed graph translation model is needed to handle this task. In total, 823 pairs of SC and FC samples are used and 5-fold cross-validation is performed.

Online Breast Cancer Community Dataset: We adopt the dataset from Gao [56] for validation of the proposed di-NEC-DGT. The data is collected through the Breast Cancer Community,[2] which is one of the largest online forums designed for patients to share information related to breast cancer. The forum data collected for this study covers an eight-year period from the beginning of 2010 to the end of 2017. There are 80 subforums, such as "Not Diagnosed But Worried" and "Breast Reconstruction". The user subforum activity transition is defined as being when the users posted new topics or replied to existing topics and the time window was set as one month. After removing common words and stop words, 59 top-frequency keywords from the forum content construct the feature vectors for the sub-forums. Also, each user may come across different health stages. The health stages consists of "Dx," "Chemotherapy," "Targeted," "Hormonal," "Radiation," "Surgery,". Each transition graph of a user reflects the stage of the user, and it is obvious that the graph can change as the user transfers to another health stage since the things that concern them change. Our goal is to predict how the user transitions across subforums when their health stage changes from the current to the next, given the current transition graph. We randomly selected 70% of users who provided their health stage history for training, another 10% for validation, and the remaining 20% for testing. The predicted transition graphs were validated against the real transition graphs in the target stage. We treat "Dx"

[2]https://community.breastcancer.org/

as the initial input stage and treat the others as the target stages.

3.5.2 Comparison methods

Since there is no existing method handling the multi-attributed graph translation problem, NEC-DGT is compared with two categories of methods: 1) graph topology generation methods, and 2) graph node attributes prediction methods.

Graph topology generation methods: 1) GraphRNN [35] is a recent graph generation method based on sequential generation with LSTM model; 2) Graph Variational Auto-encoder (GraphVAE) [23] is a VAE based graph generation method for small graphs; 3) Graph Translation-Generative Adversarial Networks (GT-GAN) [26] is a new graph topology translation method based on graph generative adversarial network.

Node attributes prediction methods: 1) Interaction Network (IN) [80] is a node state updating network considering the interaction of neighboring nodes; 2) DCRNN [81] is a node attribute prediction network for tranffic flow prediction; 3) Spatio-Temporal Graph Convolutional Networks (STGCN) [82] is a node attribute prediction model for traffic speed forecast. Furthermore, to validate the effectiveness of the graph spectral-based regularization, we conduct a comparison model (named as NR-DGT) which has the same architecture of NEC-DGT but without the graph regularization.

3.5.3 Evaluation metrics

A set of metrics are used to measure the similarity between the generated and real target graphs in terms of node and edge attributes. To measure the attributes which are Boolean values, the Acc (accuracy) is utilized to evaluate the ratio of nodes or edges that are correctly predicted among all the nodes or possible node pairs. To measure the attributes which are continuous values, MSE (mean squared error), R2 (coefficient of determination score), Pearson and Spearman correlation are computed between attributes of generated and real target graphs. $N- < metric >$ represents metrics evaluated on node attributes and $E- < metric >$ represents metrics evaluated on edge attributes.

Table 3.2: Evaluation of Generated Target Graphs for Synthetic Dataset (N for node attributes, E for edge attributes, P for Pearson correlation, SP for Spearman correlation and Acc for accuracy).

dataset	Method	N-MSE	N-R2	N-P	N-Sp	Method	E-Acc
	IN	5.97	0.06	0.48	0.44	GraphRNN	0.6212
	DCRNN	51.36	0.12	0.44	0.45	GraphVAE	0.6591
Syn-I	STGCN	15.44	0.19	0.42	0.56	GT-GAN	0.7039
	NR-DGT	2.13	**0.87**	0.90	0.89	NR-DGT	0.7017
	NEC-DGT	**1.98**	0.76	**0.93**	**0.91**	NEC-DGT	**0.7129**
	IN	**1.36**	0.85	0.77	0.87	GraphRNN	0.5621
	DCRNN	71.07	0.11	0.39	0.37	GraphVAE	0.4639
Syn-II	STGCN	33.11	0.21	0.15	0.15	GT-GAN	0.7005
	NR-DGT	1.43	0.91	0.94	**0.97**	NR-DGT	0.7016
	NEC-DGT	1.91	**0.93**	**0.97**	**0.97**	NEC-DGT	**0.7203**
	IN	35.46	0.31	0.59	0.56	GraphRNN	0.4528
	DCRNN	263.23	0.09	0.41	0.39	GraphVAE	0.3702
Syn-III	STGCN	43.34	0.22	0.48	0.47	GT-GAN	0.5770
	NR-DGT	5.90	0.90	0.94	0.92	NR-DGT	0.6259
	NEC-DGT	**4.56**	**0.93**	**0.97**	**0.96**	NEC-DGT	**0.6588**
	IN	4.63	0.10	0.53	0.51	GraphRNN	0.5172
	DCRNN	63.03	0.12	0.22	0.16	GraphVAE	0.3001
Syn-IV	STGCN	6.52	0.08	0.11	0.10	GT-GAN	0.8052
	NR-DGT	4.49	0.12	0.55	0.54	NR-DGT	0.6704
	NEC-DGT	**1.86**	**0.73**	**0.93**	**0.89**	NEC-DGT	**0.8437**

3.5.4 Evaluation for synthetic datasets

For synthetic datasets, we compare the generated and real target graphs on various metrics and visualize the patterns captured in the generated graphs.

Metric-based evaluation. Table 3.2 summarizes the effectiveness comparison for four synthetic datasets. The node attributes are continuous values evaluated by N-MSE, N-R2, N-P, and N-SP. The edge attributes are binary values evaluated by the accuracy of the correctly predicted edges. The results in Table 3.2 demonstrate that the proposed NEC-DGT outperforms other methods in both node and edge attributes prediction and is the

only method to handle both. Specifically, in terms of node attributes, the proposed NEC-DGT get smaller N-MSE value than all the node attributes prediction methods by 85%, 71%, 95% and 95% on average for four dataset respectively. Also, NEC-DGT outperforms the other methods by 46%, 36%, 44% and 58% on average for four dataset respectively on N-R2, N-P, and N-SP. This is because all the node prediction methods only consider a fixed graph topology while NEC-DGT allows the edges to vary. In terms of edges, the proposed NEC-DGT get the highest E-ACC than all the other graph generation methods. It also has higher E-ACC than graph topology translation method: GT-GAN by 7% on average since NEC-DGT considers both edge and node attributes in learning the translation mapping while GT-GAN only considers edges. The proposed NEC-DGT outperforms the NR-DTG by around 3% on average in terms of all metrics, which demonstrates the effectiveness of the graph spectral-based regularization.

Evaluation of the learned translation mapping for synthetic graphs. To evaluate whether the inherent relationship between node and edge (reflected by node degree) attributes is learned and maintained by NEC-DGT, we draw the distributions of the node attribute versus node degree of each node in the generated graphs to visualize their relationship. For comparison, a ground-truth correlation is drawn according to the predefined rule of generating the dataset, namely, each node's degree and attribute follows the function $y = x + 5$. Fig. 3.6 shows four example distributions of nodes in terms of node attributes and degree with the black line as ground-truth. As shown in Fig. 3.6, the nodes are located closely on the ground-truth, especially for the syn-I and syn-IV, where around 85% nodes are correctly located. This is largely because the proposed graph spectral-based regularization successfully discovers the patterns: the densely connected nodes all tend to have large node attributes and in reverse.

3.5.5 Evaluation for malware datasets

Table 3.3 shows the evaluation of NEC-DGT by comparing the generated and real target graphs. For malware graphs, the node attributes are evaluated by N-ACC by calculating

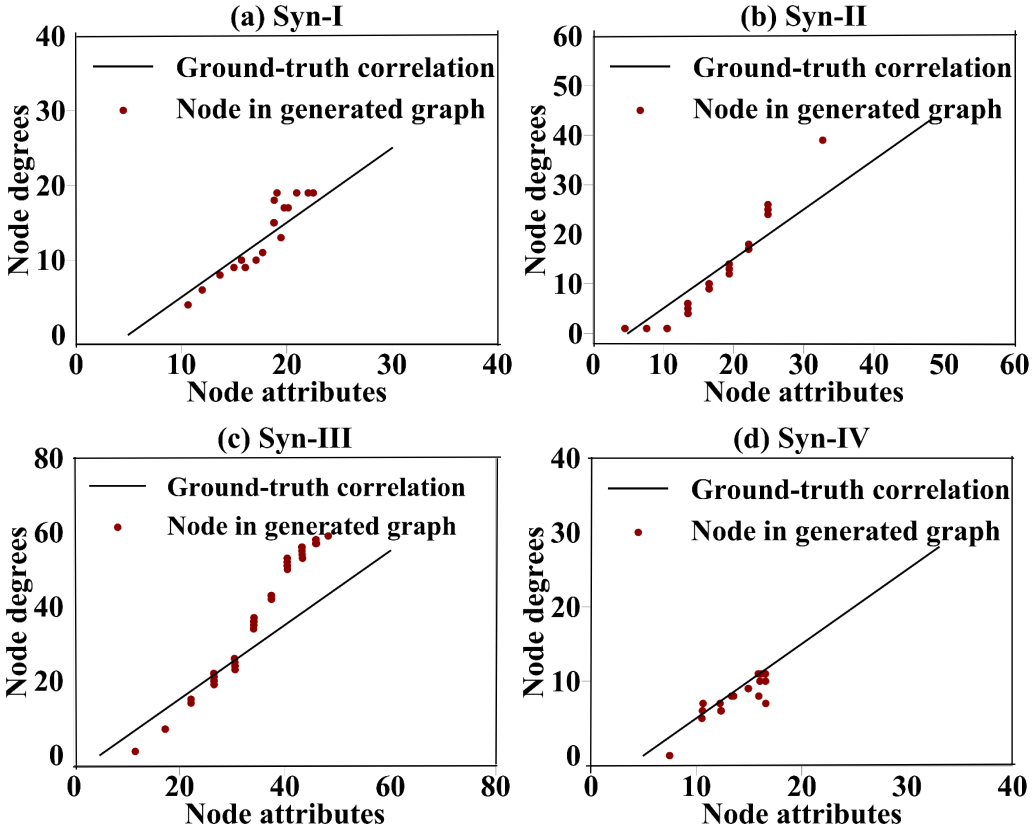


Figure 3.6: Relation visualizations between node attributes and node degrees for samples from four synthetic graphs

the percentage of nodes whose attributes are correctly predicted in all nodes. The edge attributes are continuous value evaluated by E-MSE, E-R2 and E-P. We also use E-Acc to evaluate the correct existence of edges among all pairs of nodes.

The results in Table 3.3 demonstrates that NEC-DGT performs the best for all the three datasets. In terms of E-Acc, the graph generation methods (GraphRNN and GraphVAE) cannot handle the graph translation work and got low E-Acc of around 0.6 at Mal-I,Mal-II, and 0.8 at Mal-III. GT-GAN achieves high E-ACC, but its E-MSE is about 2 folds larger than that of the proposed NEC-DGT on average. NEC-DGT successfully handle the translation tasks with high E-Acc above 0.9, and the smallest E-MSE. In terms of N-Acc, NEC-DGT outperforms other methods by around 5% on the first two datasets. In summary,

Table 3.3: Evaluation of Generated Target Graphs for Malware Dataset (N for node attributes, E for edge attributes, P for Pearson correlation, SP for Spearman correlation and Acc for accuracy).

Malware-I						
Method	E-Acc	E-MSE	E-R2	E-P	Method	N-Acc
GraphRNN	0.6107	1831.43	0.52	0.00	IN	0.8786
GraphVAE	0.5064	2453.61	0.00	0.04	DCRNN	0.8786
GT-GAN	0.6300	1718.02	0.42	0.11	STGCN	0.9232
NR-DGT	0.9107	668.57	**0.82**	**0.91**	NR-DGT	0.9108
NEC-DGT	**0.9218**	**239.79**	0.78	**0.91**	NEC-DGT	**0.9295**
Malware-II						
Method	E-Acc	E-MSE	E-R2	E-P	Method	N-Acc
GraphRNN	0.7054	1950.46	0.44	0.29	IN	0.8828
GraphVAE	0.6060	2410.57	0.73	0.16	DCRNN	0.8790
GT-GAN	0.9033	462.73	0.13	0.81	STGCN	0.9330
NR-DGT	0.9117	448.48	0.68	0.83	NR-DGT	0.8853
NEC-DGT	**0.9380**	**244.40**	**0.81**	**0.91**	NEC-DGT	**0.9340**
Malware-III						
Method	E-Acc	E-MSE	E-R2	E-P	Method	N-Acc
GraphRNN	0.8397	1775.58	0.16	0.23	IN	0.8738
GraphVAE	0.8119	2109.64	0.39	0.32	DCRNN	0.8738
GT-GAN	0.9453	550.30	0.63	0.80	STGCN	**0.9375**
NR-DGT	0.9543	341.10	0.76	0.88	NR-DGT	0.8773
NEC-DGT	**0.9604**	**273.67**	**0.81**	**0.90**	NEC-DGT	0.9002

the proposed NEC-DGT can not only jointly predict the node and edges attributes, but also performs the best in most of metrics. The superiority of NEC-DGT over the NR-DGT in terms of E-MSE demonstrates that the graph spectral-based regularization indeed improve modeling translation mapping.

Case study for malware dataset. Fig. 3.7 investigates three cases of input, real target and generated target graph by NEC-DGT. The green nodes refer to the uncompromised devices while the red nodes refer to the compromised devices. The width of each edge reflects the distance between two devices. In the first case, both in generated and real target graphs, Devices 4 and 6 are restored to normal, while Device 19 get attacked and is isolated from the other devices. It validates that our NEC-DGT successfully finds the

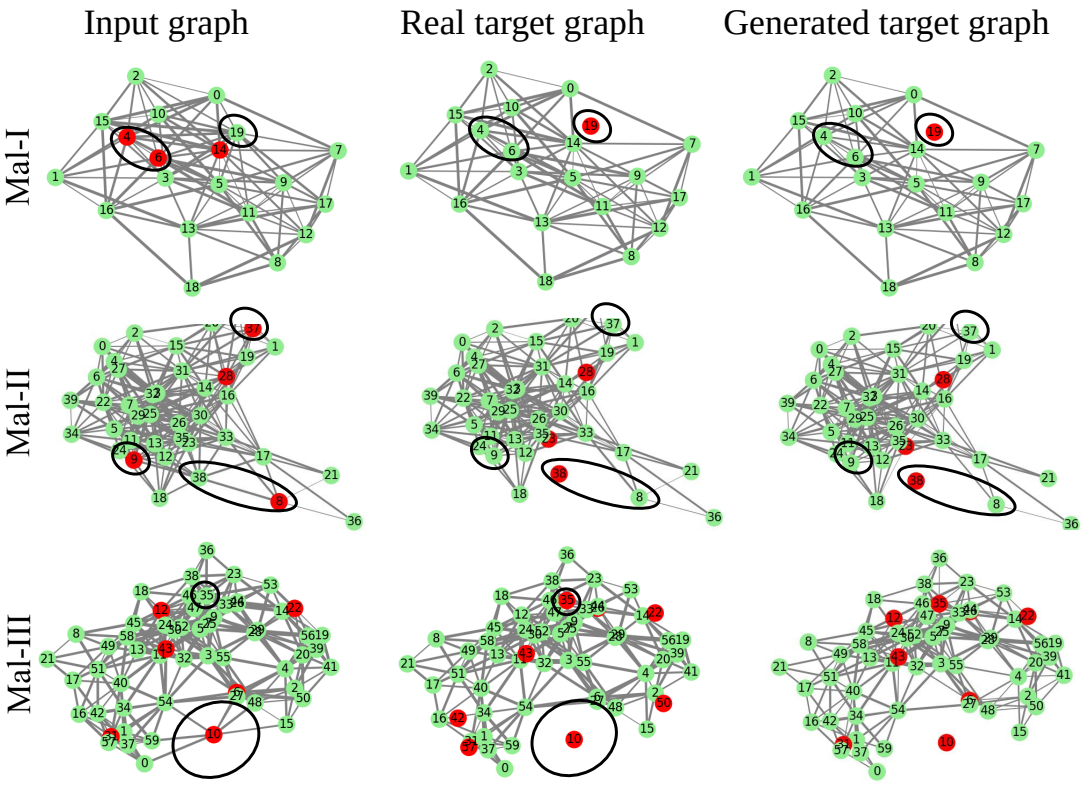


Figure 3.7: Cases of Malware translation by NEC-DGT

rules of translating nodes and performs like the true confinement process. In the second case, Device 8 propagates the malware to Device 38, which is also modeled by NEC-DGT in generated graphs. In addition, the NEC-DGT not only correctly predicts the nodes attributes, but also discovers the change in edge attributes, e.g. in the third case, most of the connections of compromised Device 10 were cut both in generated and real target graphs.

3.5.6 Evaluation for Molecule Reaction datasets

In this task, the NEC-DGT is compared to the Weisfeiler-Lehman Difference Network (WLDN) [114], which is a graph learning model specially for reaction prediction. Table 3.4 shows the performance of our NEC-DGT on the reaction dataset on five metrics, which are the same with the synthetic datasets. The proposed NEC-DGT outperforms both the

Table 3.4: Evaluation of Generated Target Graphs for Molecule Dataset: N for node attributes, E for edge attributes

Method	N-MSE	N-R2	N-P	N-Sp	Method	E-Acc
IN	0.0805	0.46	0.13	0.12	GT-GAN	0.8687
STGCN	0.0006	0.98	**0.99**	0.97	WLDN	0.9667
NR-DGT	0.0008	0.97	**0.99**	**0.99**	NR-DGT	0.9918
NEC-DGT	**0.0004**	**0.99**	**0.99**	**0.99**	NEC-DGT	**0.9925**

translation model GT-GAN and the WLDN by 5% on average. Though the atoms do not change during reaction, we evaluate the capacity of our NEC-DGT to copy the input node features. As shown in Table 3.4, The NEC-DGT get the smallest N-MSE and get higher N-R2 than other comparison methods by around 18%. This shows that our NEC-DGT can deal with a wide range of real-world applications, whether the edges and nodes need change or keep stable.

Metric-based Evaluation for HCP Datasets

We consider two classic brain network prediction methods that use SC to FC [73, 74] as the comparison methods for this experiment. Abdelnour et al. [74] considered the graph spectral transformation kernels by assuming that SC and FC share the identical eigenvectors on their Laplacians. Another method directly considers the graph translation between SC and FC. The goal is to predict the FC given the SC when the brain is doing different tasks.

Table 3.5 shows the Pearson coefficient by comparing the predicted graphs with the empirical target graphs. Our method achieves the highest Pearson coefficient on seven out of eight datasets with superiority of 3.8%, and the highest average Pearson correlation. Specifically, the proposed sign-NEC-DGT outperformed the comparison methods by 5.8% on average in the resting task and 5.4% in Gambling tasks. Also, the proposed sign-NEC-DGT has a 1.3% higher Pearson coefficient than the one without regularization, which demonstrates the effectiveness of the proposed signed graph spectral regularization. This superiority is mainly because the proposed sign-NEC-DGT has the most freedom to learn

Table 3.5: Pearson correlation between the predicted graph and empirical graph on HCP datasets (Res for Resting, Emo for Emotion, Gam for Gambling, Re for Relational)

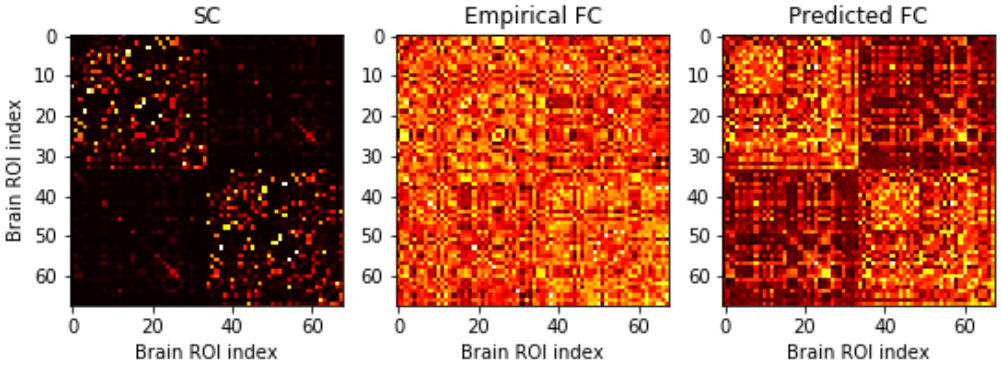
Method	Res	Emo	Gam	Lang	Motor	Re	Social	WM
Ganlan2008	0.41	0.42	0.44	0.44	0.45	0.44	0.45	0.45
Abdelnour2018	0.40	0.41	0.43	0.43	0.44	0.43	0.44	0.45
sign-NEC-DGT(NR)	0.42	**0.43**	0.44	**0.45**	**0.46**	**0.45**	0.45	**0.46**
sign-NEC-DGT	**0.43**	**0.43**	**0.46**	**0.45**	0.45	**0.45**	**0.46**	**0.46**

different parameters for dealing with positive and negative edge attributes.

