BAYESIAN MACHINE LEARNING ON GRAPHS

A Dissertation

by

ARMAN HASANZADEHMOGHIMI

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Krishna Narayanan
Co-Chair of Committee,	Nick Duffield
Committee Members,	Jean-Francois Chamberland
	Matthias Katzfuss
Head of Department,	Miroslav M. Begovic

December 2021

Major Subject: Electrical Engineering

ABSTRACT

In this dissertation, we propose novel Bayesian machine learning models to solve various graph analytics problems, including graph representation learning, graph generative modeling, structured semi-supervised learning, and relational learning. Our proposed methods model different components of a graph including nodes, node attributes, and the graph structure, as *distributions*. More specifically, our proposed methods are Bayesian generative models with robust variational inference and hence are equipped with natural uncertainty estimates.

First, we propose ***Semi-Implicit Graph Variational Autoencoders*** (SIG-VAE) (Chapter 3) for probabilistic representation learning in graph-structured data. SIG-VAE employs a hierarchical variational framework to enable neighboring node distribution sharing for better generative modeling of graph dependency structure, together with a Bernoulli-Poisson link decoder. SIG-VAE integrates a carefully designed generative model, well suited to model real-world sparse graphs, and a sophisticated semi-implicit variational inference network, which propagates the graph structural information and distribution uncertainty to capture complex posteriors which may exhibit heavy tails, multiple modes, and skewness. SIG-VAE provides highly interpretable latent representations and significantly outperforms state-of-the-art methods on several different graph analytic tasks.

In addition, we propose ***Bayesian Graph Neural Networks with Graph DropConnect*** (Chapter 4) by introducing a unified framework for adaptive connection sampling in graph neural networks (GNNs), called Graph DropConnect (GDC), that generalizes existing stochastic regularization methods for training GNNs. The proposed framework not only alleviates over-smoothing and over-fitting tendencies of deep GNNs, but also enables learning with uncertainty in graph analytic tasks with GNNs. Instead of using fixed sampling rates or hand-tuning them as model hyperparameters as in existing stochastic regularization methods, our GDC can be trained jointly with GNN model parameters. GNN training with GDC is shown to be mathematically equivalent to an efficient approximation of training Bayesian GNNs.

Finally, we propose ***MoReL: Multi-modal Relational Learning*** (Chapter 5) to infer hidden relations among features in heterogeneous views using a fused Gromov-Wasserstein (FGW) regularization between latent representations of corresponding views. Such an optimal transport regularization in the deep Bayesian generative model not only allows incorporating view-specific side information, either with graph-structured or unstructured data in different views, but it also increases model flexibility with the distribution-based regularization. We apply MoReL to integrative analysis in multi-omics data inferring molecular interactions.

DEDICATION

To my mom and dad.

ACKNOWLEDGMENTS

I would like to sincerely thank my academic advisors, Professor Krishna Narayanan and Professor Nick Duffield, for their help, advice, and unconditional encouragement and support. I also thank Professor Xiaoning Qian and Professor Mingyuan Zhou for their constructive comments and good-natured support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors Krishna Narayanan [advisor], Nick Duffield [co-advisor] and Jean-Francois Chamberland of the Department of Electrical and Computer Engineering and Professor Matthias Katzfuss of the Department of Statistics.

All work for the dissertation was completed by the student, under the advisement of Professors Krishna Narayanan, and Nick Duffield of the Department of Electrical and Computer Engineering.

Funding Sources

My work was supported by the National Science Foundation, through NSF Grants CCF-1934904, ECCS-1839816, and IIS-1848596.

NOMENCLATURE

NN	Neural Network
CNN	Convolutional Neural Network
GNN	Graph Neural Network
GCN	Graph Convolutional Networks
VI	Variational Inference
ELBO	Evidence Lower Bound
KL	Kullback–Leibler Divergence
VGAE	Variational Graph Autoencoder
GAE	Graph Autoencoder
SIVI	Semi-implicit Variational Inference
SIG-VAE	Semi-Implicit Graph Variational Auto-Encoder
NF	Normalizing Flow
MCMC	Markov Chain Monte-Carlo
GDC	Graph DropConnect
UQ	Uncertainty Quantification
BNN	Bayesian Neural Network
ARM	Augment Reinforce Merge
MoReL	Multi-Modal Relational Learning
WD	Wasserstein Distance
GWD	Gromov-Wasserstein Distance
FGWD	Fused Gromov-Wasserstein Distance
CCA	Canonical Correlation Analysis