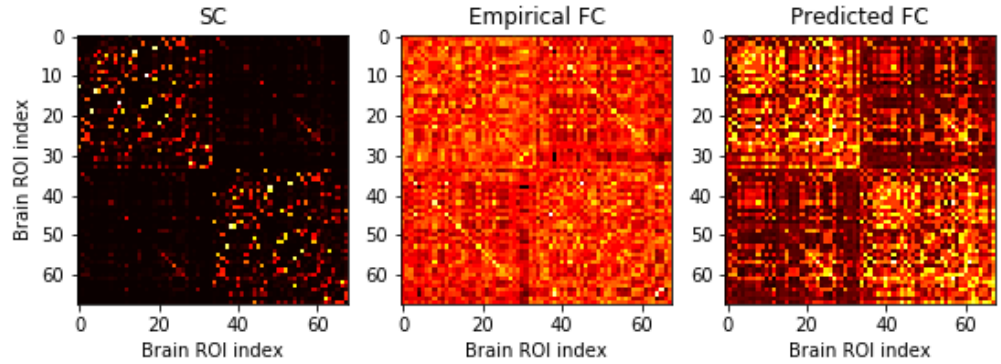
Case Study for HCP Dataset

Figure 3.8 plots two subjects: 1) structural connectivity (i.e., the adjacent matrix of the source graph shown on the left column), 2) empirical functional connectivity (i.e., the adjacent matrix of the target graph shown on the middle column), and 3) predicted functional connectivity (i.e., the adjacent matrix of the target graph shown on the right column).

As shown in Figure 3.8, the predicted FC using Subject 87's SC is very close to the same subject's empirical FC. On the other hand, the predicted FC using Subject 71's SC is different from Subject 87's empirical FC, although Subject 87's SC is very similar to Subject 71's SC. This is because SC reflects the human brain's anatomical neural network, which has relatively fewer individual differences among human beings. Unlike SC, the FC used in this dataset reflects the Pearson correlations between two time series (i.e., Blood Oxygen Level Dependent (BOLD) signal) of different brain regions of interest (ROIs), when the subject is instructed under the resting-state. Practically, it is difficult to control these subjects' brain activities, which causes the empirical FC to be very noisy such that it may affect the performance of all prediction methods.



(a) Subject 87



(b) Subject 71

Figure 3.8: Cases of FC prediction by sign-NEC-DGT

Metric-based Evaluation for Breast Cancer Community Dataset

In this task, the goal is to predict the direct transition network of a user at one health stage given the transition network at another health stage (Dx). Thus, there are five tasks relating to five different target stages. Due to the requirement of handling the directed graphs, we use two comparison methods: GT-GAN [26] and IN [80]. Table 3.6 shows the metric-based evaluation results for this task.

As shown in Table 3.6, the proposed di-NEC-DGT is the only method that not only can handle the node attribute prediction but also can predict the edge attributes for the directed graph. More importantly, it achieves a better performance than the comparison methods for all the different tasks. Specifically, for node prediction tasks, the proposed di-NEC-DGT achieves the a better performance than IN on four metrics on average by about 47.9% in the

Table 3.6: Evaluation of Generated Target Graphs for Breast Cancer Dataset: N for node attributes, E for edge attributes, Chem for Chemotherapy, Ra for Radiation, Hor for Hormonal, Sur for Surgery, Tar for Targeted.

Task	Method	N-MSE	N-R2	N-P	N-Acc	Method	E-Acc
Chem	IN	0.0666	0.31	0.72	0.9167	GT-GAN	**0.9993**
	di-NEC-DGT(NR)	0.0068	0.93	**0.98**	0.9991	di-NEC-DGT(NR)	0.9988
	di-NEC-DGT	**0.0060**	**0.94**	**0.98**	**0.9995**	di-NEC-DGT	0.9988
Ra	IN	0.0667	0.31	0.72	0.9167	GT-GAN	0.9953
	di-NEC-DGT(NR)	0.0076	**0.92**	0.97	**0.9989**	di-NEC-DGT(NR)	0.9981
	di-NEC-DGT	**0.0073**	**0.92**	**0.98**	0.9985	di-NEC-DGT	**0.9982**
Hor	IN	0.0661	0.32	0.73	0.9177	GT-GAN	0.9945
	di-NEC-DGT(NR)	0.0078	0.92	0.97	0.9983	di-NEC-DGT(NR)	**0.9973**
	di-NEC-DGT	**0.0061**	**0.94**	**0.98**	**0.9996**	dir-NEC-DGT	**0.9973**
Sur	IN	0.0661	0.32	0.73	0.9175	GT-GAN	**0.9981**
	di-NEC-DGT(NR)	0.0066	0.93	**0.98**	**0.9992**	dir-NEC-DGT(NR)	**0.9981**
	di-NEC-DGT	**0.0064**	**0.94**	**0.98**	**0.9992**	dir-NEC-DGT	**0.9981**
Tar	IN	0.0671	0.31	0.71	0.9158	GT-GAN	0.9986
	di-NEC-DGT(NR)	0.0205	0.79	0.90	0.9866	di-NEC-DGT(NR)	**0.9989**
	di-NEC-DGT	**0.0073**	**0.92**	**0.98**	**0.9989**	di-NEC-DGT	**0.9989**

Chemotherapy task, 44.8% in the Radiation task, 45.4% in the Hormonal task, 45.3% in the Surgery task, and 47.1% in the Targeted task. The success of the proposed di-NEC-DGT lies mainly on its outgoing and incoming edge parameters and the customized approximation of the direct spectral Laplacian matrix. In addition, for some specific tasks where the edge and node may have many more relations, the spectral regularization term plays a very important role for good prediction. For example, in the Targeted task, di-NEC-DGT has a 17.5% better metric score on average than the same model without regularization.

3.6 Conclusion

This chapter focuses on a new problem: multi-attributed graph translation. To achieve this, we propose a novel NEC-DGT consisting of several blocks which translates a multi-attributed input graph to a target graph. To jointly tackle the different types of interactions among nodes and edges, node and edge translation paths are proposed in each block and

the graph spectral-based regularization is proposed to preserve the consistent spectral property of graphs. Extensive experiments have been conducted on the synthetic and real-world datasets. As the extension of the undirected NEC-DGT, this chapter also proposed the sign-NEC-DGT for the signed graphs and di-NEC-DGT for the directed graphs. Experiment results show that our NEC-DGT can discover the ground-truth translation rules and significantly outperform comparison methods in terms effectiveness. This chapter provides a further step of research for graph transformation problems in more general scenarios. In the next step, We are excited about the prospect of introducing the interpretability of the generation process for explainable deep graph generation.

Chapter 4: Interpretable Deep Graph Generation with Node-edge Co-disentanglement

4.1 Introduction

Recent advances in deep generative models, such as variational auto-encoders (VAE) [20] and generative adversarial networks (GAN) [21], 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 [23, 24, 97]. These involves real-world applications, such as discovering new chemical and molecular structures [36, 37], 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 [25]. 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. 4.1 (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

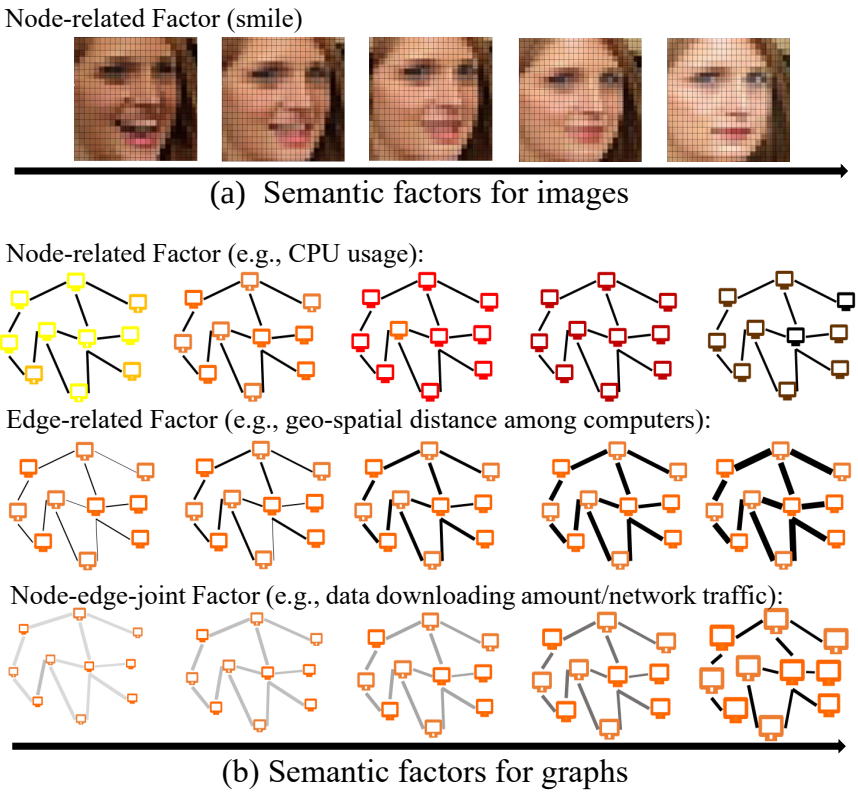


Figure 4.1: Two examples of disentanglement: (a) semantic factors of images, where each pixel is a node and each pixel is connected to its eight neighboring pixels, and (b) semantic factors of cyber networks, where each computer is a node and the link between each pair of computers is an edge (better seen in color).

of Things (IoT). As shown in Fig. 4.1, 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. 4.1 (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. 4.1(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 [115–117], as yet they only have identified the latent factors that caused the edge between a node and its neighbors.

In this chapter, 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. 4.1. 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. 4.1 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) [20,118]. 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.

4.2 Related Works

Disentangled Representation Learning. Disentangled representation learning has gained considerable attention, in particular in the field of image representation learning [119–122]. 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 [123], and can be used to enhance generalizability as well as improve robustness against adversarial attack [120]. The disentangled representations are inherently more interpretable, and can thus potentially facilitate debugging and auditing [124]. This has prompted a number of approaches that modify the VAE objective by adding, removing, or altering the weight of individual terms [120–122, 125–128]. 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) [43, 92], especially graph convolutional networks [99, 129], is attracting considerable attention, because of their remarkable success in multiple domains such as natural language processing [55,130], computer vision [131], software engineering [132] and traffic flow prediction [133]. Self-attention mechanisms and sub graph-level information have also been explored as ways to potentially improve the representation power of learned node embeddings [29, 48, 56]. Most of the existing GNN based graph generation methods are based on VAE [23, 38] and generative adversarial nets (GANs) [49], and others [24, 35]. For example, GraphRNN [35] builds an autoregressive generative model on these sequences utilizing LSTM model and has demonstrated good scalability; while GraphVAE [23] 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 [8] and VGAE [11] 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 [26, 61].

4.3 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. 4.1, 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.

4.4 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.

4.4.1 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.

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. 4.1, where ϵ specifies the strength of the applied constraint:

$$\max_{\theta,\phi} \mathbb{E}_{G\sim D}[\mathbb{E}_{q_\phi(Z|G)} log p_\theta(E, F|z_e, z_f, z_g)]$$

$$s.t. D_{KL}(q_\phi(z_e, z_f, z_g|E, F)||p(z_e, z_f, z_g) < \epsilon. \tag{4.1}$$

Eq. 4.1 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 [119] formulation, which takes the form of the familiar variational free energy objective function:

$$\begin{aligned}\mathcal{L}(\theta, \phi, G, Z, \beta) =& \mathbb{E}_{q_\phi(Z|G)}[log p_\theta(E, F|z_e, z_f, z_g)] \\ & - \beta D_{KL}(q_\phi(z_e, z_f, z_g|E, F)||p(z_e, z_f, z_g).\end{aligned} \tag{4.2}$$

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:

$$q_\phi(z_e, z_f, z_g|E, F) = q_\phi(z_f|F)q_\phi(z_e|E)q_\phi(z_g|E, F) \tag{4.3}$$

$$p_\theta(E, F|z_e, z_f, z_g) = p_\theta(F|z_f, z_g)p_\theta(E|z_e, z_g) \tag{4.4}$$

We can now rewrite the loss function as:

$$\begin{aligned}\mathcal{L}(\theta, \phi, G, Z, \beta) &= \mathbb{E}_{q_\phi(Z|G)}[log p_\theta(F|z_f, z_g)p_\theta(E|z_e, z_g)] \\ &- \beta D_{KL}(q_\phi(z_f|F)q_\phi(z_e|E)q_\phi(z_g|E, F)||p(z_e)p(z_f)p(z_g)) \\ &= \mathbb{E}_{q_\phi(Z|G)}[log p_\theta(F|z_f, z_g)p_\theta(E|z_e, z_g)] - \beta D_{KL}(q_\phi(z_f|F)||p(z_f)) \\ &- \beta D_{KL}(q_\phi(z_e|E)||p(z_e)) - \beta D_{KL}(q_\phi(z_g|E, F)||p(z_g))\end{aligned} \tag{4.5}$$

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

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. 4.2.

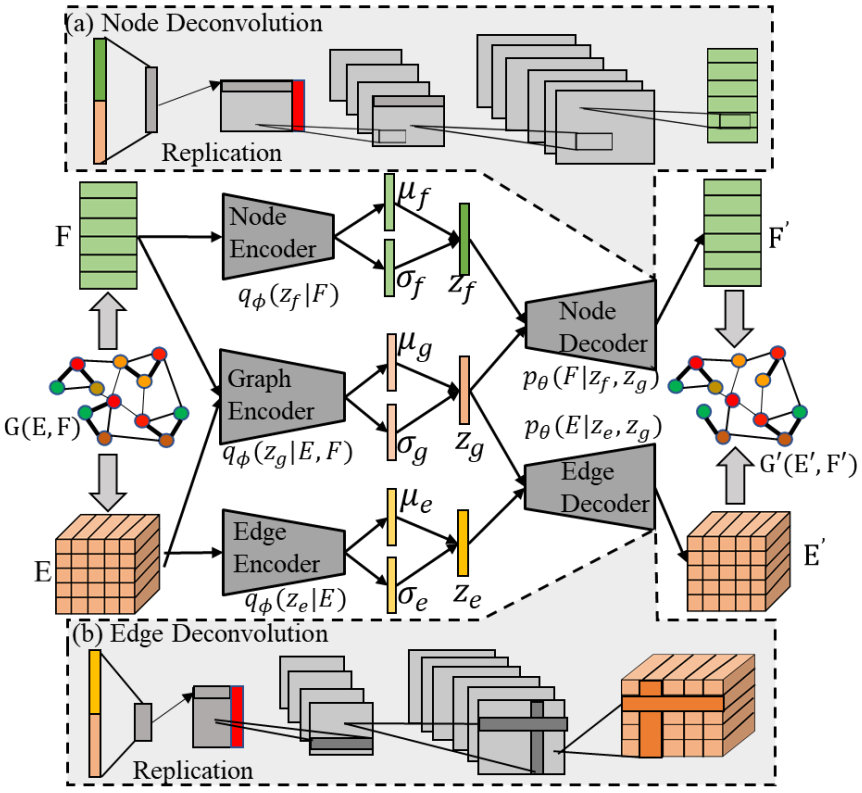


Figure 4.2: The architecture of the proposed NED-VAE consist of three sub-encoders to inference z_e, z_f and z_g, as well as two sub-decoders to reconstruct E and F simultaneously.

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 [26] 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 [97] 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. 4.2 (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. 4.2 (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.

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

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 [26] 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. 4.2 (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.

4.4.2 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. First, 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). Next, 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 (denoted as Ⓐ) for group-wise disentanglement and a variable-wise disentanglement term (denoted as Ⓒ). At last, 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,

Table 4.1: Summary of objectives of the extensions of NED-VAE model. ($\textcircled{C}^*$ refers to the sum of $\textcircled{C}_e$, $\textcircled{C}_f$ and $\textcircled{C}_g$; $\textcircled{2}_e^a$ can be changed to $\textcircled{2}_f^a$ or $\textcircled{2}_g^a$)

NED-VAE	$\textcircled{1}+\textcircled{3}+\beta(\textcircled{2}+\textcircled{4})$
NED-IPVAE-I	$\textcircled{1}+\textcircled{3}+\textcircled{2}+\lambda\textcircled{4}$
NED-IPVAE-II	$\textcircled{1}+\textcircled{3}+\lambda\textcircled{4}$
NED-HCVAE	$\textcircled{1}+\textcircled{3}+\textcircled{2}+\gamma\textcircled{A}$
NED-TCVAE	$\textcircled{1}+\textcircled{3}+\textcircled{2}+\textcircled{C}+\beta\textcircled{A}$
NED-VTCVAE	$\textcircled{1}+\textcircled{3}+\textcircled{2}+\textcircled{C}^*+\beta\textcircled{A}+\gamma_1\textcircled{A}_f+\gamma_2\textcircled{A}_e+\gamma_3\textcircled{A}_g$
NED-AnchorVAE	$\textcircled{1}+\textcircled{3}+\textcircled{2}+\textcircled{4}\text{-}\lambda\textcircled{2}_e^a$

six extensions of the base NED-VAE models are proposed that enforce different terms, as shown in Table 4.1.

Overall graph disentanglement framework

As proved by [128], 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. 4.4, the NED-VAE objective for the graph data can be defined as:

$$
\begin{aligned}
&- D_{KL}(p_\theta(z_e, z_f, z_g, E, F)||q_\phi(E, F, z_e, z_f, z_g)) \\
&= \mathbb{E}_{q_\phi(Z,G)}[log\frac{p_\theta(E, F, z_e, z_f, z_g)}{p_\theta(E, F)p(z_e, z_f, z_g)} + log\frac{q(E, F)q_\phi(z_e, z_f, z_g)}{q_\phi(E, F, z_e, z_f, z_g)} + log\frac{p_\theta(E, F)}{q(E, F)} + log\frac{p(z_e, z_f, z_g)}{q_\phi(z_e, z_f, z_g)}] \\
&= \mathbb{E}_{q_\phi(Z,G)}[log\frac{p_\theta(E, F|z_e, z_f, z_g)}{p_\theta(E, F)} - log\frac{q_\phi(z_e, z_f, z_g|E, F)}{q_\phi(z_e, z_f, z_g)}] - KL(q(E, F)||p_\theta(E, F)) \\
&- KL(q_\phi(z_e, z_f, z_g)||p(z_e, z_f, z_g)) \\
&= \mathbb{E}_{q_\phi(Z,G)}[\underbrace{log\frac{p_\theta(F|z_f, z_g)p_\theta(E|z_e, z_g)}{p_\theta(E, F)}}_{\textcircled{1}} - \underbrace{log\frac{q_\phi(z_e|E)q_\phi(z_f|F)q_\phi(z_g|E, F)}{q_\phi(z_e)q_\phi z_f)q_\phi(z_g)}}_{\textcircled{2}}] \\
&\underbrace{-KL(q(E, F)||p_\theta(E, F)))}_{\textcircled{3}} - \underbrace{KL(q_\phi(z_e, z_f, z_g)||p(z_f)p(z_e)p(z_g))}_{\textcircled{4}}
\end{aligned}
\tag{4.6}
$$

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 $\mathbb{E}_{q(E,F)} log p_\theta(E, F)$; 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 [134]. 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 4.1.

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. 4.1). 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):

$$④ \rightarrow -\mathbb{E}_{q_\phi(Z)}[log\frac{q_\phi(Z)}{q_\phi(z_e)q_\phi(z_f)q_\phi(z_g)} + log\frac{q_\phi(z_e)q_\phi(z_f)q_\phi(z_g)}{p(z_e)p(z_f)p(z_g)} + log\frac{p(z_e)p(z_f)p(z_g)}{p(Z)}] \quad (4.7)$$

$$= -E_{q_\phi(z)}\underbrace{[log\frac{q_\phi(Z)}{q_\phi(z_e)q_\phi(z_f)q_\phi(z_g)}}_{Ⓐ} + \underbrace{log\frac{p(z_e)p(z_f)p(z_g)}{p(Z)}]}_{Ⓑ}$$

$$- \underbrace{D_{KL}(q_\phi(z_e)||p(z_e)) - D_{KL}(q_\phi(z_f)||p(z_f)) - D_{KL}(q_\phi(z_g)||p(z_g))}_{Ⓒ}$$

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 4.1. 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.4.1.

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 [121].

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:

$$-D_{KL}(q_\phi(z_f)||p(z_f)) = -\mathbb{E}_{q_\phi(z_f)}[log\frac{q_\phi(z_f)}{\prod_d q_\phi(z_f^d)} + log\frac{\prod_d q_\phi(z_f^d)}{\prod_d p(z_f^d)} + log\frac{\prod_d p(z_f^d)}{p(z_f)}] \tag{4.8}$$

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

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 [135]. 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. 4.1, can flexibly enforce both the group-wise disentanglement and the variable-wise disentanglement with pre-defined weights.

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. 4.6, we have:

$$
\begin{aligned}
& - log \frac{q_\phi(z_e|E) q_\phi(z_f|F) q_\phi(z_g|E,F)}{q_\phi(z_e) q_\phi(z_f) q_\phi(z_g)} \qquad (4.9) \\
& = \underbrace{log \frac{q_\phi(z_e)}{q_\phi(z_e|E)}}_{②^a_e} + \underbrace{log \frac{q_\phi(z_f)}{q_\phi(z_f|F)}}_{②^a_f} + \underbrace{log \frac{q_\phi(z_g)}{q_\phi(z_g|E,F)}}_{②^a_g} .
\end{aligned}
$$

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 4.1.

Relation to existing models

Next we elaborate the graph version for the remaining disentangle VAE models in Table. 4.1.

First, as a special case of attributed graph, image only involves node attributes and node-related factors matters. Hence in this special case, the NED-VAE objective can be rewritten by ignoring z_e and z_g as:

$$
\mathcal{L}(\theta, \phi, G, Z, \beta) = \mathbb{E}_{q_\phi(Z|F)}[logp_\theta(F|z_f) - \beta D_{KL}(q_\phi(z_f|F)||p(z_f)), \qquad (4.10)
$$

which is the same objective as that defined in β-VAE [119] for the image domain. In the same way, we can easily demonstrated that DIP-VAE [126] is a special case of the the proposed NED-IPVAE-I, obtained by enforcing the inferred priors disentanglement, and InforVAE is a special case of the proposed NED-IPVAE-II.

In addition, the proposed NED-TCVAE is a more general form that includes the objective of two existing methods FactorVAE [122] and β-TCVAE [121] which share the same objectives. For example, when the weight β of Term Ⓐ is 0, and there is no z_e and z_g, there

is no need to enforce the group-wise disentanglement among the edge-latent z_e, node-related latent z_f and node-edge joint latent z_g. Only the variable-wise disentanglement $\mathbb{C}_f$ is used.

4.4.3 Complexity Analysis

The proposed NED-VAE requires $O(N^2)$ operations in time complexity and $O(N^2)$ computation complexity in terms of number of nodes in the graph, which paves the way toward modest scale graphs with hundreds or thousands of nodes, compared to most of the existing graph generation methods, which often have $O(N^3)$ or even $O(N^4)$ computational costs. For example, GraphVAE [23] requires $O(N^4)$ operations in the worst case and [24] uses graph neural networks to perform a form of message passing with $O(MN^2)$ operations to generate a graph.

4.5 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].

4.5.1 Dataset

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 [18]; 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: $F_{i,2} = degree(i) + 10 * c$

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

(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.

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 [19]; 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.

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. To generate the dataset, we simulate the dynamic folding process of a protein peptide with a sequence AGAAAAGA, which for our purposes can be considered as a graph of 8 nodes with node attributes (x, y, z) corresponding to 3D coordination of the C_α atom of each amino acid. The protein contact map (graph topology) is generated based on fully atomistic molecular dynamics simulations. The details of molecule simulation process are provided in Appendix B.3. 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.

4.5.2 Comparison Methods

Since GraphVAE [23] 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 4.1 are all tested and compared.

4.5.3 Evaluation Metrics

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.

Quantitative Metrics

We used four quantitative metrics to evaluate the disentanglement of the proposed models. β-M [119] and F-M [122] measure disentanglement by examining the accuracy of a classifier that predicts the index of a fixed factor of variation; and the modularity score (mod) [136] 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 [137] computes the entropy of the distribution obtained by normalizing the importance of each

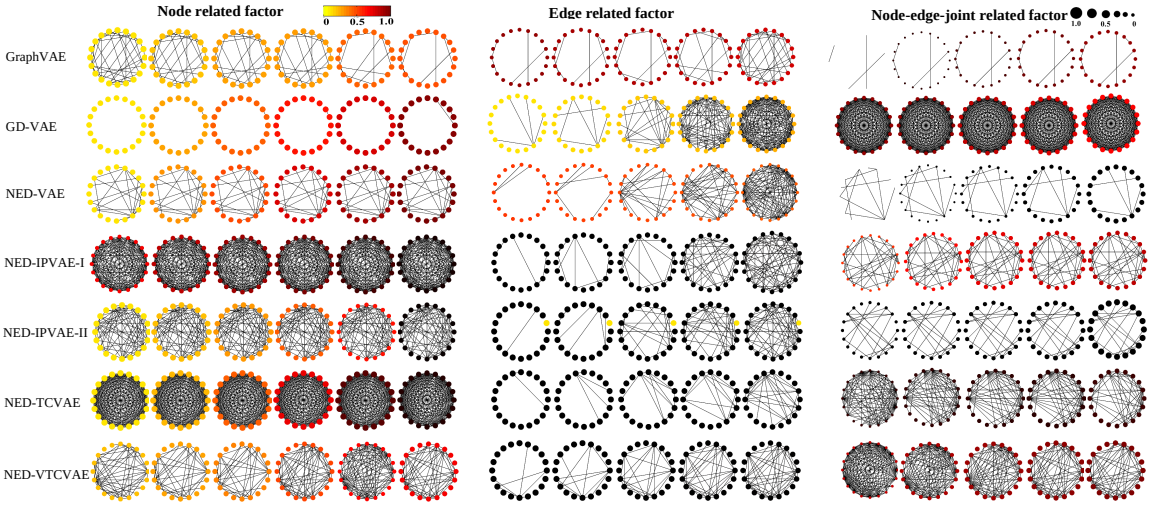


Figure 4.3: Generated Graphs from different models when the related latent variables range from 0 to 10 for ER graphs: (a) node-related factor which is reflected by color of node icon; (b) edge-related factor which is reflected by edge density (c) node-edge-joint related factor which is reflected by the size of node icon.

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 [138].

4.5.4 Results for ER dataset

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. 4.3. 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

Table 4.2: 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

the powerful co-decoder in the generation of both nodes.

Quantitative Evaluation

Four quantitative evaluation metrics are tested on different models and compared in Table 4.2. 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 β-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-IPVAE 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.

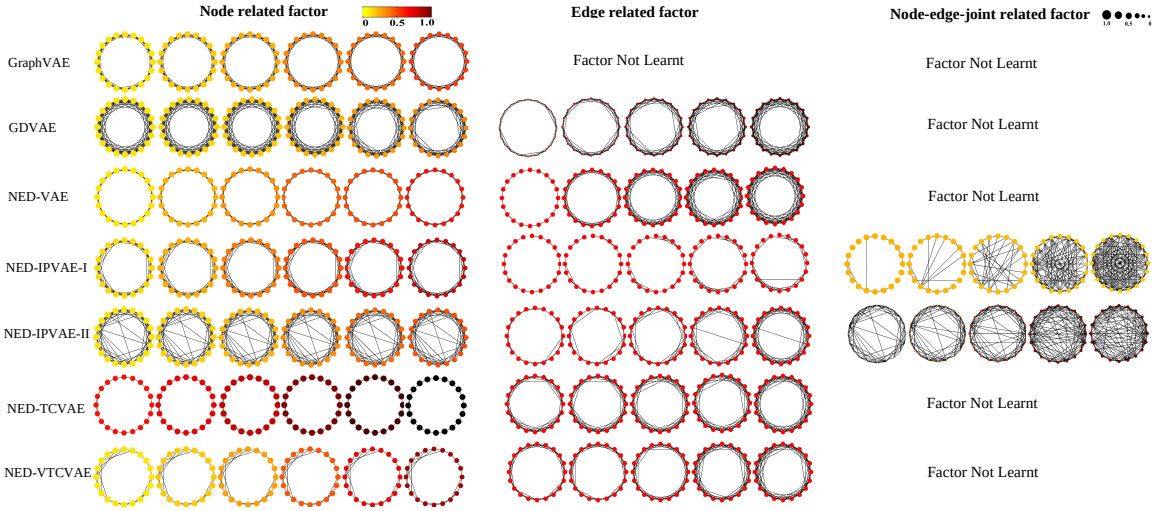


Figure 4.4: Generated Graphs from different graph disentangled models when the related latent variable value ranges in $[0, 10]$ for WS graphs: (a) node-related factor, which is reflected by color of node icon; (b) edge-related factor which is reflected by number of rings in topology and (c) node-edge-joint related factor which is reflected by edge density and the size of node icon.

4.5.5 Results for WR dataset

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. 4.4. 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.

Quantitative Evaluation

Four quantitative evaluation metrics are tested on WS dataset on different models and compared in Table 4.2. 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% regarding β-M, outperforming comparison methods by 26.9% and other proposed extensions by 3.9%. NED-IPVAE-II achieves 97.8% score regarding F-M, outperforming comparison methods by 60.3% and other proposed extensions by 30.8%.

4.5.6 Results for Protein Structure dataset

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. 4.5. 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.

Quantitative Evaluation

Four quantitative evaluation metrics are also tested on protein dataset on different models and compared in Table 4.2. 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.

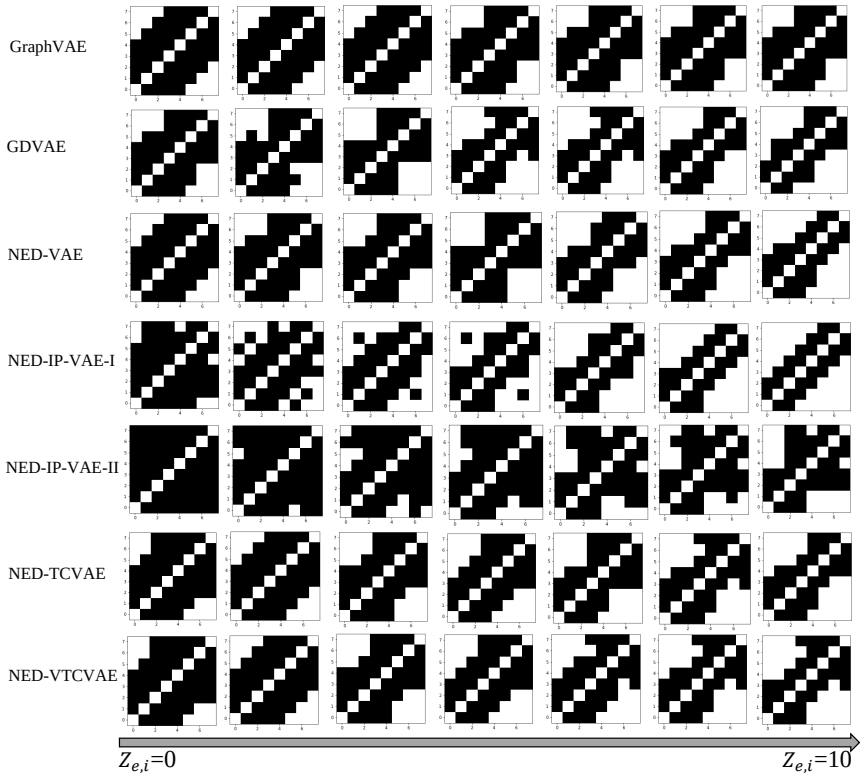


Figure 4.5: Generated contact maps from different models when one edge-related latent variable ranges from 0 to 10 in protein dataset: more blank spaces indicates higher degree of protein folding

4.6 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, which validates the effectiveness of the graph disentanglement architecture and the necessities of separately learning three types of latent representations. In the next step, we would like to further explore how can we precisely control the properties of the generated graphs.

Chapter 5: Property Controllable Deep Graph Generation via Graph Manipulation

5.1 Introduction

Important progress has been made towards learning the underlying low-dimensional representation and generative process of complex high dimensional data such as images [139], natural languages [140], chemical molecules [61, 141] and geo-spatial data [142] via deep generative models. We also have addressed the intepretation for structured data in the last chapter. In recent years, a surge of research has developed new ways to further enhance the disentanglement and independence of the latent dimensions, creating models with better robustness, improved interpretability, and greater generalizability with inductive bias (see Figures 5.1(a) and 5.1(b)) [1–3] or without any bias [121,126,143]. Although it is generally assumed that the complex data is generated from the latent representations, their latent dimensions are typically not associated with physical meaning and hence cannot reflect real data generation mechanisms such as the relationships between structural and functional characteristics. A critical problem that remains unsolved is how to best identify and enforce the correspondence between the learned latent dimensions and key aspects of the data, such as the bio-physical properties of a molecule. Knowing such properties is crucial for many applications that depend on being able to interpret and control the data generation process with the desired properties.

In an effort to achieve this, several researchers [4, 144] have suggested methods that enforce a subset of latent dimensions correspond to targeted categorical properties, as shown in Figure 5.1(c). Though the initial results have been encouraging, critical challenges remain unsolved such as: (1) **Difficulty in handling continuous-valued properties**. The

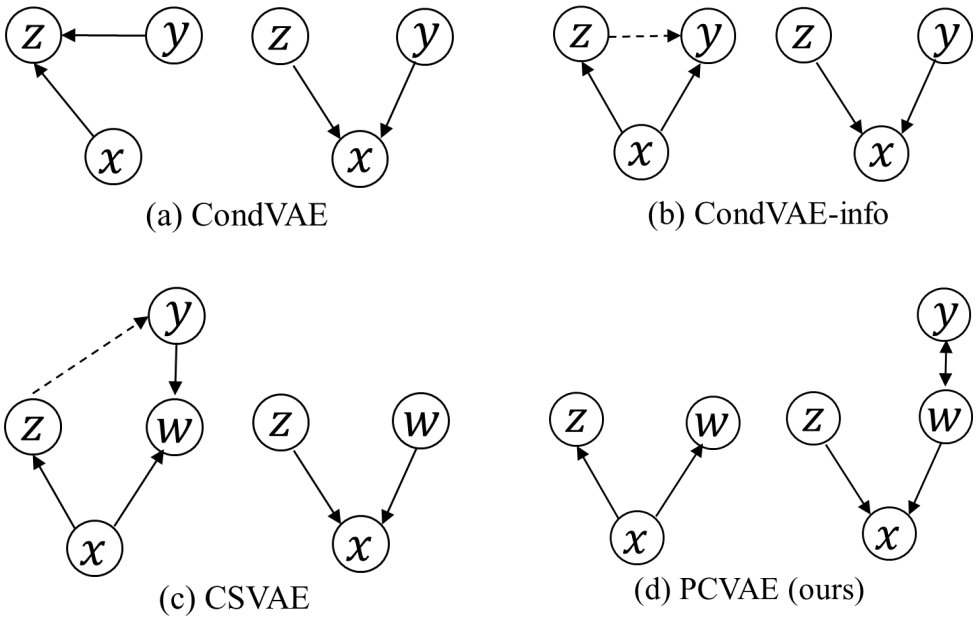


Figure 5.1: While most existing models (e.g., Sub-figures (a) [1,2] and (b) [3]) do not explicitly learn the correspondence between latent dimensions and data properties, some recent work (Sub-figures (c) [4] and (d)) has started to explore this. The generative model (right) and its model inference (left) are shown in each sub-figure. Dotted arrows represent the enforcement of independence and double arrows represent the invertible dependence between two variables. x refers to data, z and w refer to two subsets of latent variables, and y refers to the properties.

control imposed on data generation limits existing techniques to categorical (typically binary) properties, to enable tractable model inference and sufficient coverage of the data. However, continuous-valued properties (e.g., the scale and light level of images) are also common in real world data, while their model inference usually can be easily intractable. Also, many cases require to generate data with properties of which the values are unseen during training process. This cannot be achieved by conventional techniques such as conditional models without making strong assumption on the model distributions. (2) **Difficulty in efficiently enhancing mutual independence among latent variables relevant and irrelevant to the properties**. This problem requires to ensure that each property is only correlated to its corresponding latent variable(s) and independent of all the others. Directly enforcing such mutual independence inherently between all pairs of latent variables incurs quadratic number of optimization efforts. Hence an efficient way is imperative. (3)

Difficulty in capturing and controlling correlated properties. It is feasible that several independent latent variables can capture multiple independent properties. But when the properties are correlated, they cannot be "one-on-one" mapped to corresponding independent latent variables anymore. However, correlated properties are commonly found in formatting a real world data.

To solve the above challenges, we propose a new model, Property-controllable VAE (PCVAE), where a new Bayesian model is proposed to inductively bias the latent representation using explicit data properties via novel group-wise and property-wise disentanglement terms. Each data property is seamlessly linked to the corresponding latent variable by innovatively enforcing an invertible mutual dependence between them, as shown in Figure 5.1(d). Hence, when generating data, the corresponding latent variables are manipulated to simultaneously control multiple desired properties without influencing the others. We have also further extended our model to handle inter-correlated properties. Our key contributions are summarized as follows:

- **A new Bayesian model and derivation process is provided.** A new Bayesian model that inductively biases the latent representation using explicit real data properties is proposed. A variational inference strategy and inference model have been customized to ensure effective Bayesian inference.

- **The disentanglement enforcement terms of structured latent representation is proposed**. Group-wise and property-wise disentanglement terms are proposed to enhance the mutual independence among property, relevant and irrelevant latent variables.

- **A novel a property controllable graph decoder is proposed.** The invertible mutual dependence between property-latent variable pair is achieved by enforcing an invertibility constraint over a residual-based decoder.

- **Comprehensive experiments to validate the effectiveness of the model**. The quantitative and qualitative evaluation performed for this study revealed our PCVAE

outperforms existing methods by up to 28% in capturing and 65% in manipulating the desired properties.

5.2 Related works

Disentanglement Representation Learning. An important relevant area of research is disentangled representation learning [120–122, 143], which structures the latent space by minimizing the mutual information between all pairs of latent variables. The goal here is to learn representations that separate out the underlying explanatory factors that are responsible for variations in the data, as these have been shown to be relatively resilient with respect to the complex variants involved [123,145,146], and thus can be used to enhance generalizability as well as improve robustness against adversarial attack. As noted by [138], it is impossible for disentangled representation learning to capture the desired properties without supervision and inductive biases.

Learning latent representations via supervision. This ensures that the latent variables capture the desired properties though supervision, generally by directly defining properties as latent variables in the model [144]. Unfortunately, apart from providing an explicit variable for the labelled property, this yields no other easily interpretable structures, such as discovering latent variables that are correlated to the properties, as the model proposed in the current study does. This is also an issue with other methods of structuring latent space that have been explored, such as batching data according to labels [2, 147] or using a discriminator network in a non-generative model [148]. Some researchers addressed this problem by introducing the architecture bias through a two-way factored autoencoder and realize the supervision based on a pair-wise contrastive loss [149]. Other researchers addressed this problem by linking latent variables with observed labels through adversarial learning [3, 150–152]. The most relevant work for our purpose is CSVAE [4], where a subset of latent variables are correlated with binary properties via an adversarial learning. All the above works can not handle multiple continuous-valued properties due to their strict

assumptions on the distribution of properties.

Data manipulation and generation. Here, trained machine learning models are utilized to manipulate and generate data in a controllable way with the desired properties, which is especially useful for applications in the image domain. Several works have specifically considered transferring attributes in images, which is the same goal as that in the CASVE. These earlier works [153–155] all transfer attributes from a source image onto a target image. These models can only perform categorical attribute transformation between images (e.g., "splice the beard style of image A onto image B"), but only through interpolation between existing images. Once trained, our proposed model can generate an objects with any value of a certain property (either observed or unobserved during training) that can be encoded in the subset of latent variables.

5.3 Property Controllable VAE

5.3.1 Problem Formulation

Suppose we are given a dataset $\mathcal{D}$ where each data instance is (x, y) with $x \in \mathbb{R}^n$ and $y = \{y_k \in \mathbb{R}\}_{k=1}^K$ to represent K properties of interest of x. For example, if x is a molecule, then we may have properties of interest, such as *cLogP* and *cLogS*. We assume that the data (x, y) are generated by some random process from continuous latent random variables (z, w). Each variable in w controls one of the properties of interest in y, while the variables in z control all the other aspects of x.

Our goal is to learn such a generative model involving (x, y) and (z, w), where the subset of variables (i.e., z) are disentangled from subset w, and the variables inside w are disentangled from each other. Once this model has been learned, then we can expect different elements of variables in w to control different properties of interest, which is a highly desirable goal for many data generation downstream tasks. For example, we may want to decrease the value of a specific property (e.g., protein energy) by changing the value of the corresponding element in w. It is also possible to directly set a desired property value

(e.g., the mass of a molecule) and then generate the corresponding x with this target value (i.e., a molecule with the target mass value).

5.3.2 Overall Objective

In this section, we first introduce the Bayesian variational inference of PCVAE. Then we introduce the group-wise and property-wise disentanglement terms as part of the overall objective. Following this, an invertibility constraint is introduced to enforce mutual dependence between each property-latent variable pair. At last, PCVAE is extended to capture and control multiple correlated properties.