PCCA	Probabilistic Canonical Correlation Analysis
BCCA	BayesianCanonical Correlation Analysis
SRCA	Spearman's Rank Correlation Analysis
CF	Cystic Fibrosis
rRNA	ribosomal RNA
TCGA	The Cancer Genome Atlas
BRCA	Breast Cancer
GRN	Gene Regulatory Networks
AML	Acute Myeloid Leukemia
ROC	Receiver Operating Characteristic
PR	Precision-Recall
AUC	Area Under the Curve
CONCAT	Concatenation Operator

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Real-world data often exhibits structure that can be formalized as a graph with complex interactions and relationships between objects. Images, transportation networks, social networks, and gene co-expression networks are a few example datasets that can be modeled as graphs, where each node represents an agent (e.g., pixel, road intersection, user, and gene) and the edges manifest the interactions between the agents [28, 45, 76]. Analyzing graph data is an important machine learning task with a wide variety of applications. For example, rumor source detection in social networks could be formulated as semi-supervised node classification problem on graphs, while drug repurposing can be cast as link prediction problem in a bipartite graph.

A key problem in machine learning is- how can we incorporate *structure* of data into models? A common way, specifically in deep learning, is introducing architectural inductive bias into a deep neural network. These biases structure the computations in a deep neural network to faithfully capture the underlying structure in data. For example, convolutional neural networks (CNNs) take advantage of the grid structure in data and learn localized features using shared kernels. Graph neural networks (GNNs) [28, 56, 60] follow the same principle to extract local features through localized computations, usually in the form of message passing between nodes. Due to high expressive power and scalable computations, GNN-based approaches enjoyed immense success in graph analytic tasks. Link prediction [55, 117], representation learning [110, 13, 43], and node classification [99, 111, 61] are some of the graph analytic tasks that have been addressed with such models.

Despite their empirical success over traditional graph analytic methods, the vast majority of GNN-based models can be viewed as *deterministic functions*. These models produce point estimates of parameters and predictions, and do not produce uncertainty information nor possess interpretability of Bayesian probabilistic models. Model uncertainty and Bayesian interpretability are indispensable in many graph applications usually involving *decision making*. For example, testing link prediction results in drug-disease graphs requires expensive and prolonged pharmaceutical

tests. Knowing the confidence level of prediction in such scenarios is invaluable.

The main goal of this dissertation to develop Bayesian probabilistic models for high-dimensional graph-structured data, addressing a variety of important graph analytic tasks. Our proposed models offer interpertability of probabilistic models while enjoying the scalability of deep models. Our contributions are guided by the following research questions:

- **Research Question 1:** *Can we develop a scalable unsupervised Bayesian graph representation model than infers complex implicit posteriors?*

  In Chapter 3, we introduce a novel unsupervised graph representation learning model called *semi-implicit graph variational autoencoder* (SIG-VAE). Combining the advantages of semi-implicit hierarchical variational distribution and variational graph autoencoder with a Bernoulli-Poisson link decoder, SIG-VAE enriches the representation power of the posterior distribution of node embedding given graphs so that both the graph structural and node attribute information can be best captured in the latent space. By providing a surrogate evidence lower bound that is asymptotically exact, the SIG-VAE model inference is amenable via stochastic gradient descent, without compromising the flexbility of its variational distribution. Our experiments with different graph datasets shows the promising capability of SIG-VAE in a range of graph analysis applications with interpretable latent representations, thanks to the hierarchical construction that diffuses the distributions of neighborhood nodes in given graphs.

- **Research Question 2:** *How can we determine model uncertainty in graph neural networks? Can we address over-smoothing and over-fitting at the same time?*

  In Chapter 4, we propose a unified framework for adaptive connection sampling in GNNs that generalizes existing stochastic regularization techniques for training GNNs. Our proposed method, *Graph DropConnect* (GDC), not only alleviates over-smoothing and over-fitting tendencies of deep GNNs, but also enables learning with uncertainty in graph analytic tasks with GNNs. Instead of using fixed sampling rates, our GDC technique parameters can be

trained jointly with GNN model parameters. We further show that training a GNN with GDC is equivalent to an approximation of a Bayesian GNN. We demonstrate significant advantages of GDC both in semi-supervised node classification and uncertainty quantification.