Bayesian Variational Inference of PCVAE

The goal in Section 5.3.1 requires us to not only model the dependence between x and (w, z) for latent representation learning and data generation, but also model the dependence between y and w for property manipulation. We propose to achieve this by maximizing a form of variational lower bound on the joint log likelihood $p(x, y)$ of our model. Given an approximate posterior $q(z, w|x, y)$, we can use the Jensen's equality to obtain the variational lower bound of $p(x, y)$ as:

$$\begin{aligned}\log p(x,y) &= \log \mathbb{E}_{q(z,w|x,y)}[p(x,y,w,z)/q(z,w|x,y)] \\ &\geq \mathbb{E}_{q(z,w|x,y)}[\log p(x,y,w,z)/q(z,w|x,y)]. \end{aligned} \tag{5.1}$$

The joint likelihood $\log p(x, y, w, z)$ can be decomposed as $\log p(x, y|z, w) + \log p(z, w)$. We have two assumptions: (1) w only encodes the information from y, namely, x and y are conditionally independent given w (i.e., $x \perp y|w$); (2) z is independent from w and y, namely $z \perp w$ and $z \perp y$, which is equal to $y \perp z|w$ (see derivation in Appendix C.1.2). First, based on the two assumptions, we can get $x \perp y|(z, w)$ (see derivation in Appendix C.1.3). Thus, we have $\log p(x, y|z, w) = \log p(x|z, w) + \log p(y|z, w)$. Then, based on the assumption $y \perp z|w$, we can have $\log p(y|z, w) = \log p(y|w)$. Then we get

$\log p(x, y|z, w) = \log p(x|z, w) + \log p(y|w)$. To explicitly represent the dependence between x and (z, w) as well as the dependence between y and w, we can parameterize the joint log-likelihood as $\log p_{\theta,\gamma}(x, y, w, z)$ with θ and γ as:

$$\log p_{\theta,\gamma}(x, y, w, z) = \log p_\theta(x|z, w) + \log p(z, w) + \log p_\gamma(y|w). \tag{5.2}$$

Given the condition that a parameterized $q_\phi(z, w|x, y) = q_\phi(z, w|x) = q_\phi(z|x)q_\phi(w|x)$ (since the information on y is included in x), by taking Eq. 5.2 into the above variational lower bound term in Eq. 5.1, we obtain the negative part as an upper bound on $-\log p_{\theta,\gamma}(x, y)$ (as shown in the right sub-figure of Figure 5.1(d)):

$$\mathcal{L}_1 = -\mathbb{E}_{q_\phi(z,w|x)}[\log p_\theta(x|z, w)] - \mathbb{E}_{q_\phi(w|x)}[\log p_\gamma(y|w)] + D_{KL}(q_\phi(z, w|x)||p(z, w)) \tag{5.3}$$

This gives us the proposed Bayesian variational inference of PCVAE. The detailed derivation of Eq. 5.3 can be found in Appendix C.1.1. As there are K properties of interest in y which are assigned and disentangled by the latent variables in w, the second term in Eq. 5.3 can be detailed as $\sum_k^K \mathbb{E}_{q_\phi(w|x)}[\log p_\gamma(y_k|w_k)]$.

Group-wise and Property-wise Disentanglement

Considering that the above derivation is conditional on two requirements: (1) z is independent from w and y and (2) the variables in w are independent from each other, while in practice minimizing the above objective $\mathcal{L}_1$ will not imply that our model will satisfy these conditions. We therefore propose to further penalize the novel *Group-wise-* and *Property-wise Disentanglement* terms.

We first decompose the KL (Kullback-Leibler) divergence term in Eq. 5.3 as:

$$\mathbb{E}_{p(x)}[D_{KL}(q_\phi(z,w|x)||p(z,w))]$$

$$= D_{KL}(q_\phi(z,w,x) \parallel q(z,w)p(x)) + \underbrace{D_{KL}(q(z,w) \parallel \prod_{i,j} q(z_i)q(w_j))}_{\textit{total correlation term}}$$

$$+ \sum_i D_{KL}(q(z_i) \parallel p(z_i)) + \sum_j D_{KL}(q(w_j) \parallel p(w_j)) \tag{5.4}$$

The second term in the right of the above equation is referred to as the total correlation (TC), as proposed by Chen et al. [121], which is one of many generalizations of mutual information to more than two random variables. The detailed derivation of this decomposition can be found in Appendix C.1.1. The penalty on this TC term forces the model to find statistically independent factors in the data distribution. A heavier penalty on this term induces a more disentangled representation among all variables in both z and w, but as stated in our problem formulation, we only require that (1) variables in w are disentangled to capture different properties, and (2) although z is disentangled from w, the latent variables inside z do not need to be disentangled from each other. Roughly enforcing the disentanglement between all pairs of latent variables in w and z, as done by the existing TC term, can incur at least quadratic number of redundant optimization efforts and could lead to poor convergence. Thus, we further decompose and analyze the TC term as:

$$D_{KL}(q(z,w) \parallel \prod_{i,j} q(z_i)q(w_j)) = \underbrace{D_{KL}(q(z,w) \parallel q(z)q(w))}_{\textit{group-wise disentanglement}} + \underbrace{D_{KL}(q(w) \parallel \prod_i q(w_i))}_{\textit{property-wise disentanglement}}$$

$$+ D_{KL}(q(z) \parallel \prod_i q(z_i)). \tag{5.5}$$

The first term in the right part of above decomposition enforces the independence between the two subsets of latent variables z and w, which we term *group-wise disentanglement*. The second term enforces the independences of variables inside w, ensuring that each latent variable can only capture the information of the single property assigned to it. We

term this *property-wise disentanglement*. Imposing a heavy penalty on these two terms can satisfy the two requirements mentioned above. We can now obtain the second part for the objective of PCVAE by introducing the coefficient ρ as:

$$\mathcal{L}_2 = D_{KL}(q(z,w)||q(z)q(w)) + \rho D_{KL}(q(w)||\prod_i q(w_i)) \tag{5.6}$$

Invertible Constraint for Property Control

As stated in the problem formulation, an important goal for our new model is to generate a data point x that retains the original property value of a given property y_k with great precision. More importantly, there should be no strict assumptions of parameters for $p(y_k)$ and $q(w_k|y_k)$. The most straightforward way to do this is to model both the mutual dependence between y_k and its relevant latent variable w_k, namely, $q(w_k|y_k)$ and $p(y_k|w_k)$, which, however, could incur double errors in this two-way mapping. To address it, we innovatively propose instead an invertible function that mathematically ensures the exact recovery of w_k given y_k based on the following deduction.

In the above objective, we only explicitly model the conditional distribution of $p_\gamma(y_k|w_k)$, hence, to achieve the precisely control of property via z and w, which is necessary to generate x with a certain property $y_k = m$, we need to maximize the probability that $y_k = m$ as follows:

$$x \sim p_\theta(x|z,w), \quad z,w \leftarrow \arg\max_{z\sim p(z),w\sim p(w)} p_\gamma(y_k = m|z,w), \tag{5.7}$$

which is equal to:

$$x \sim p_\theta(x|z,w),\ z \sim p(z),\ w_{j,j\neq k} \sim p(w_j), \quad w_k \leftarrow \arg\max_{w_k\sim p(w_k)} p_\gamma(y_k = m|w_k) \tag{5.8}$$

where w_k can be determined as follows ($\mathcal{N}$ in the followings denotes Gaussian distribution):

$$\begin{aligned} & w_k \leftarrow \arg\max_{w_k \sim p(w_k)} p_\gamma(y_k = m | w_k) \\ & = w_k \leftarrow \arg\max_{w_k \sim p(w_k)} \log \mathcal{N}(y_k = m | f_k(w_k; \gamma), \sigma_k) \\ & = w_k \leftarrow \arg\max_{w_k \sim p(w_k)} -(m - f_k(w_k; \gamma))^2 = w_k \leftarrow {f_k}^{-1}(m) \end{aligned} \tag{5.9}$$

Therefore, by learning an invertible function $f_k(w_k; \gamma)$ from w_k to the expectation of y_k to model $p_\gamma(y_k|w_k)$, we can easily achieve the desired precise control of the property. The above derivation are based on the assumption that y is a continuous-value. It can also be extended into the situation when y is discrete property, as detailed in Appendix 5.3.2. To learn an invertible function $f_k(w_k; \gamma)$, we propose to leverage an invertible neural network. Inspired by the invertible ResNet [156], we decompose the function $f_k(w_k; \gamma)$ as $f_k(w_k; \gamma) = \bar{f}_k(w_k; \gamma) + w_k$. As proved by [156], the sufficient condition that $f_k(w_k; \gamma)$ is invertible is that $Lip(\bar{f}_k) < 1$, where $Lip(\bar{f}_k)$ is the Lipschitz-constant of $\bar{f}_k(w_k; \gamma)$.

Thus, the overall objective of the proposed PCVAE is finally formalized as: $\min_{\theta,\phi,\gamma} \mathcal{L}_1 + \alpha \mathcal{L}_2$ with subject to $Lip(\bar{f}_k) < 1$ for all $k \in K$, where α is the coefficient parameter.

Remark 1 (Monotonic relationship of property-latent variable pair)**.** *Given the condition that $f_k(w_k; \gamma)$ is invertible and continuous ($Lip(\bar{f}_k)$ is less than 1), $f_k(w_k; \gamma)$ is thus a monotonic function. This is very important to increase (or decrease) the value of property y_k by increasing (or decreasing) w_k, especially when the desired value of property is not available.*

Extension of the invertible function to discrete-valued Properties

Here we consider the situation when property y is discrete-valued and we denote $y_k = \{0, 1\} \in \mathbb{R}^C$ as a one-hot vector here to represent its category and C is the number of categories. In the overall objective, we only explicitly model the conditional distribution

of $p_\gamma(y_k|w_k)$, hence, to achieve the precisely control of property via z and w, which is necessary to generate x with a certain property y_k that belong to the mth category, we need to maximize the probability that $y_k = M$ as follows:

$$x \sim p_\theta(x|z,w), \qquad z,w \leftarrow \arg\max_{z\sim p(z),w\sim p(w)} p_\gamma(y_k = M|z,w), \tag{5.10}$$

where M is also an one-hot vector with $M[m] = 1$. Then the above equation is equal to:

$$x \sim p_\theta(x|z,w),\ z \sim p(z),\ w_{j,j\neq k} \sim p(w_j), \quad w_k \leftarrow \arg\max_{w_k\sim p(w_k)} p_\gamma(y_k = M|w_k) \tag{5.11}$$

where w_k can be determined as follows based on cross entropy objective (**Cat** in the followings denotes categorical distribution):

$$\begin{aligned}
& w_k \leftarrow \arg\max_{w_k\sim p(w_k)} p_\gamma(y_k = M|w_k) \\
&= w_k \leftarrow \arg\max_{w_k\sim p(w_k)} \log \mathbf{Cat}(y_k = M|\nu_k(w_k;\gamma)) \\
&= w_k \leftarrow \arg\max_{w_k\sim p(w_k)} \log \frac{exp(\nu_k(w_k;\gamma)[m])}{\sum_j^C exp(\nu_k(w_k;\gamma)[j]} \\
&= w_k \leftarrow \nu_K^{-1}(M)
\end{aligned} \tag{5.12}$$

Generalization of handling correlated properties

As stated in the third challenge in Section 5.1, there are usually several groups of properties involved in describing the data x and each group has several properties. These different groups are independent, but the properties within the same group are correlated. Thus, we can further generalize the above objective framework to handle the correlated properties inside the same group.

The notation for y_k is extended to $y_k = \{y_{j,k} \in \mathbb{R}\}_{j=1}^{M_k}$, signifying that there are M_k correlated properties inside the k-th property group. The properties inside the same property

set y_k are correlated, while the different property sets (e.g. y_p and y_k) are independent. Similarly, the notation for w_k is extended as a group of latent variables to control the corresponding property set. For the properties inside the same group, we assume all depend on the same group of latent variables w_k as

$$p(y_{j,k}|w_k) = \mathcal{N}(y_{j,k}|f_k(w_k;\gamma)[j], \sigma_{j,k})), \tag{5.13}$$

where $f_k(w_k;\gamma)[j]$ denotes the j-th element of the output of $f_k(w_k;\gamma)$. Thus, the second term in Eq. 5.3 can be generalized as $\sum_k^K \sum_j^{M_k} \mathbb{E}_{q_\phi(w|x)}[\log p_\gamma(y_{j,k}|w_k)]$.

5.3.3 Neural Network Architecture of PCVAE

As shown in Figure 5.1 (d), there is an encoder (left-hand side of Figure 1(d)) that models the distribution $q(z, w|x)$ and two decoders (right-hand side of Figure 1(d)) that model the distribution $p(y|w)$ and $p(x|z, w)$. To implement the encoder and decoders in the first objective (i.e., $\mathcal{L}_1$), we use Multi-perceptions (MLPs), Convolution Neural Networks (CNNs) or Graph Neural Networks (GNNs) to represent the distributions over relevant random variables.

To implement the second part $\mathcal{L}_2$, it is necessary to calculate the group-wise and property-wise disentanglement terms. Noting that the calculation of the density $q(z)$, $q(w)$ and $q(w_i)$ in group-wise and property-wise disentanglement terms depends on the entire data space. As such, it is inaccessible to compute it exactly during training. Thus, as the same operation conducted by Chen et al. [121], we utilize the Naïve Monte Carlo approximation based on a mini-batch of samples to underestimate $q(z)$, $q(w)$ and $q(w_k)$. The detailed operation is described in Appendix C.2.3.

To implement the invertible constraint and model the distribution of $p_\gamma(y_k|w_k)$, we utilize MLPs to model the function $\bar{f}_k(\cdot)$. Since the function $\bar{f}_k(\cdot)$ modeled by MLPs is a composition of contractive nonlinearities (e.g., ReLU, ELU, tanh) and linear mappings, based on the definition of Lipschitz-constant we have $Lip(\bar{f}_k) < 1$ if $\| W_l \|_2 < 1$ for $l \in L$,

where W_l refer to the weights of the l-th layer in $\bar{f}_k$, $\|\cdot\|_2$ denotes the spectral norm, and L refers to the number of layer in the MLPs. To realize the above constraint on weights of neural networks, we propose to use the spectral normalization for each layer of MLPs, as introduced by Behrmann et al. [156].

5.3.4 Precisely Property Controllable Generation

Our proposed model can be applied to an important downstream task, namely precisely property controllable generation. Given the value of a required property y_k, the goal of property controllable generation is to generate a data x which holds the same value as this desired property. To achieve this, three steps are conducted: (1) infer the value of w_k based on the well-trained neural network $\bar{f}_k(\cdot)$ and the given property y_k via fixed-point iteration by following $w_k^{i+1} := y_k - \bar{f}_k(w_k^i; \gamma)$, where w_k^i is the updated latent variable at the i-th iteration step and $w_k^0 = y_k$; (2) randomly sample the values of z and the remaining variables in w from their prior distributions (i.e., Gaussian distribution) to obtain all the latent variables; and (3) generate a data x using the decoder based on the latent variables that are inferred from the previous two steps.

5.4 Experiment

This section reports the results of the qualitative and quantitative evaluation carried out to test the performance of the proposed model on two datasets in two domains, namely images and molecules. All experiments were conducted on a 64-bit machine with an NVIDIA GPU (GTX 1080 Ti, 11016 MHz, 11 GB GDDR5). The architectures and hyper-parameters can be found in Appendix C.2.[1].

[1]The code for the proposed PCVAE is available at: `https://github.com/xguo7/PCVAE`

5.4.1 Experiment Setup

Datasets

The **dSprites** dataset [157] consists of 2D shapes procedurally generated from ground truth independent semantic factors. The factors that are explored as properties of data in this experiment are *scale*, and the x and y positions (mentioned as *x_pos* and *y_pos*) of a sprite. All possible combinations of these semantic factors are used for generating a total of $730k$ images, where $580k/146k$ is the training/testing set split.

The **3Dshapes** dataset [158] consists of 3D shapes procedurally generated from ground truth independent semantic factors. The factors that are explored as properties of data in this experiment are *wall hue*, *floor hue* and *scale*. All possible combinations of these semantic factors are used for generating a total of $480k$ images, where $390k/90k$ is the training/testing set split.

The **QM9** dataset [159] consists of $134k$ stable small organic molecules, where $120k/20k$ is the training/testing set split.

Comparison Methods

In order to validate the superiority of our proposed model in capturing and manipulating the property during generation, we compare the performance of PCVAE to those achieved by three comparison models that are most relevant to our problem: (1) *semi-VAE* [144] is a semi-supervised model that enforces the value of each latent variable to be equal to the value of each property. Here we utilize all the labels for supervision for fairness; (2) *CSVAE* [4] is a VAE-based model that utilizes mutual information minimization to learn latent dimensions associated with only binary properties. Here we adjust this model to handle continuous property by assuming a Gaussian distribution; and (3) *PCVAE_tc* is a baseline model that holds the same inference model and property controlling strategy as PCVAE, of which the proposed group-wise and property-wise disentanglement terms are replaced with the TC term, namely, $D_{KL}(q(z,w)||\prod_{i,j} q(z_i)q(w_j))$, as proposed in β-*TCVAE* [121]. It is used as

an ablation study to validate the effectiveness of the proposed group-wise and property-wise disentanglement. (4) *PCVAE_nsp* is a baseline model that holds the same architecture and disentanglement terms as PCVAE except for the spectral normalization. It is used as an ablation study to validate the effectiveness of spectral normalization.

5.4.2 Evaluation for Disentangled Latent Variables

In this section, we explore (1) whether each variable w_k successfully captures the information of the property that is assigned to it through supervision, and (2) whether the subset z of latent variables is independent from the properties.

Quantitative evaluation. We calculate the normalized mutual information between each encoded latent variable w_k and the property y_k that is assigned to it, as well as the average mutual information between latent z and each y_k. Figure 5.2 shows the mutual information heat map by each model on *dSprites*. The element in the row of z_{avg} denotes to the average of all the mutual information between z and each property.

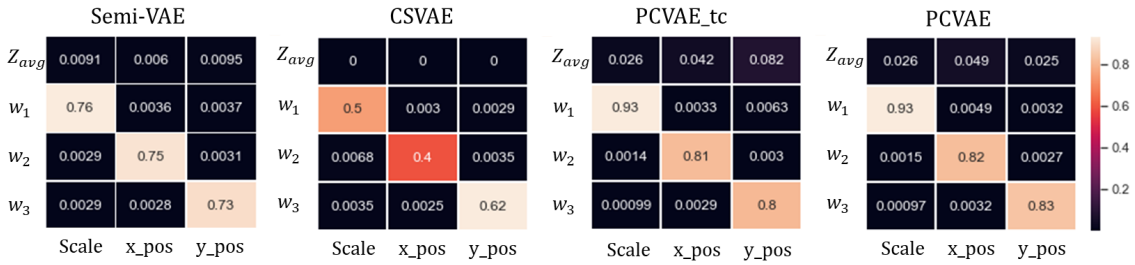


Figure 5.2: Heat-maps of the mutual information between latent variables and three properties by each model for the dSprites dataset. Data value in each cell denotes the normalized mutual information.

We also calculate the mutual information between each encoded latent variable w_k and the property y_k that assigned to it, as well as the average mutual information between latent z and each y_k, as shown in Figure 5.3 for the results on molecule QM9 dataset. For this difficult task in the molecule domain which contains the implicit properties, the proposed

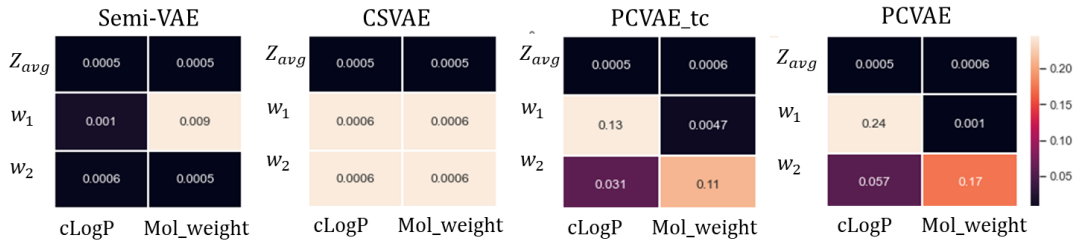


Figure 5.3: Heat-maps of the mutual information between latent variables and two properties by each model for the molecule QM9 dataset. Data value in each cell denotes the normalized mutual information.

PCVAE still shows significant advantage in capturing the *Mol_weight* and *cLogP* properties.

In addition, we utilize the metric $avgMI$[2] proposed by Locatello et al. [144] to show an overall quantitative comparison between different methods, as shown in Table 5.1. The proposed PCVAE achieves the least avgMI of 0.257, which demonstrate its strength in enforcing the relationship between w_k and y_k. These results also validate the effectiveness and necessity of the proposed group-wise and property-wise disentanglement term, as PCVAE outperforms the baseline model PCVAE_tc on avgMI by around 28%. Though CSVAE shows an good performance in disentangling z and w, its latent variables in w turn in a poor performance for capturing the properties. Similar conclusions can also be drawn from the results on the 3Dshapes and QM9 dataset. For example, PCVAE outperforms the comparison models by about 16% in capturing two independent properties *cLogP* and *Molweights*.

Qualitative evaluation. We also qualitatively evaluate the dependence of each latent variable and its relevant property by visualizing the variation of the properties when traversing the priors of each latent variable. As shown in Figure 5.4, as the values of w_1, w_2 and w_3 change between $(-0.5, 0.5)$, the continuous variations of the assigned properties of *scale*, *x_position* and *y_position* of the generated images are clearly visible (as highlighted in red rectangle). The variables in $z = \{z_1, z_2, z_3\}$ have almost no influence on these properties,

[2] $avgMI = \| I(w,y) - E(k)) \|_F^2$, where k is the number of properties. $I(w,y)$ is mutual information matrix. The details can be found in Appendix C.2.4

Table 5.1: The avgMI achieved by each model on the dSprites and QM9 datasets.

Methods	dSpirites	3Dshapes	QM9
Semi-VAE	0.439	0.115	1.413
CSVAE	0.868	1.348	1.411
PCVAE_tc	0.285	0.018	1.245
PCVAE_nsp	0.266	0.031	1.162
PCVAE	**0.257**	**0.016**	**1.125**

Table 5.2: MSE between the expected and actual property of the generated molecules (PCVAE (cor) denotes the extended model for correlated properties).

Model	cLogP	Molweights	cLogS
Semi-VAE	1.40	122.34	N/A
CSVAE	4.69	180.92	N/A
PCVAE_tc	5.02	131.15	N/A
PCVAE_nsp	1.81	176.94	N/A
PCVAE	**1.29**	**87.62**	N/A
PCVAE (cor)	**1.33**	**53.49**	**1.15**

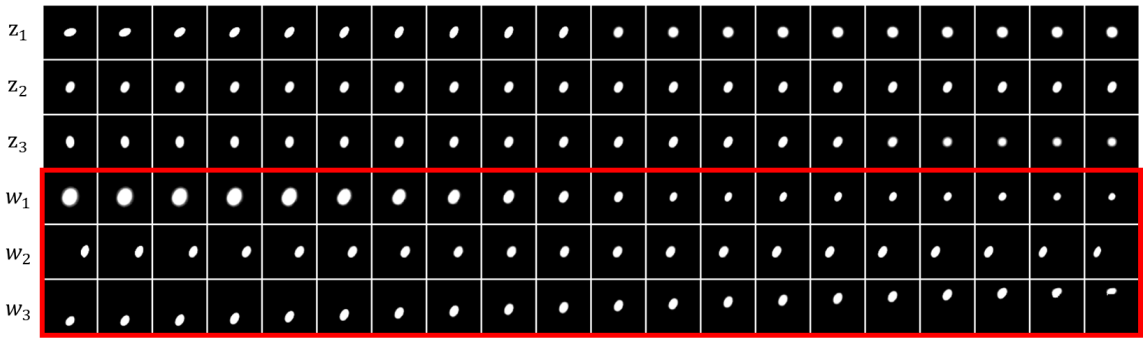


Figure 5.4: The generated images by PCVAE when traversing three latent variables in subset z (upper 3 rows) and three latent variables in subset w (bottom 3 rows) for the dSprites dataset.

which validates the effectiveness of the group-wise disentanglement term.

We also provide the qualitative evaluation on the comparison experiments when traversing the values of latent variables in Fig. 5.5. As shown here, the latent variables w learned by the baseline model PCVAE_tc could successfully capture the three properties, which validate the effectiveness of the proposed overall inference model. The latent variables w_2 and w_3 learned by CSVAE can capture the *x_pos* and *y_pos* properties, while w_1 fails in capture the *scale* property. Semi-VAE can capture the three properties but the quality of the generated images is very bad and biased.

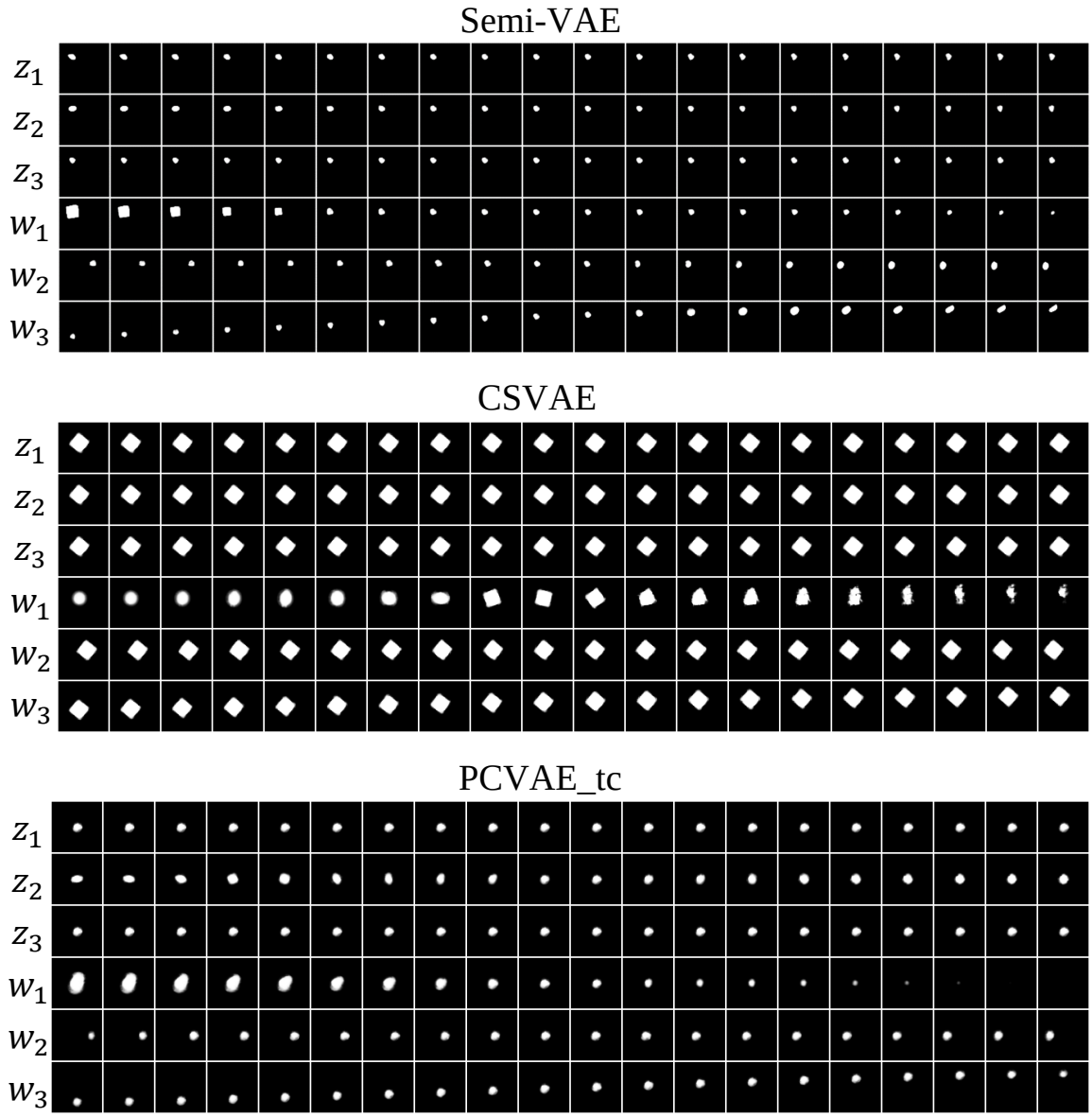


Figure 5.5: The generated images by comparison models when traversing on three latent variables in subset of latent z (upper3 rows) and three latent variables in subset of latent w (bottom 3 rows) for the Dsprits dataset

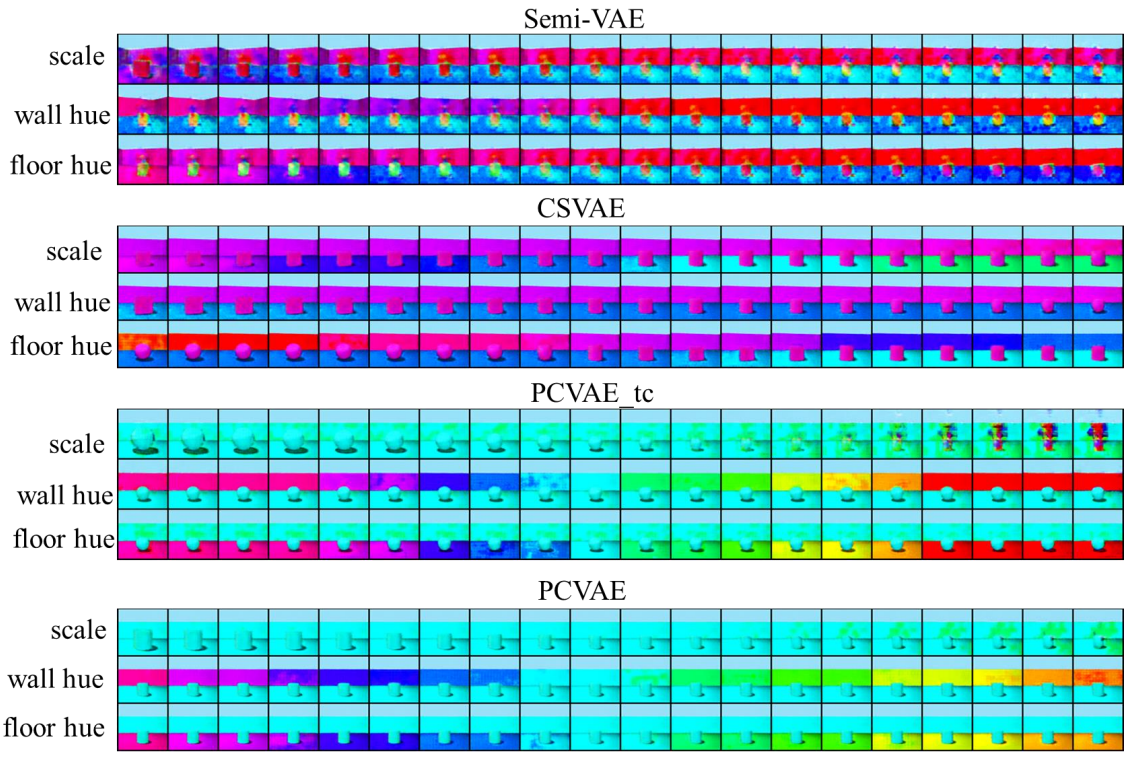


Figure 5.6: The generated images by PCVAE and comparison methods when traversing on three latent variables in subset of latent w (bottom 3 rows) for the 3Dshapes dataset

Next, we provide the qualitative evaluation on the 3Dshapes when traversing the values of each latent variables w in Fig. 5.5. As shown here, the latent variables w learned by the proposed model PCVAE could successfully capture the three properties, object scale, wall hue and the floor hue in the images, which validate the effectiveness of the proposed overall inference model.

We also qualitative evaluate the relationship of each latent variable and its relevant properties. We visualize the variation of the properties on QM9 datasets, when traversing on the priors of each latent variable, as shown in Figure 5.7. The variable w_1 and w_2 could successfully capture the properties *Molweight* and *cLogP*.

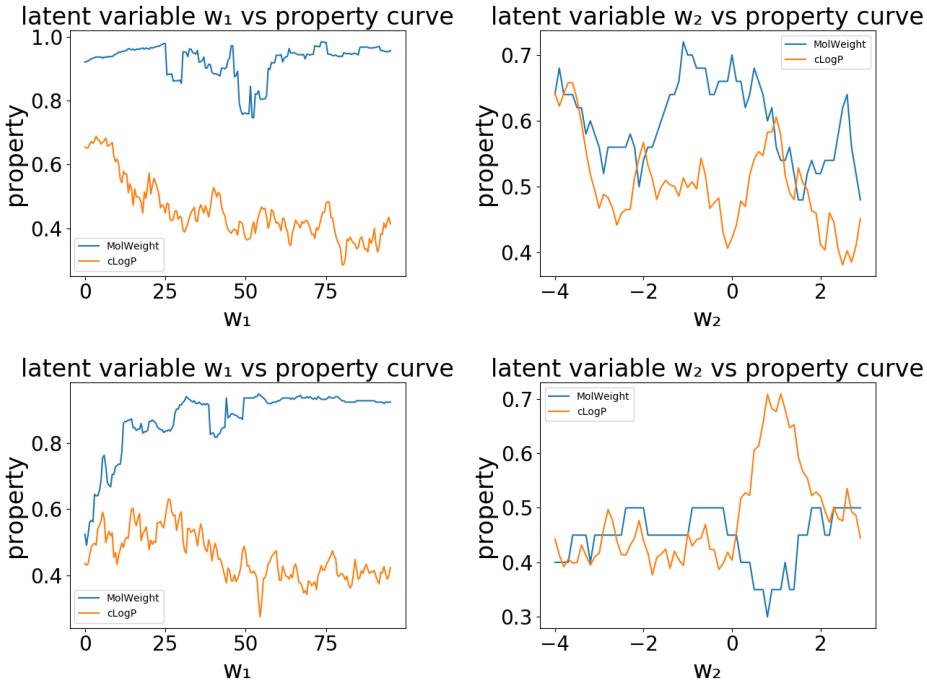


Figure 5.7: The properties of generated molecules when traversing on the corresponding latent variables in sub-space w by PCVAE_tc (left) and PCVAE (right).

5.4.3 Evaluation for property controllable generation

In this section, we validate the performance of the property controllable generation. Specifically, given a predefined value of property y_k, the aim is to explore whether the proposed model could generate a data point x with a property y'_k that is the same as y_k.

Qualitative evaluation. For dSprites and 3Dshapes dataset, since we have no ground-truth method with which to calculate the property y'_k of the generated images, we directly visualize the images that are generated given different values of property y_k. In Figure 5.8, each column contains four images generated given the same value of y_k (here y_k refers to the x_position property) but given the different values for the other two properties. The objects of the generated images in the same column clearly share the same *x_position*.

Similar results can be observed in Figure 5.9, the wall hue or floor hue of four images in the same column are the same given the same desired property. At each column, given

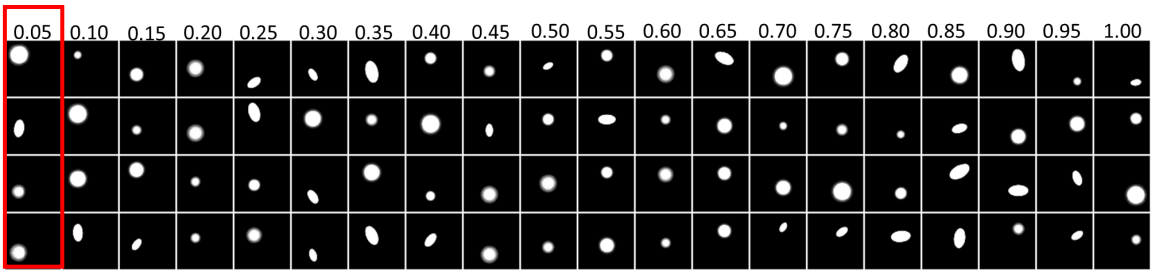
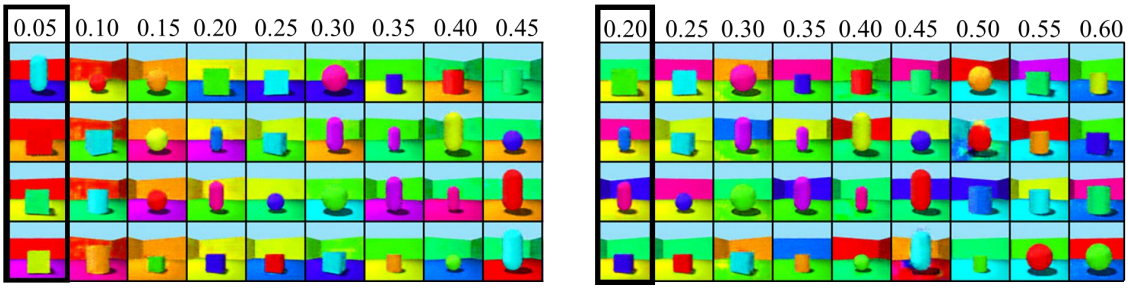


Figure 5.8: The generated images on dSprites dataset when traversing the desired value of property *X-position*. Each column of images are generated given the same value of the desired property.



(a)Control the property of "wall hue" (b)Control the property of "floor hue"

Figure 5.9: The generated images on 3Dshapes dataset when traversing the desired value of property (a) *wall hue* and (b)*floor hue*. Each column of images are generated given the same value of the desired property and random values of other properties.

the same desired values of the relevant properties, the properties of the generated images are the same.

Quantitative evaluation. For the QM9 dataset, since the properties can not be visualized directly from the molecule, we quantitatively measure the property controllable performance in terms of the **MSE** (mean squared error) between the actual property y'_k of generated molecule and the desired property y_k, as shown in Table 5.2. The proposed PCVAE outperforms the other comparison models in successfully controlling *cLogP* and *Molweights* in molecule generation with a smaller **MSE** than that of the comparison methods by around 65.1% and 40.5% on average, respectively. This demonstrate the superiority

of PCVAE in precisely controlling the continuous-valued properties due to the effective invertible property prediction network. In addition, the obvious advantage of PCVAE over PCVAE_tc demonstrate the effectiveness and necessity of the proposed group-wise and property-wise disentanglement term in precisely property controllable generation.

5.4.4 Evaluation for handling correlated properties

In this section, we access the ability of the proposed model to capture and control the correlated properties. The performance is tested on the QM9 molecule set for two tasks: property prediction and property controllable generation. Three properties are selected: *Molweights*, *cLogP* and *cLogS*. *Molweights* is independent from *cLogP* and *cLogS*, while *cLogP* and *cLogS* are inner-correlated.

First, we evaluate whether the subset of latent variables w can power the ability of property prediction, which is a very important task for new compound design in drug discovery. Given an input molecule, the trained encoder (inference model) is used to get the relevant latent variable w_k, and then utilize the invertible function of $p(y_k|w_k)$ to predict the property y_k. Table 5.3 compares the performance of property prediction task on PCVAE and comparison models for uncorrelated properties, as well as the extended model (denoted as PCVAE(cor)) for correlated properties. Though PCVAE(cor) deals with a more difficult case than PCVAE, where an additional property *clogS* that is correlated with *cLogP* is included, PCVAE(cor) still successfully captures the information of the added property cLogS with ignorable influence on the prediction of cLogP and Molweight. Specifically, as shown in Table 5.3, regarding the prediction of independent properties, PCVAE outperforms semi-VAE and CSVAE with a smaller MSE of 33.33 in terms of *Molweights*. It can be also observed that the prediction results of PCVAE_nsp is better than PCVAE, which shows that enforcing both directions' dependence (i.e., $p(w|y)$ as well as $p(y|w)$ via spectral normalization) can introduce more errors than only modelling the dependence $p(y|w)$.

Next, we further explore the performance of the PCVAE(cor) to control the generation of the correlated properties. As shown in Table 5.2, PCVAE(cor) achieves the smallest MSE

Table 5.3: Evaluation of the property prediction task in term of MSE between the predicted and real properties.

Latent Variable	*Molweights*	*cLogP*	*cLogS*
semi-VAE	975.18	1.44	N/A
CSVAE	63.78	1.21	N/A
PCVAE_tc	34.37	0.86	N/A
PCVAE_nsp	**31.50**	**0.64**	N/A
PCVAE	33.33	1.21	N/A
PCVAE (cor)	**33.04**	**0.96**	**0.53**

on all the properties. It demonstrate that adding the supervision of its correlated property *cLogS* does not influence the control of the property *cLogP*. This also demonstrates the effectiveness of the invertible function for handling multi-input and multi-output data. To test the necessities of the proposed PCVAE (cor), we also evaluate the performance of PCVAE and comparison models in dealing with correlated properties, of which the results could be found in Appendix 5.4.5.

5.4.5 Evaluation on the necessaries of PCVAE(cor) for dealing with correlated property

To validate the necessity of PCVAE (cor) for dealing with the correlated properties, we evaluated the performance of PCVAE and comparison models in dealing with correlated properties. As shown in Table 5.4, for generation task, the proposed PCVAE (cor) achieved much smaller MSE than those achieved by CSVAE and PCVAE by averagely around 72.9%, 52.5% and 58.0% on the control of *Molweigt*, *cLogP* and *cLogS*, respectively. This validates that traditional disentangled-based VAE models cannot handle the controllable generation for correlated properties. For prediction task, the proposed PCVAE (cor) also achieved much smaller MSE than those achieved by CSVAE and PCVAE by averagely around 53.12%, 13.2% and 22.1% on the prediction of *Molweight*, *cLogP* and *cLogS* respectively. The bad performance on PCVAE and CSVAE in dealing with correlated properties is caused by the conflicts between the independence enforcement on latent variables w and the dependence

Table 5.4: Comparison of models in dealing with the correlated properties

	Prediction task			Generation task		
Model	*Molweight*	*cLogP*	*cLogS*	*Molweight*	*cLogP*	*cLogS*
CSVAE	88.03	1.21	0.63	183.41	3.84	2.56
PCVAE	52.43	1.03	0.73	168.52	1.76	2.95
PCVAE (cor)	**33.04**	**0.96**	**0.53**	**53.49**	**1.33**	**1.15**

relationship enforcement on w and the correlated properties y, which largely deteriorate the optimization of the whole model.