- **Research Question 3:** *Can we develop a generative model to learn hidden relations among features in heterogeneous views of data without any pre-known relations across views?*

  In Chapter 5, we have introduce *multi-modal relational learning* (MoReL), a novel Bayesian deep generative model that efficiently infers hidden relations across heterogeneous views of data. By using a fused Gromov-Wasserstein (FGW) based decoder, MoReL is more flexible compared to the existing relational learning models and can integrate both structured and unstructured views of data while accounting for arbitrarily permutation and/or transformation caused by processing features with different deep functions across the views. Our experiments on real-world biological datasets demonstrates substantial improvement in inferring meaningful relations as well as improving prediction sensitivity compared to the competing methods.

2. BACKGROUND

In this chapter, we will provide a brief introduction to a few topics that are extensively used in the dissertation.

2.1 Graph neural networks

Graph neural networks are a class deep models designed to process data defined over nodes of a graph[1]. Given the graph structure and node features, a GNN layer computes the new set of features based on the input features and graph structure. GNNs could be divided into two main categories: 1) *spectral* GNNs which are defined in the graph frequency domain, and 2) *message-passing*-based GNNs that are defined in the graph domain. While spectral GNNs offer more interpertable functions, message-passing-based GNNs show better scalability and are more suitable for large-scale graphs. Figure 2.1 depicts the computational graph of a two layer message-passing-based GNN. In this dissertation, we focus on message-passing-based GNNs and refer to them as GNNs.

Assume that $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of an undirected graph with n nodes, $\mathbf{D} \in \mathbb{R}^{n \times n}$ is the diagonal degree matrix, and $\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \dots, \mathbf{h}_n^{(l)}]$ are nodes' attributes at layer l with $\mathbf{H}^{(0)} \in \mathbb{R}^{n \times k_0}$ being the input features. Then, a general GNN layer, without skip connections, is defined as:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(f\left(\mathbf{h}_i^{(l)}, \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\, \mathbf{h}_j^{(l)}\right)\right) \tag{2.1}$$

where $\mathcal{N}(i)$ is the first-hop neighborhood of node i, $\alpha_{i,j}$ is the non-learnable or learnable (depending on the GNN architecture) weighting factor that is a function of graph structure, f is the aggregation function, and σ is the activation function. The main difference between GNN architectures is the aggregation function. The most commonly used aggregation function is mean followed by a linear mapping which leads to a architecture known as graph convolutional neural networks (GCN) [56].

[1]There are several GNN architectures that could incorporate edge features. We do not include them here, as they are out of scope of this dissertation.

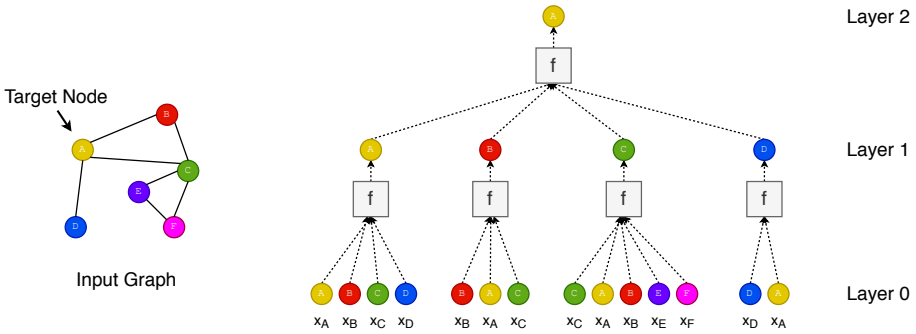

Figure 2.1: Illustration of the computational graph of a two layer message-passing-based GNN. $\mathbf{x}$ denotes the node attributes, and f denotes the aggregation function.

More precisely, a GCN layer is defined as follows [43]:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{MEAN}\left(\{\mathbf{h}_i^{(l)}\} \cup \{\mathbf{h}_j^{(l)}\}_{j \in \mathcal{N}(i)}\right)\right), \tag{2.2}$$

where $\mathbf{W}$ is learnable weight matrix. With a few simple mathematical calculations, we can re-formulate a GCN layer as follows [56]:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\,\tilde{\mathbf{A}}\,\tilde{\mathbf{D}}^{\frac{1}{2}}\,\mathbf{H}^{(l)}\,\mathbf{W}^{(l)}\right), \tag{2.3}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$, and $\tilde{\mathbf{D}}$ is a diagonal matrix with $\tilde{D}_{i,i} = \sum_j A_{i,j}$.