5.4.6 Evaluation on the quality of generation on QM9

We evaluate the quality of the generated molecules on the QM9 dataset by three metrics: *Novelty* measures the fraction of generated molecules that are not in the training dataset; *Uniqueness* measures the fraction of generated molecules after and before removing duplicates; *Validity* measures the fraction of generated molecules that are chemically valid. The results of the evaluation are shown in Table 5.5. As shown in Table 5.5, the proposed PCVAE still achieve 100% valdity and 99.5%, which is desirable in the problem of controlling generation. We could also found that the proposed PCVAE can have an influence on the uniqueness of the generated data, which may be explained by the supervision of the model. However, considering our focus is on data generation given the desired property, is not a critical issue, whereas the validity and novelty are still very high.

Table 5.5: Quantitative evaluation of the generated molecules.

Method	Validity	Novelty	Uniqueness
GrammarVAE [36]	30.00%	95.44%	9.30%
GraphVAE [160]	61.00%	85.00%	40.90%
CGVAE [161]	**100.00%**	96.33%	**98.03%**
PCVAE_tc	**100.00%**	99.10%	63.50%
PCVAE_nsp	**100.00%**	99.10%	63.50%
PCVAE	**100.00%**	**99.50%**	33.40%

5.5 Conclusion

In this chapter, we have proposed the PCVAE and its extended model, which learns a latent space that separates information correlated with the properties into a predefined subset of latent variables. To accomplish this, we first propose a novel Bayesian variational inference of PCVAE to jointly learn the distribution of data and properties followed by a novel group-wise and property-wise disentanglement term to deal with the complex dependency of subsets of latent variables. Then, we propose to enforce an invertible mutual dependence to allow the precise property controllable generation. At last, we demonstrate through quantitative and qualitative evaluations from three aspects that our proposed model achieves better performance than existing and baseline models. In future work, we plan to extend PCVAE to a semi-supervised setting, where some of the property labels are missing.

Chapter 6: Conclusions and Future Works

This dissertation aims to handle the conditional and intepretable graph generation problems in the domain of deep learning: conditional deep graph generation (i.e., deep graph transformation) and deep graph interpretation. For the conditional deep graph generation problem, we explore two sub-tasks, namely the conditional generation for graph topology and the multi-attributed graph. To further interpret the graph generation process, we explore two sub-tasks, namely, the disentangled graph generation and the property controllable graph generation, which provide the interpretation for graph generation from different aspects.

To deal with the conditional graph topology generation problem, we propose a novel GT-GAN model that is able to transform an input graph to a target graph. We first propose a conditional graph GAN architecture that consists of an encoder-decoder style translator and a conditional graph discriminator to learn the conditional distribution for graph topology transformation. In addition, a novel graph U-net translator with graph skips is proposed to guarantee the model to learn multiple levels' information in the graph encoder as well as select valuable ones to generate the target graphs in graph decoder. To corporate with the U-net structure, a set of mirrored graph convolution and deconvolution layers are designed including both the edge and the node convolution and deconvolution. Furthermore, the proposed graph convolution and deconvolution layers are able to jointly embed the local and global information. Experimental results show that our GT-GAN can discover the ground-truth transformation rules, and significantly outperforms other baselines in terms of both effectiveness and scalability.

The above problem is then extended to a more general setting, namely, the conditional multi-attributed graph generation problem. To solve it, We propose a more generic node-edge co-evolution framework for the multi-attributed graph transformation, where a

novel graph spectral-based regularization is proposed to enforce the consistent of nodes and edges in the generated graphs. To jointly tackle the different types of interactions among nodes and edges, the node and edge translation paths are proposed in each block and the graph spectral-based regularization is proposed to preserve the consistent spectral property of graphs. Extensive experiments have been conducted on the synthetic and real-world datasets. As the extension of the undirected NEC-DGT, we also propose the sign-NEC-DGT for the signed graphs and di-NEC-DGT for the directed graphs. Experimental results show that our NEC-DGT can discover the ground-truth translation rules and significantly outperform comparison methods in terms effectiveness.

To interpret the conditional deep generation techniques proposed in the first two sub-tasks, we then resort to the disentangled representation learning for the structured data. We introduced NED-VAE, a novel and the first method for disentanglement representation learning 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 perform an experimental evaluation of disentangling performance qualitatively and quantitatively of the proposed NED-VAE, its extensions, and the comparison models. 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.

Beyond exploring the relationship between the latent variables and generated graphs as mentioned above, it is more important to manipulate the properties of the generated graph. To solve this problem, we have proposed the PCVAE and its extended model, which learns a latent space that separates information correlated with the properties into a pre-defined subset of latent variables. To accomplish this, we first propose a novel Bayesian variational inference of PCVAE to jointly learn the distribution of data and properties followed by a novel group-wise and property-wise disentanglement term to deal with the

complex dependency of subsets of latent variables. Then, we propose to enforce an invertible mutual dependence to allow the precise property controllable generation. At last, we demonstrate through quantitative and qualitative evaluations from three aspects that our proposed model achieves better performance than existing and baseline models.

6.1 Research Tasks

The major research tasks are described as follows. The current status of these tasks is listed in Table 6.1.

6.1.1 Conditional Deep Graph Topology Generation

- **The definition and proposal of the novel problem (A1)**. We first define and mathematically formulate the deep graph translation problem. We also propose a novel indirect evaluation metric for evaluating the generated graph based on a supervised graph classification model.
- **A generic framework for deep graph topology transformation (A2)**. We develop a generic framework called GT-GAN consisting of a novel graph translator and conditional graph discriminator for learning a conditional distribution of target graphs given the input graphs.
- **A novel graph convolution-based encoder (A3)**. We propose a novel graph encoder consisting of "edge convolution" layers that extract various relations among nodes containing both local and global information, and "node convolution" layers that embed the node representations based on the extracted relations.
- **A novel graph deconvolution-based decoder (A4)**. We propose a novel graph decoder consisting of the "edge deconvolution" and "node deconvolution" layers, which can decode the node representations first into the latent relations of the target graph and then generate the final target graph. The graph skip-connection is also utilized to map the learned latent relations between the input and target graphs.

- **Validating performance on synthetic and real-world dataset (A5)**. We conduct extensive experiments on both synthetic and real-world datasets on six performance metrics to demonstrate the effectiveness and efficiency of the proposed model.

6.1.2 Conditional Deep Multi-attributed Graph Generation with Node-Edge Co-evolution

- **The development of a new framework for multi-attributed graph translation (B1)**. We formulate, for the first time, a multi-attributed graph transformation problem and propose the NEC-DGT to tackle this problem. The proposed framework is generic for different applications where both node and edge attributes can change through transformation.

- **The proposal of novel and generic edge translation layers and blocks (B2)**. A new edge translation path is proposed to translate the edge attributes from the input domain to the output domain. Existing edge translation methods were proven to be special cases of ours, which can handle broad multi-attribute edges and nodes.

- **The proposal of a spectral-based regularization that ensures consistency of the predicted nodes and edges (B3)**. In order to discover and maintain the inherent relationships between predicted nodes and edges, a new non-parametric graph Laplacian regularization with a graph frequency regularization is proposed and leveraged.

- **The conduct of extensive experiments to validate the effectiveness and efficiency of the proposed model (B4)**. Extensive experiments on four synthetic and four real-world datasets demonstrated that NEC-DGT is capable of generating graphs close to ground-truth target graphs and significantly outperforms other generative models.

- **The extension of the proposed model to signed graph translation (B5)**. A variant of the proposed model called sign-NEC-DGT is designed by proposing a novel

signed spectral-based regularization, which is based on a specially designed definition of graph Laplacian for signed networks.

- **The extension of the proposed model to directed graph translation (B6)**. A variant of the proposed model called di-NEC-DGT is designed by proposing a novel direct spectral-based regularization, which is based on the approximation of the Perron vector.

6.1.3 Interpretable Deep Graph Generation with Node-edge Co-disentanglement

- **Derivation of a novel objective framework for the disentanglement of attributed graph generation (C1)**. In order to jointly disentangle the nodes and edges, we derive a novel objective framework for learning three factors that are exclusive to node patterns, exclusive to edge patterns, and those spanning node-edge-joint patterns.

- **The proposal of a novel architecture for disentanglement learning on graphs. (C2)** Derived from the theoretical objective of our framework, a novel architecture is proposed for the representation learning of graphs to learn different types of representations and to co-generate both nodes and edges.

- **Constrains on simultaneous group-wise and variable-wise disentanglement (C3)**. We conduct the hierarchically disentanglement of the attributed graph generation according to node, edge, and their joint factors.

- **Comprehensive experiments to validate the effectiveness of the proposed model (C4)**. Qualitative and quantitative experiments on both synthetic and two real-world datasets are used to validate whether the proposed model is indeed capable of learning disentangled factors for different types of graphs.

Table 6.1: Research tasks and status

Task	Description	Status
Research Area A	Deep Generative model for Graph Topology Transformation	
A1	The definition and proposal of the novel problem	Completed
A2	Proposal of the GT-GAN based framework	Completed
A3	Proposal of the graph convolution-based encoder	Completed
A4	Proposal of the graph deconvolution-based decoder	Completed
A5	Validation on synthetic and real-world datasets	Completed
Research Area B	Multi-attributed Graph Transformation with Node-edge Co-disentanglement	
B1	Proposal the node-edge co-evolution framework	Completed
B2	Proposal of generic edge convolution layers	Completed
B3	Proposal of the graph spectral regularization	Completed
B4	Validation on synthetic and real-world datasets	Completed
B5	Extension to signed graph translation	Completed
B6	Extension to directed graph translation	Completed
Research Area C	Interpretable Deep Graph Generation with Node-edge Co-disentanglement	
C1	Derivation of graph disentanglement objective.	Completed
C2	Proposal of node-edge co-disentanglement framework	Completed
C3	Constrains on group/variable-wise disentanglement	Completed
C4	Validation on synthetic and real-world datasets	Completed
Research Area D	Property Controllable Generative Model for Structured Data Generation	
D1	Proposal of graphical model and inference process	Completed
D2	Proposal of disentangled enforcement for structured latent	Completed
D3	Proposal of property controllable component	Completed
D4	Validation on image and molecule datasets	Completed
E	Dissertation Writing and revision	Completed

6.1.4 Property Controllable Graph Generation via Graph Manipulation

- **The inference of the graphical model and objective (D1)**. A new Bayesian model that inductively biases the latent representation using explicit real data properties is proposed. A variational inference strategy and inference model have been customized to ensure effective Bayesian inference.

- **The disentanglement enforcement of structured latent representation (D2)**. Group-wise and property-wise disentanglement terms are proposed to enhance the mutual independence among property, relevant and irrelevant latent variables .

- **The proposal of a property controllable component (D3)**. The invertible

mutual dependence between property-latent variable pair is achieved by enforcing an invertibility constraint over a residual-based decoder.

- **Comprehensive experiments to validate the effectiveness of the model (D4)**. The quantitative and qualitative evaluation performed on both the image and molecule datasets is used to reveal that our PCVAE is effective in capturing and manipulating the desired properties.

6.2 Publications

6.2.1 Published papers at GMU

1. Xiaojie Guo, Liang Zhao, Houman Homayoun, Sai Manoj Pudukotai Dinakarrao. Deep Graph Transformation for Attributed, Directed, and Signed Networks". Knowledge and Information Systems (KAIS), (impact factor: 2.936), to appear. [**Best of ICDMs**]

2. Xiaojie Guo , Yuanqi Du, Liang Zhao. Property Controllable Variational Autoencoder via Invertible Mutual Dependence. The 9th International Conference on Learning Representations (ICLR 2021), (Acceptance Rate: 28.7%). May 4-7, 2021, Virtual Event, Vienna, USA. to appear.

3. 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 2020), (Ac- ceptance Rate: 16.8%), August 23-27, 2020, Virtual Event, CA, USA. ACM, New York, NY, USA.

4. Xiaojie Guo, Liang Zhao, Cameron Nowzari, Setareh Rafatirad, Houman Homayoun, and Sai Dinakarrao. Deep Multi-attributed Graph Translation with Node-Edge Co-evolution.The 19th International Conference on Data Mining (ICDM 2019), long

paper, (acceptance rate: 9.08%), pp. to appear, Beijing, China, Nov 2019. **[Best Paper Award]**

5. Xiaojie Guo, Amir Alipour-Fanid, Lingfei Wu, Hemant Purohit, Xiang Chen, Kai Zeng and Liang Zhao. Deep Classifier Cascades for Open World Recognition. The 28th ACM International Conference on Information and Knowledge Management (CIKM 2019), long paper, (acceptance rate: 19.4%), pp. 179-188, Beijing, China, Nov 2019.

6. Yuanqi Du, Xiaojie Guo, Liang Zhao, and Amarda Shehu. Interpretable Molecule Generation via Disentanglement Learning. The ACM Conference of Bioinformatics and Computational Biology: Computational Structural Biology Workshop (BCB CSBW 2020).

7. P D Sai Manoj, Xiaojie Guo, Hossein Sayadi, Cameron Nowzari, Avesta Sasan, Setareh Rafatirad, Liang Zhao, Houman Homayoun. Cognitive and Scalable Technique for Securing IoT Networks Against Malware Epidemics, in IEEE Access, vol.8, pp. 138508 138528, 2020.

8. Liang Zhao, Junxiang Wang, and Xiaojie Guo. Distant-supervision of heterogeneous multitask learning for social event forecasting with multilingual indicators. Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018), Oral presentation (acceptance rate: 11.0%), pp. 4498-4505, New Orleans, US, Feb 2018.

9. Yuyang Gao., Xiaojie Guo, Liang Zhao. Local event forecasting and synthesis using unpaired deep graph translations. In Proceedings of the 2nd ACM SIGSPATIAL Workshop on Analytics for Local Events and News (LENS 2018), pp. 1-8, seattle, US, Nov 2018.

6.2.2 Previously published papers before GMU

1. Xiaojie Guo, Liang Chen, Changqing Shen. Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis. Measurement, 2016,

v93, 490-502.

2. Xiaojie Guo, Changqing Shen, Liang Chen. Deep fault recognizer: an integrated model to denoise and extract features for fault diagnosis in rotating machinery. Applied Science, 2016, 7(1), 41.

3. Xiaojie, Guo, Liang Chen, et al. An improved K-means algorithm and its application in the evaluation of air quality levels. Proceedings of the 27th Chinese Control and Decision Conference (CCDC 2015), pp. 3324-3329, IEEE, Qingdao, China, May 2015.

4. Liang Chen, Xiaojie Guo, et al. Human face recognition based on adaptive deep Convolution Neural Network. Proceedings of the 35th Chinese Control Conference (CCC 2016), pp. 6967-6970, IEEE, Sichuan, China, July 2016.

6.2.3 Submitted papers

1. Xiaojie Guo, Lingfei Wu, and Liang Zhao. Deep graph Translation. IEEE Transactions on Neural Networks and Learning Systems (TNNLS),submitted.

2. Xiaojie Guo, Yuanqi Du, and Liang Zhao. Disentangled Deep Generative Models for Spatial Networks. ACM SIGKDDConference on Knowledge Discovery and Data Mining (KDD 2021),submitted.

3. Xiaojie Guo, Liang Zhao. A Systematic Survey on Deep Generative Models for Graph Generation. ACM Computing Surveys (CSUR), submitted.

4. Xiaojie Guo, Sivani Tadepalli, Liang Zhao, and Amarda Shehu.Generating Tertiary Protein Structures via an Interpretative Variational Autoencoder. Scientific reports, submitted.

6.3 Future Research Directions

6.3.1 Disentangled Representation Learning for Spatial-temporal Graphs generation

- Graph neural networks for spatial-temporal graphs. Designing a graph neural network for spatial-temporal graphs is the foundation for the spatial-temporal graph generation. The proposed GNN should be able to capture both the temporal and spatial information of a series of graphs.

- Deep generative models for spatial-temporal graph generation. The goal is to disentangle the spatial-related and temporal-related semantic factors of the spatial-temporal graphs. For the temporal-related information, the time-invariant and time variant semantic factors should be disentangled. For the spatial-related information, the graph-related and spatial-related semantic factors should be disentangled.

6.3.2 Deep Graph Transformation for NLP tasks

- Graph-to-graph transformation for information extraction. We plan to formalize the information extraction as a graph transformation problem, where the goal is to learn a mapping from the input dependency parser graph to the information graph. During the transformation, three tasks (i.e., entity recognition, relation extraction and coreference linking) are jointly conducted.

- Graph-to-graph transformation for AMR semantic parsing. The goal is to propose an end-to-end graph transformation framework for semantic parsing which bridges the sentence dependency parsing graph to AMR logits graph. The model should be able to address the graph sparsity issue in neural AMR parsing and conduct the unpaired transformation in dealing with the situation where less AMR annotations are available.

6.3.3 Disentanglement Graph Generative Models for Neural-Symbolic Computing

- Scene graph interpretation and manipulation. The novel problem of image manipulation from scene graphs requires to edit images by merely applying changes in the nodes or edges of a semantic graph of the image. Our goal is to encode image information in a given constellation and from there on generate new constellations, such as replacing objects or even changing relationships between objects.

- Interpretable Spatial-Temporal Scene Graph Representation Learning. For the interpretable generation of video scene graphs, the goal is to design the spatial-temporal graph encoder and decoder, which is invariant to Rotation and shift, disentangling spatial, temporal factors on latent representation of each object.

Appendix A: Deep Multi-attributed Graph Transformation for Node Edge Co-evolution

A.1 Proof of Theorem 1

Proof. Due to $E_{s,(i,j)} \in [1,0]$, The influence calculated from the effects function in node translation path of C-DGT for block s of node i can be summarized as:

$$\bar{I}_{s,i}^{F} = \sum_{j=1}^{d} \phi_{I_F}(E_{s-1,(i,j)}[F_{s-1,i}; 1; F_{s-1,j}]) \tag{A.1}$$

Then after the updating function, the node i at stage s can be modeled as: $F_{s,i} = \phi U_F(F_{0,i}; C; \bar{I}_{s,i}^{F})$. $F_{0,i}$ denotes the node attributes from the input graphs.

In DCNN, the message information learned for node i is formalized as:

$$\hat{I}_{s,i}^{F} = \sum_{n=0}^{s-1} \sum_{j=1}^{d} p_{n,(i,j)} F_{0,j} W_n^{s-1} \tag{A.2}$$

where s denotes the pre-defined number of hop information utilized in DCNN. $W_n \in \mathbb{R}^{d \times F}$ denotes the weights related to nth hop. and F denotes to the dimension of node effects. The $p_{s,(i,j)}$ is the s-hop reachable of two nodes calculated by an iteration: $p_{s+1,(i,j)} = \sum_{k=1}^{d} p_{0,(i,k)} p_{s,(k,j)}$ and $p_{0,(i,j)} = E_{0,(i,j)}$. For DCNN, $F_{s,i} = F_{0,i}$. Due to the iterative nature of n-hop problem, we use Iterative proof to validate that $\bar{I}_{s,i}^{F} = \hat{I}_{s,i}^{F}$ when the parameters of C-DGT are trained to meet some requirements.

(1) If $s = 1$, $p_{0,(i,j)} = E_{0,(i,j)}$ and $E_{0,(i,j)} \in [1,0]$.

For DCNN we got:

$$\hat{I}_{1,i}^{F} = \sum_{j=1}^{d} E_{0,(i,j)} F_{0,j} W_0^0 \tag{A.3}$$

where $W_0^0 \in \mathbb{R}^{d\times F}$.

For C-DGT, since ϕ_{E_O} is implemented by a two-layer MLP. After the first layer, the output can be written as:

$$\bar{I'}_{1,i}^{F} = \sum_{j=1}^{d} E_{0,(i,j)}[F_{0,i}; 1; F_{0,j}]W_1 \tag{A.4}$$

$$= \sum_{j=1}^{d} E_{0,(i,j)}(F_{0,i}w_{11} + w_{12} + F_{0,j}w_{13}) \tag{A.5}$$

where $W_1 \in \mathbb{R}^{(2d+1)\times d}$ and can be divided into $w_{11} \in \mathbb{R}^{d\times d}$, $w_{12} \in \mathbb{R}^{1\times d}$ and $w_{13} \in \mathbb{R}^{d\times d}$. When $w_{11} = \mathbf{0}$ and $w_{12} = \overrightarrow{0}$, we can got:

$$\bar{I}_{s,i}^{F} = \sum_{j=1}^{d} E_{0,(i,j)}F_{0,j}w_{13} \tag{A.6}$$

After the second layer of MLP, it can be expressed as:

$$\bar{I}_{s,i}^{F} = \sum_{j=1}^{d} E_{0,(i,j)}F_{0,j}w_{13}W_2 \tag{A.7}$$

where $W_2 \in \mathbb{R}^{d\times F}$. Thus, if $w_{13}W_2 = W_0^0$, it is proved that $\bar{I}_{1,i}^{F} = \hat{I}_{1,i}^{F}$.

(2) Given $\bar{I}_{m,i}^{F} = \hat{I}_{m,i}^{F}$ is valid (s=m), we need prove $\bar{I}_{m+1,i}^{F} = \hat{I}_{m+1,i}^{F}$, namely to prove:

$$\sum_{j=1}^{d} \phi_{I_F}(E_{m,(i,j)}[F_{m,i}; 1; F_{m,j}]) = \sum_{n=0}^{m}\sum_{j=1}^{d} p_{n,(i,j)})F_{0,j}W_n^m \tag{A.8}$$

For the right of part of equation A.8, we can write it as:

$$right = \sum_{n=1}^{m}\sum_{j=1}^{d} p_{n,(i,j)} F_{0,j} W_n^m + \sum_{j=1}^{d} p_{0,(i,j)} F_{0,j} W_0^m \tag{A.9}$$

$$= \sum_{n=1}^{m}\sum_{j=1}^{d}\sum_{k=1}^{d} p_{0,(i,k)} p_{n-1,(k,j)} F_{0,j} W_n^m + \sum_{j=1}^{d} p_{0,(i,j)} F_{0,j} W_0^m \tag{A.10}$$

$$= \sum_{n=1}^{m}\sum_{j=1}^{d}\sum_{k=1}^{d} E_{0,(i,k)} p_{n-1,(k,j)} F_{0,j} W_n^m + \sum_{j=1}^{d} E_{0,(i,j)} F_{0,j} W_0^m \tag{A.11}$$

For the left part of Equation A.8, we can write as:

$$left = \sum_{j=1}^{d} E_{m,(i,j)} [F_{m,i}; 1; F_{m,j}] W_6 \tag{A.12}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} (F_{m,i} w_{61} + E_{m,(i,j)} w_{62} + F_{m,j} w_{63}) \tag{A.13}$$

where $W_6 \in \mathbb{R}^{(2d+1)\times d}$ and can be divided as $w_{61} \in \mathbb{R}^{d\times d}$, $w_{62} \in \mathbb{R}^{1\times d}$ and $w_{63} \in \mathbb{R}^{d\times d}$.

When $w_6 1 = \mathbf{0}$ and $w_{62} = \overrightarrow{0}$, we can got:

$$left = \sum_{j=1}^{d} E_{m,(i,j)} F_{m,j} w_{63} \tag{A.14}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} \phi_{U_F}(F_{0,j}; \bar{I}^F_{m,j}) w_{63} \tag{A.15}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} (F_{0,j} w_{70} + \bar{I}^F_{m,j} w_{71}) w_{63} \tag{A.16}$$

$$\sum_{j=1}^{d} \phi_{I_F}(E_{m,(i,j)}[F_{m,i}; 1; F_{m,j}]) = \sum_{j=1}^{d} E_{m,(i,j)} (F_{0,j} w_{70} w_{63} + \bar{I}^F_{m,j} w_{71} w_{63}) \tag{A.17}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} (F_{0,j} w_{70} w_{63} + \hat{I}^F_{m,j} w_{71} w_{63}) \tag{A.18}$$

$$= \sum_{j=1}^{d} E_{m,(i,j)} F_{0,j} w_{70} w_{63} + \sum_{n=0}^{m-1} \sum_{j=1}^{d} \sum_{k=1}^{d} E_{m,(i,j)} p_{n,(j,k)} F_{0,k} w_{71} w_{63} W_n^{m-1} \tag{A.19}$$

Since $E_{m,(i,j)} = E_{0,(i,j)}$, when $w_{70} w_{63} = W_0^m$ and $W_n^{m-1} w_{71} w_{63} = W_{n+1}^m$, it is proved that $\bar{I}_{m+1,i} = \hat{I}_{m+1,i}$.

□

Appendix B: Interpretable Deep Graph Generation with Node Edge Co-disentanglement

B.1 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 B.1. The architecture details of comparison methods graphVAE and baseline GDVAE are shown in Table B.3 and Table B.2 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.2 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 [26] 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

Table B.1: Encoders and decoders architectures (Each layers is expressed in the format as $< filter\_size >< layertype >< Num\_channel >< Activationfunction >< stridesize >$. *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 B.2: Encoders and decoders architectures of Baseline GDVAE (Each layers is expressed in the format as $< filter\_size >< layertype >< Num\_channel >< Activationfunction >< stridesize >$. *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

edge representations distribution.

The **graph encoder** consists of several graph convolution layers [97] 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.

B.3 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

Table B.3: Encoders and decoders architectures of graphVAE (Each layers is expressed in the format as $< filter\_size >< layertype >< Num\_channel >< Activationfunction >< stridesize >$. 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

role in causing the cellular prion protein to change to a misfold form and lead to prion diseases [162]. 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. 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 $8A (D_{ij} >= 8A)$.

Appendix C: Property Controllable Variational Auto-encoders via Invertible Mutual Dependence

C.1 Additional derivations about Methodology

C.1.1 Detailed Derivation of Bayesian variational inference of PCVAE

Given an approximate posterior $q(z, w|x, y)$, we can use the Jensen's equality to obtain the variational lower bound of $p(x, y)$ as:

$$\log p(x, y) \geq \mathbb{E}_{q(z,w|x,y)}[\log p(x, y, w, z)/q(z, w|x, y)]. \tag{C.1}$$

We have two assumptions: (1) w only encode the information from y, namely, x and y are conditionally independent given w (i.e., $x \perp y|w$); (2) z is independent from w and y, namely $z \perp w$ and $z \perp y$, which is equal to $y \perp z|w$ (see derivation in Appendix C.1.2). First, based on the two assumptions, we can get $x \perp y|(z, w)$ (see derivation in Appendix C.1.3). Thus, we have $\log p(x, y|z, w) = \log p(x|z, w) + \log p(y|z, w)$. Then, based on the assumption $y \perp z|w$, we can have $\log p(y|z, w) = \log p(y|w)$. Then we get $\log p(x, y|z, w) = \log p(x|z, w) + \log p(y|w)$. To explicitly represent the dependence between x and (z, w) as well as the dependence between y and w, we can parameterize the joint log-likelihood as $\log p_{\theta,\gamma}(x, y, w, z)$ with θ and γ as:

$$\log p_{\theta,\gamma}(x, y, w, z) = \log p_\theta(x|z, w) + \log p(z, w) + \log p_\gamma(y|w). \tag{C.2}$$

Given the condition that a parameterized $q_\phi(z, w|x, y) = q_\phi(z, w|x) = q_\phi(z|x)q_\phi(w|x)$ (since the information on y is included in x), by taking it into the above variational lower

bound term, we obtain the negative part as an upper bound on $-\log p_{\theta,\gamma}(x,y)$ as:

$$\begin{aligned}
\mathcal{L}_1 =& \mathbb{E}_{q(z,w|x,y)}[\log p_\theta(x|z,w) - \log p(z,w) - \log p_\gamma(y|w) + \log q(z,w|x,y)] \\
&= -\mathbb{E}_{q(z,w|x,y)}[-\log p_\theta(x|z,w)] - \mathbb{E}_{q(z,w|x,y)}[\log p_\gamma(y|w)] \\
&+ \mathbb{E}_{q(z,w|x,y)}[\log q(z,w|x,y) - \log p(z,w)] \qquad \text{(C.3)} \\
&= -\mathbb{E}_{q(z,w|x)}[-\log p_\theta(x|z,w)] - \mathbb{E}_{q(w|x)}[\log p_\gamma(y|w)] + \mathbb{E}_{q(z,w|x)}[\log \frac{q(z,w|x)}{p(z,w)}] \\
&= -\mathbb{E}_{q_\phi(z,w|x)}[\log p_\theta(x|z,w)] - \mathbb{E}_{q_\phi(w|x)}[\log p_\gamma(y|w)] + D_{KL}(q_\phi(z,w|x)||p(z,w))
\end{aligned}$$

Based on the above derivation of $D_{KL}(q_\phi(z,w|x)||p(z,w))$, we could further decompose it as:

$$\begin{aligned}
&D_{KL}(q_\phi(z,w|x)||p(z,w)) = \mathbb{E}_{q(z,w|x)}[\log \frac{q(z,w|x)}{p(z,w)}] \\
&= \mathbb{E}_{q(z,w|x)}[\log q(z,w|x) - \log p(z,w)] \\
&= \mathbb{E}_{q(z,w|x)}[\log \frac{q(z,w|x)}{q(z,w)} + \log \frac{q(z,w)}{\prod_{i,j} q(z_i)q(w_j))} + \log \frac{\prod_i q(z_i)}{\prod_i p(z_i)} + \log \frac{\prod_i q(w_i)}{\prod_i p(w_i)}].
\end{aligned}$$

Considering that $q(z,w) = \mathbb{E}_{p(x)} q(z,w|x)$, we can get:

$$\mathbb{E}_{p(x)}[D_{KL}(q_\phi(z,w|x)||p(z,w))]$$

$$= \mathbb{E}_{p(x)}[\mathbb{E}_{q(z,w|x)}[\log \frac{q(z,w|x)}{q(z,w)} + \log \frac{q(z,w)}{\prod_{i,j} q(z_i)q(w_j))} + \log \frac{\prod_i q(z_i)}{\prod_i p(z_i)} + \log \frac{\prod_i q(w_i)}{\prod_i p(w_i)}]]$$

$$= \mathbb{E}_{q(z,w,x)}[\log \frac{q(z,w,x)}{q(z,w)p(x)} + \mathbb{E}_{q(z,w)}[\log \frac{q(z,w)}{\prod_{i,j} q(z_i)q(w_j))}]$$

$$+ \mathbb{E}_{q(z)}[\log \frac{\prod_i q(z_i)}{\prod_i p(z_i)}] + \mathbb{E}_{q(w)}[\log \frac{\prod_i q(w_i)}{\prod_i p(w_i)}] \tag{C.4}$$

$$= D_{KL}(q_\phi(z,w,x) \parallel q(z,w)p(x)) + D_{KL}(q(z,w) \parallel \prod\nolimits_{i,j} q(z_i)q(w_j))$$

$$+ \sum\nolimits_i D_{KL}(q(z_i) \parallel p(z_i)) + \sum\nolimits_j D_{KL}(q(w_j) \parallel p(w_j)) \tag{C.5}$$

C.1.2 Derivation process explanation 1

In this section, we proof that if $z \perp w$ and $z \perp y$, we can have $y \perp z|w$.

First, based on the Bayesian theory, we could have $p(y,z|w) = p(z|y,w)p(y|w) = p(y|z,w)p(z|w)$, namely,

$$p(z|y,w)p(y|w) = p(y|z,w)p(z|w). \tag{C.6}$$

Then, considering that $z \perp w$ and $z \perp y$, we can have $p(z|w) = p(z)$ and also $p(z|y,w) = p(z)$. Then the right and left sides of Equation C.6 can be replaced as: $p(z)p(y|w) = p(y|z,w)p(z)$, and then we have $p(y|w) = p(y|z,w)$. Given $p(y|w) = p(y|z,w)$, both sides of the equations are multiplied by $p(z|w)$, and we have $p(z|w)p(y|w) = p(y|z,w)p(z|w) = p(y,z|w)$. Thus, $y \perp z|w$.

C.1.3 Derivation process explanation 2

In this section, we proof that if $x \perp y|w$, $y \perp z$, and $z \perp w$, we can have $x \perp y|(w,z)$.

First, based on the Bayesian theory, we could have $p(x,y|w,z) = p(y|x,z,w)p(x|z,w) =$

$p(x|y,z,w)p(y|z,w)$, namely,

$$p(y|x,z,w)p(x|z,w) = p(x|y,z,w)p(y|z,w) \tag{C.7}$$

Then, considering that $y \perp z$ and $z \perp w$ (namely $y \perp z|w$, as proved in Section C.1.2), as well as $y \perp x|w$, we can have $p(y|x,z,w) = p(y|w)$ and also $p(y|z,w) = p(y|w)$. Then the right and left sides of Equation C.7 can be replaced as $p(y|w)p(x|z,w) = p(x|z,y,w)p(y|w)$, and then we have $p(x|z,w) = p(x|y,z,w)$. Thus, we get $x \perp y|(w,z)$.

C.2 Architecture and Hyper-parameters

C.2.1 Architecture and Hyper-parameters for dSprits and 3Dshapes dataset

Based on the description of the implementation of the proposed objective, there are three components, namely, *encoder_1* to model $p_\theta(z,w|x)$, *decoder_1* to model $p_\phi(x|z,w)$ and *decoder_2* to model $p_\gamma(w|y)$. When evaluate the dSprites data, the number of latent dimensions in z is 3 and the number of latent dimensions in w is also 3. The detailed architectures are shown in Table C.1. The hyper-paramter used for training is detailed in Table C.3.

Table C.1: Encoders and decoders architectures of PCVAE for dSprites dataset (Each layers is expressed in the format as $<$ *kernel_size* $><$ *layer_type* $><$ *Num_channel* $><$ *Activation_function* $><$ *stride_size* $>$. *FC* refers to the fully connected layers).

Encoder_1	Decoder_1	Decoder_2
Input: $X \in \mathbb{R}^{64\times64}$	Input$[z,w] \in \mathbb{R}^6$	Input:$w_k \in \mathbb{R}$
4×4 Conv.32 ReLU.stride 2	FC.256 ReLU	FC.50 ReLU
4×4 Conv.32 ReLU.stride 2	FC.256 ReLU	Spectral_Norm Layer
4×4 Conv.32 ReLU.stride 2	FC.512 ReLU	FC.1
4×4 Conv.32 ReLU.stride 2	4×4 ConvTranspose.32 ReLU.stride 2	N/A
FC.256 ReLU	4×4 ConvTranspose.32 ReLU.stride 2	N/A
FC.256 ReLU	4×4 ConvTranspose.32 ReLU.stride 2	N/A
FC.12	4×4 ConvTranspose.64 ReLU.stride 2	N/A

Table C.2: Encoders and decoders architectures of PCVAE for QM9 dataset (Each layers is expressed in the format as $< kernel\_size >< layer\_type >< Num\_channel >< Activation\_function >< stride\_size >$. *FC* refers to the fully connected layers).

Encoder_1	Decoder_1	Decoder_2
Input: $G(X, A), X \in \mathbb{R}^9$	Input$[z, w] \in \mathbb{R}^{100}, h_type \in \mathbb{R}^{101}$	Input:$w_k \in \mathbb{R}$
FC.100 ReLU	FC.100 ReLU	FC.20 ReLU
GGNN.100 ReLU	GGNN.100 ReLU	Spectral_Norm Layer
GGNN.100 ReLU	GGNN.100 ReLU	FC.1
FC.100	FC.9 (node) FC.3 (edge)	N/A

C.2.2 Architecture and Hyper-parameters for molecule QM9 dataset

The architecture used for evaluation on the QM9 dataset are totally borrowed from the work proposed by Liu et al. [163]. A molecule is represented as a graph $G(X, A)$, where each atom is a node and X refers to the features for all nodes; each bond is an edge, where A denotes to the adjacent matrix of the graph. We briefly introduce the model and provide it architecture parameters in Table C.2. We recommend the reader to the work [163] for more details.

Molecule Encoder and Decoder. A *encoder_1* is constructed to model $p_\phi(z, w|x)$ based on a gated graph neural network (GGNN). As a result, by sampling from the modelled distribution, (z, w) are obtained variables containing the graph representation vectors. The molecule *decoder_1* models the distribution $p_\theta(x|z, w)$ to generate the molecule graph G. The molecule *decoder_2* models the distribution $p_\theta(y|w)$ to predict the properties y. The process proceeds in an auto-regressive style. In each step a focus node is chosen to be visited, and then the edges are generated related to this focus node. The nodes are ordered by using the breadth-first traversal. The molecule decoder mainly contains three steps, namely *node initialization*, *node update* and *edge selection and labelling*.

Node Initialization. We first define N as an upper bound on the number of nodes in the final generated graph. An initial state $h_i^{(t=0)}$ is assigned with each node v_i in a set of initially unconnected nodes. Specifically, $h_i^{(t=0)}$ is the concatenation as $[(z, w), \tau_i]$, where τ_i

is an one-hot vector indicating the atom type. τ_i is derived from (z, w) by sampling from the softmax output of a learned mapping $\tau_i \sim f(Z_i)$. From these node-level states, we can calculate global representations $H(t)$, which is the average representation of nodes in the connected component at generation step t. In addition to N working nodes, a special "stop node" is also initialized to a learned representation h_{end} for managing algorithm termination detailed as below.

Edge Selection and Labeling At each step t, a focus node v_i is picked from the queue of nodes. Then an edges $e_{i,j}$ is selected from node v_i to node v_j with label $E_{i,j}$. Specifically, for each non-focus node v_j, we construct a feature vector $\eta_{i,j}^{(t)} = [h_i^{(t)}, h_j^{(t)}, d_{i,j}, H(t), H(0)]$, where $d_{i,j}$ is the graph distance (i.e. path) between two nodes v_i, v_j. We use these representations to produce a distribution over candidate edges as $p(e_{i,j}, E_{i,j}|\eta_{i,j}^{(t)}) = p(E_{i,j}|\eta_{i,j}^{(t)}, e_{i,j}) \cdot p(e_{i,j}|\eta_{i,j}^{(t)})$.

The parameters of the distribution are calculated as softmax outputs from neural networks. New edges are sampled one by one from the above learned distributions. Any nodes that are connected to the graph for the first time during this edge selection are added to the node queue.

Node Update. Whenever we obtain a new graph $G^{(t+1)}$ at step t, the previous node states $h_i^{(t)}$ is discarded and a new node representations $h_i^{(t+1)}$ for each node is calculated by taking their (possibly changed) neighborhood into account. To this end, a standard gated graph neural network (GGNN) is utilized through S steps, which is defined as a recurrent operation over messages $r_i^{(s)}$.

Termination. In the edge generation process of each node, the edges to a node v_i is kept added until an edge to the stop node is selected. Then we move the focus from the node v_i, and regard v_i as a "closed" node. The next focus node is then selected from the focus queue. In this way, a single connected component is grown in a breadth-first manner. The node and edge generations continue until the queue of nodes is empty.

Table C.3: hyper-paramter used for training on dSprites and QM9 datasets

Dataset	Learning_rate	Batch_size	α	ρ	Num_iteration	c (spectral_norm)
dSprites	5e-4	64	3	1	20	0.97
QM9	5e-4	32	1	6	100	1

C.2.3 Estimation of Group-wise and property-wise disentanglement terms

To evaluate the density $q(z)$, $q(w)$ and $q(w_i)$ in the second loss $\mathcal{L}_2$, a naïve Monte Carlo approximation [164] is utilized for the estimation. We describe the operation by taking $q(z)$ as an example. A naïve Monte Carlo approximation based on a minibatch of samples from $p(n)$ (n is the data sample index) is likely to underestimate $q(z)$. As stated by [121], this can be intuitively seen by viewing $q(z)$ as a mixture distribution where the data index n indicates the mixture component. With a randomly sampled component, $q(z|n)$ is close to 0, whereas $q(z|n)$ would be large if n is the component that z came from. So it is much better to sample this component and weight the probability appropriately. Thus, we can use a weighted version for estimating the function $\log q(z)$ during training. When provided with a mini-batch of samples $\{n_1, ..., n_M\}$, we can use the estimator as:

$$\mathbb{E}_{q(z)}[\log q(z)] \approx \frac{1}{M}\sum_{i=1}[\log \frac{1}{MN}\sum_{j=1}^{M} q(z(n_i)|n_i)], \tag{C.8}$$

where $z(n_i)$ is a sample from $q(z|n)$, and N is the count of the whole samples in dataset. M is the count of samples in a mini-batch.

C.2.4 Detailed description of avgMI

To evaluate the performance of the disentangled representation learning of the inference model, we adopt the metric avgMI proposed by [144]. The goal of avgMI is to evaluate the whether each latent variable w_k only capture the information of the relevant property y_k and has nothing correlation with the other properties. We utilize the MI matrix (i.e. the matrix

of pairwise mutual information between w and y) to represent the overall disentanglement performance. The optimal MI matrix should be like an identity matrix with diagonal entries all 1 and other entries all 0, where the mutual information between each w_k and y_k is 1 and the MI between w_k and other property y_j is 0. Then avgMI score is calculated as the distance between the real MI matrix and the optimal MI matrix. The smaller the avgMI is, the better the performance are.

Each entry, namely, mutual information $\text{MI}(w_i, y_j)$, in the mutual information matrix $I(x, y)$ is calculated as: $\text{MI}(w_i, y_j) = \sum_{w_i} \sum_{y_j} [(p(w_i, y_j) \log \frac{p(w_i, y_j)}{p(w_i)p(y_j)})]$. Therefore, to empirically estimate $p(w_i)$, $p(y_j)$, and $p(w_i, y_j)$, we need to have w and y in the experiments. And as we know, we have the observations on x and y, and w is generated from x by the encoder.

Bibliography

Bibliography

[1] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in neural information processing systems*, 2014, pp. 3581–3589.

[2] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in neural information processing systems*, 2015, pp. 2539–2547.

[3] A. Creswell, A. A. Bharath, and B. Sengupta, "Conditional autoencoders with adversarial information factorization," *space*, vol. 19, p. 24, 2017.

[4] J. Klys, J. Snell, and R. Zemel, "Learning latent subspaces in variational autoencoders," in *Advances in Neural Information Processing Systems*, 2018, pp. 6444–6454.

[5] A.-L. Barabási *et al.*, *Network science*. Cambridge university press, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[8] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," in *ICML*, 2019, pp. 2434–2444.

[9] C. Tran, W.-Y. Shin, A. Spitz, and M. Gertz, "Deepnc: Deep generative network completion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[10] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," *arXiv preprint arXiv:1711.08267*, 2017.

[11] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[12] G. Salha, S. Limnios, R. Hennequin, V.-A. Tran, and M. Vazirgiannis, "Gravity-inspired graph autoencoders for directed link prediction," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 589–598.

[13] S. Ranu and A. K. Singh, "Graphsig: A scalable approach to mining significant subgraphs in large graph databases," in *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 2009, pp. 844–855.

[14] M. Newman, *Networks*. Oxford university press, 2018.

[15] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[16] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: an approach to modeling networks." *Journal of Machine Learning Research*, vol. 11, no. 2, 2010.

[17] G. Robins, T. Snijders, P. Wang, M. Handcock, and P. Pattison, "Recent developments in exponential random graph (p*) models for social networks," *Social networks*, vol. 29, no. 2, pp. 192–215, 2007.

[18] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[19] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[22] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," *arXiv preprint arXiv:1803.10459*, 2018.

[23] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," *arXiv preprint arXiv:1802.03480*, 2018.

[24] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," *arXiv preprint arXiv:1803.03324*, 2018.

[25] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, 2017.

[26] X. Guo, L. Wu, and L. Zhao, "Deep graph translation," *arXiv preprint arXiv:1805.09980*, 2018.

[27] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *International Conference on Learning Representations*, 2016.

[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[29] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[30] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[31] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1025–1035.

[32] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML*, 2016, pp. 2014–2023.

[33] J. Atwood, S. Pal, D. Towsley, and A. Swami, "Sparse diffusion-convolutional neural networks," in *NIPS*, 2016.

[34] B. Wu, Y. Liu, B. Lang, and L. Huang, "Dgcnn: Disordered graph convolutional neural network based on the gaussian mixture model," *arXiv:1712.03563*, 2017.

[35] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *International Conference on Machine Learning*, 2018, pp. 5694–5703.

[36] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, "Grammar variational autoencoder," in *International Conference on Machine Learning*, 2017, pp. 1945–1954.

[37] H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song, "Syntax-directed variational autoencoder for structured data," *arXiv preprint arXiv:1802.08786*, 2018.

[38] B. Samanta, A. De, N. Ganguly, and M. Gomez-Rodriguez, "Designing random graph models using variational autoencoders with applications to chemical design," *arXiv preprint arXiv:1802.05283*, 2018.

[39] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," *arXiv preprint arXiv:1802.04364*, 2018.

[40] E. Neha, "Cyber security threats statistics around the world," *Identity*, 2017.

[41] K. Do, T. Tran, and S. Venkatesh, "Graph transformation policy network for chemical reaction prediction," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 750–760.

[42] M. Sun and P. Li, "Graph to graph: a topology aware approach for graph structures learning and generation," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 2946–2955.

[43] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 2. IEEE, 2005, pp. 729–734.

[44] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[45] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[46] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[47] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," *arXiv preprint arXiv:1809.10341*, 2018.

[48] Y. Bai, H. Ding, Y. Qiao, A. Marinovic, K. Gu, T. Chen, Y. Sun, and W. Wang, "Unsupervised inductive graph-level representation learning via graph-graph proximity," *International Joint Conferences on Artificial Intelligence*, 2019.

[49] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "Netgan: Generating graphs via random walks," *arXiv preprint arXiv:1803.00816*, 2018.

[50] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," *arXiv preprint arXiv:1806.09835*, 2018.

[51] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," *arXiv preprint arXiv:1704.04675*, 2017.

[52] K. Xu, L. Wu, Z. Wang, and V. Sheinin, "Graph2seq: Graph to sequence learning with attention-based neural networks," *arXiv preprint arXiv:1804.00823*, 2018.

[53] K. Xu, L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin, "Exploiting rich syntactic information for semantic parsing with graph-to-sequence model," *arXiv preprint arXiv:1808.07624*, 2018.

[54] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "A graph-to-sequence model for amr-to-text generation," *arXiv preprint arXiv:1805.02473*, 2018.

[55] Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," *arXiv preprint arXiv:1908.04942*, 2019.

[56] Y. Gao, L. Wu, H. Homayoun, and L. Zhao, "Dyngraph2seq: Dynamic-graph-to-sequence interpretable learning for health stage prediction in online health forums," *arXiv preprint arXiv:1908.08497*, 2019.

[57] X. Peng, D. Gildea, and G. Satta, "Amr parsing with cache transition systems," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[58] D. Gildea, G. Satta, and X. Peng, "Cache transition systems for graph parsing," *Computational Linguistics*, vol. 44, no. 1, pp. 85–118, 2018.

[59] Y. Wang, W. Che, J. Guo, and T. Liu, "A neural transition-based approach for semantic dependency graph parsing," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[60] W. Jin, K. Yang, R. Barzilay, and T. Jaakkola, "Learning multimodal graph-to-graph translation for molecular optimization," *arXiv preprint arXiv:1812.01070*, 2018.

[61] X. Guo, L. Zhao, C. Nowzari, S. Rafatirad, H. Homayoun, and S. M. P. Dinakarrao, "Deep multi-attributed graph translation with node-edge co-evolution," in *he 19th International Conference on Data Mining (ICDM 2019), pp. to appear*, 2019.

[62] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7398–7402.

[63] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1125–1134.

[64] H. Gao and S. Ji, "Graph u-nets," *arXiv preprint arXiv:1905.05178*, 2019.

[65] G. Nikolentzos, P. Meladianos, A. J.-P. Tixier, K. Skianis, and M. Vazirgiannis, "Kernel graph convolutional neural networks," *arXiv preprint arXiv:1710.10689*, 2017.

[66] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan, "Directed scale-free graphs," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2003, pp. 132–139.

[67] A. D. Kent, "Comprehensive, Multi-Source Cyber-Security Events," Los Alamos National Laboratory, 2015.

[68] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium *et al.*, "The wu-minn human connectome project: an overview," *Neuroimage*, vol. 80, pp. 62–79, 2013.

[69] P. Wang, R. Kong, X. Kong, R. Liégeois, C. Orban, G. Deco, M. P. Van Den Heuvel, and B. T. Yeo, "Inversion of a large-scale circuit model reveals a cortical hierarchy in the dynamic resting human brain," *Science advances*, vol. 5, no. 1, p. eaat7854, 2019.

[70] M. Jenkinson, C. F. Beckmann, T. E. Behrens, M. W. Woolrich, and S. M. Smith, "Fsl," *Neuroimage*, vol. 62, no. 2, pp. 782–790, 2012.

[71] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978.

[72] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. Sep, pp. 2539–2561, 2011.

[73] R. F. Galán, "On how network architecture determines the dominant patterns of spontaneous neural activity," *PloS one*, vol. 3, no. 5, p. e2148, 2008.

[74] F. Abdelnour, M. Dayan *et al.*, "Functional brain connectivity is predictable from anatomic network's laplacian eigen-structure," *NeuroImage*, vol. 172, pp. 728–739, 2018.

[75] S. Smith, M. Glasser, E. Robinson, G. Salimi-Khorshidi, E. Duff, D. Van Essen, M. Woolrich, M. Jenkinson, and C. Beckmann, "Methods for network modelling from high quality rfmri data," in *OHBM 2014 Annual Meeting*, 2014.

[76] SteveSmith, "Fslnets," https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FSLNets.

[77] F. Abdelnour, H. U. Voss, and A. Raj, "Network diffusion accurately models the relationship between structural and functional brain connectivity networks," *Neuroimage*, vol. 90, pp. 335–347, 2014.

[78] J. Meier, P. Tewarie, A. Hillebrand, L. Douw, B. W. van Dijk, S. M. Stufflebeam, and P. Van Mieghem, "A mapping between structural and functional brain networks," *Brain connectivity*, vol. 6, no. 4, pp. 298–311, 2016.

[79] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[80] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende *et al.*, "Interaction networks for learning about objects, relations and physics," in *Advances in neural information processing systems*, 2016, pp. 4502–4510.

[81] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[82] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[83] H. Sayadi, N. Patel *et al.*, "Ensemble learning for hardware-based malware detection: A comprehensive analysis and classification," in *ACM/EDAA/IEEE Design Automation Conference*, 2018.

[84] B. M. Smith, "A dual graph translation of a problem in 'life'," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2002, pp. 402–414.

[85] M.-L. Mugnier and M. Chein, "Conceptual graphs: Fundamental notions," *Revue d'intelligence artificielle*, vol. 6, no. 4, pp. 365–406, 1992.

[86] A. C. D. H. J. Hartmanis, T. Henzinger, J. H. N. J. T. Leighton, and M. Nivat, "Monographs in theoretical computer science an eatcs series," 2006.

[87] J. Bézivin and R. Heckel, "04101 abstracts collection–language engineering for model-driven software development," in *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2005.

[88] B. König and V. Kozioura, "Towards the verification of attributed graph transformation systems," in *International Conference on Graph Transformation*. Springer, 2008, pp. 305–320.

[89] D. Plump and S. Steinert, "Towards graph programs for graph algorithms," in *International Conference on Graph Transformation*. Springer, 2004, pp. 128–143.

[90] H. Ehrig, U. Prange, and G. Taentzer, "Fundamental theory for typed attributed graph transformation," in *International conference on graph transformation*. Springer, 2004, pp. 161–177.

[91] D. Corbett, "Conceptual graph theory applied to reasoning in ontologies," 2002.

[92] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[93] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[94] S. F. Mousavi, M. Safayani, A. Mirzaei, and H. Bahonar, "Hierarchical graph embedding in vector space by graph pyramid," *Pattern Recognition*, vol. 61, pp. 245–254, 2017.

[95] J. Kawahara, C. J. Brown, S. P. Miller, B. G. Booth, V. Chau, R. E. Grunau, J. G. Zwicker, and G. Hamarneh, "Brainnetcnn: convolutional neural networks for brain networks; towards predicting neurodevelopment," *NeuroImage*, vol. 146, pp. 1038–1049, 2017.

[96] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations." in *AAAI*, 2016, pp. 1145–1152.

[97] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[98] L. Wu, I. E.-H. Yen, Z. Zhang, K. Xu, L. Zhao, X. Peng, Y. Xia, and C. Aggarwal, "Scalable global alignment graph kernel using random features: From node embedding to graph embedding," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1418–1428.

[99] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[100] Y. Gao, X. Guo, and L. Zhao, "Local event forecasting and synthesis using unpaired deep graph translations," in *Proceedings of the 2nd ACM SIGSPATIAL Workshop on Analytics for Local Events and News*. ACM, 2018, p. 5.

[101] B. B. Kivilcim, I. O. Ertugrul *et al.*, "Modeling brain networks with artificial neural networks," in *Graphs in Biomedical Image Analysis and Integrating Medical Imaging and Non-Imaging Modalities*. Springer, 2018, pp. 43–53.

[102] P. Sturmfels, S. Rutherford *et al.*, "A domain guided cnn architecture for predicting age from structural brain images," *arXiv preprint arXiv:1808.04362*, 2018.

[103] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[104] J. Gallier, "Spectral theory of unsigned and signed graphs. applications to graph clustering: a survey," *arXiv preprint arXiv:1601.04692*, 2016.

[105] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, 2005.

[106] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[107] C. M. López, "Chip firing and the tutte polynomial," *Annals of Combinatorics*, vol. 1, no. 1, pp. 253–259, 1997.

[108] A.-L. Barabási *et al.*, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[109] M. R. Guthaus, J. S. Ringenberg *et al.*, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*. IEEE, 2001, pp. 3–14.

[110] J. L. Henning, "Spec cpu2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[111] H. S. e. a. P. D. Sai Manoj, "Lightweight node-level malware detection and network-level malware confinement in IoT networks," in *ACM/EDAA/IEEE Design Automation and Test in Europe (DATE)*, 2019.

[112] H. Sayadi and H. M. et al, "2SMaRT: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *ACM/EDAA/IEEE Design Automation and Test in Europe (DATE)*, 2019.

[113] D. M. Lowe, "Extraction of chemical structures and reactions from the literature," Ph.D. dissertation, University of Cambridge, 2012.

[114] W. Jin, C. Coley, R. Barzilay, and T. Jaakkola, "Predicting organic reaction outcomes with weisfeiler-lehman network," in *NeurIPS*, 2017, pp. 2607–2616.

[115] J. Ma, C. Zhou, P. Cui, H. Yang, and W. Zhu, "Learning disentangled representations for recommendation," in *NeurIPS*, 2019, pp. 5712–5723.

[116] Y. Liu, X. Wang, S. Wu, and Z. Xiao, "Independence promoted graph disentangled networks," 2019.

[117] N. Stoehr, M. Brockschmidt, and et al, "Disentangling interpretable generative parameters of random and real-world graphs," 2019.

[118] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *ICML-Volume 32*, 2014, pp. II–1278.

[119] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework." *ICLR*, vol. 2, no. 5, p. 6, 2017.

[120] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[121] T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Advances in Neural Information Processing Systems*, 2018, pp. 2610–2620.

[122] H. Kim and A. Mnih, "Disentangling by factorising," *arXiv preprint arXiv:1802.05983*, 2018.

[123] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[124] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017.

[125] S. Zhao, J. Song, and S. Ermon, "Infovae: Information maximizing variational autoencoders," *arXiv preprint arXiv:1706.02262*, 2017.

[126] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational inference of disentangled latent concepts from unlabeled observations," *arXiv preprint arXiv:1711.00848*, 2017.

[127] R. Lopez, J. Regier, M. I. Jordan, and N. Yosef, "Information constraints on auto-encoding variational bayes," in *Advances in Neural Information Processing Systems*, 2018, pp. 6114–6125.

[128] B. Esmaeili, H. Wu, S. Jain, and et al, "Structured disentangled representations," in *AISTATS*, 2019, pp. 2525–2534.

[129] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[130] Y. Chen, L. Wu, and M. J. Zaki, "Graphflow: Exploiting conversation flow with graph neural networks for conversational machine comprehension," *IJCAI*, 2020.

[131] K. Shen, L. Wu, F. Xu, S. Tang, J. Xiao, and Y. Zhuang, "Hierarchical attention based spatial-temporal graph-to-sequence learning for grounded video description," *IJCAI*, 2020.

[132] A. LeClair, S. Haque, L. Wu, and C. McMillan, "Improved code summarization via a graph neural network," *MSR*, 2020.

[133] Q. Li, A. Alipour-Fanid, M. Slawski, Y. Ye, L. Wu, K. Zeng, and L. Zhao, "Large-scale cost-aware classification using feature computational dependency graph," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[134] A. Makhzani and B. J. Frey, "Pixelgan autoencoders," in *NeurIPS*, 2017, pp. 1975–1985.

[135] S. Watanabe, "Information theoretical analysis of multivariate correlation," *IBM Journal of research and development*, vol. 4, no. 1, pp. 66–82, 1960.

[136] K. Ridgeway and M. C. Mozer, "Learning deep disentangled embeddings with the f-statistic loss," in *NeurIPS*, 2018, pp. 185–194.

[137] C. Eastwood and C. K. Williams, "A framework for the quantitative evaluation of disentangled representations," 2018.

[138] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *ICML*, 2019, pp. 4114–4124.

[139] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," in *Advances in neural information processing systems*, 2016, pp. 2352–2360.

[140] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 10–21.

[141] A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, and A. Zhavoronkov, "drugan: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico," *Molecular pharmaceutics*, vol. 14, no. 9, pp. 3098–3104, 2017.

[142] L. Zhao, "Event prediction in big data era: A systematic survey," *arXiv preprint arXiv:2007.09815*, 2020.

[143] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2017.

[144] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem, "Disentangling factors of variations using few labels," in *International Conference on Learning Representations*, 2019.

[145] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, "Disentangled graph convolutional networks," in *ICML*, 2019, pp. 4212–4221.

[146] X. Guo, L. Zhao, Z. Qin, L. Wu, A. Shehu, and Y. Ye, "Interpretable deep graph generation with node-edge co-disentanglement," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1697–1707.

[147] L. Zhang, L. Zhao, S. Qin, and D. Pfoser, "Tg-gan: Deep generative models for continuously-time temporal graph generation," *arXiv preprint arXiv:2005.08323*, 2020.

[148] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader networks: Manipulating images by sliding attributes," in *Advances in neural information processing systems*, 2017, pp. 5967–5976.

[149] P. K. Gyawali, B. M. Horacek, J. L. Sapp, and L. Wang, "Sequential factorized autoencoder for localizing the origin of ventricular activation from 12-lead electrocardiograms," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 5, pp. 1505–1516, 2019.

[150] H. Edwards and A. Storkey, "Censoring representations with an adversary," *arXiv preprint arXiv:1511.05897*, 2015.

[151] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[152] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun, "Disentangling factors of variation in deep representation using adversarial training," in *Advances in neural information processing systems*, 2016, pp. 5040–5048.

[153] S. Zhou, T. Xiao, Y. Yang, D. Feng, Q. He, and W. He, "Genegan: Learning object transfiguration and attribute subspace from unpaired data," *arXiv preprint arXiv:1705.04932*, 2017.

[154] T. Xiao, J. Hong, and J. Ma, "Dna-gan: Learning disentangled representations from multi-attribute images," *Workshop of International conference on machine learning*, 2017.

[155] ——, "Elegant: Exchanging latent encodings with gan for transferring multiple face attributes," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 168–184.

[156] J. Behrmann, W. Grathwohl, R. T. Chen, D. Duvenaud, and J.-H. Jacobsen, "Invertible residual networks," in *International Conference on Machine Learning*, 2019, pp. 573–582.

[157] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, "dsprites: Disentanglement testing sprites dataset," *URL https://github. com/deepmind/dsprites-dataset/.[Accessed on: 2018-05-08]*, 2017.

[158] C. Burgess and H. Kim, "3d shapes dataset," https://github.com/deepmind/3dshapes-dataset/, 2018.

[159] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific data*, vol. 1, no. 1, pp. 1–7, 2014.

[160] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. W. Battaglia, "Learning deep generative models of graphs," *CoRR*, vol. abs/1803.03324, 2018. [Online]. Available: http://arxiv.org/abs/1803.03324

[161] Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt, "Constrained graph variational autoencoders for molecule design," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 7795–7804.

[162] R. N. Abskharon, G. Giachin, and et al, "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*, vol. 136, no. 3, pp. 937–944, 2014.

[163] Q. Liu, M. Allamanis, M. Brockschmidt, and A. L. Gaunt, "Constrained graph variational autoencoders for molecule design," *The Thirty-second Conference on Neural Information Processing Systems*, 2018.

[164] R. E. Caflisch *et al.*, "Monte carlo and quasi-monte carlo methods," *Acta numerica*, vol. 1998, pp. 1–49, 1998.

Curriculum Vitae

Xiaojie Guo is a Ph.D. candidate at the Department of Information Science and Technology at George Mason University. She received her M.S. degree and B.E. degree in Control Engineering from Soochow University, China. Her research interests include data mining, artificial intelligence, and machine learning, with special interests in deep learning on graphs, graph transformation and generation, and interpretable representation learning. She has published over 16 papers in top-tier conferences and journals such as KDD, ICDM, ICLR, AAAI, CIKM, and KAIS. She won the Best Paper Award in ICDM 2019 and has one paper awarded as an ESI Hot and Highly Cited Paper as the first author. Xiaojie has also served as an independent peer reviewer for multiple top academic journals, such as the IEEE Transactions on Neural Networks and Learning Systems (TNNLS), IEEE Transactions on Knowledge Discovery from Data (TKDD), International Journal of Intelligent System (IJIS), as well as workshops of conferences including KDD and AAAI. She also has served as a student volunteer in many top-tier conferences such as KDD, ICDM, and CKIM, and has won three Student Travel Awards.