While any GNN architecture could be used in our proposed models, we will mostly use GCN in our experiments. Further details will be discussed in each chapter.

2.2 Bayesian models and variational inference

In machine learning, we are interested in learning probabilistic models that faithfully describes the data. The main assumption in probabilistic models is uncertainty over values of the variables. This probabilistic view requires learning probability distributions of variables as opposed to point estimates in deterministic models. A methodical approach to incorporate *a priori* knowledge about variables in the model, is through a *Bayesian framework*. In Bayesian models, we choose a *prior distribution* over the *latent* variable. This prior distribution summarizes our knowledge about

variables. In the learning process, we update the prior distribution to *posterior distribution* using the observed data. In Bayesian framework, the distribution of data is governed by the latent variables. Therefore, learning the posterior distribution of latent variables and conditional distribution of data, completely describes the marginal distribution of data. More specifically, assume $\mathbf{x}$ is the observed data and $\mathbf{z}$ is the latent (hidden) variable. The marginal distribution of data, also known as *evidence*, is defined as follows:

$$p(\mathbf{x}) = \int p(\mathbf{x} \,|\, \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}. \tag{2.4}$$

Inference in Bayesian models develop into computing the posterior of latent variables $p(\mathbf{z} \,|\, \mathbf{x})$. There are two main inference methods for Bayesian models: 1) Monte-Carlo methods, and 2) Variational Inference (VI). Monte-Carlo methods estimate the posterior from the samples obtained from a Markov chain with the posterior as the stationary distribution. Although Monte-Carlo methods are exact, they are computationally expensive and generally cannot be deployed in high-dimensional problems. Variational inference approximates the posterior by solving an optimization problem. Specifically, VI infers the optimized member of a family of distributions $\mathcal{H}$ by solving the following optimization:

$$\arg\min_{q(\mathbf{z}) \in \mathcal{H}} \mathrm{KL}\left(q(\mathbf{z}) \,||\, p(\mathbf{z} \,|\, \mathbf{x})\right), \tag{2.5}$$

where KL is Kullback-Leibler divergence. Although the above objective is intuitive, it is intractable. Therefore, an alternative objective, which is commonly knwon as Evidence Lower Bound (ELBO), is used to infer the approximate posterior. ELBO is defined as follows:

$$\mathrm{ELBO}(q) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x} \,|\, \mathbf{z})] - \mathrm{KL}[q(\mathbf{z}) \,||\, p(\mathbf{z})], \tag{2.6}$$

where $\mathbb{E}_{q(\mathbf{z})}$ is expectation with respect to $q(\mathbf{z})$.

In order to use VI, we have to parametrize conditional data distribution $p(\mathbf{x} \,|\, \mathbf{z})$, and approximate

posterior $q(\mathbf{z})$. In classical VI approximate posterior distributions are optimized for each sample, hence the number of parameters linearly grows with the number of observations. A common solution, known as amortized VI or autoencoding variational Bayes, for circumventing this problem is learning efficient mapping from samples to parameters of the approximate posterior distributions. This mapping is usually parametrized by a deep neural network.

We will use Bayesian modeling and VI in all of our proposed methods. We will further discuss this topic in each chapter.

3. SEMI-IMPLICIT GRAPH VARIATIONAL AUTO-ENCODERS*

3.1 Introduction

The main challenge for analyzing graph datasets for link prediction, clustering, or node classification, is how to deploy graph structural information in the model. Graph representation learning aims to summarize the graph structural information by a feature vector in a low-dimensional latent space, which can be used in downstream analytic tasks.

While the vast majority of existing methods assume that each node is embedded to a deterministic point in the latent space [7, 2, 81, 92, 38, 43, 17], modeling uncertainty is of crucial importance in many applications, including physics and biology. For example, when link prediction in Knowledge Graphs is used for driving expensive pharmaceutical experiments, it would be beneficial to know what is the confidence level of a model in its prediction. To address this, variational graph auto-encoder (VGAE) [55] embeds each node to a random variable in the latent space. Despite its popularity, 1) the Gaussian assumption imposed on the variational distribution restricts its variational inference flexibility when the true posterior distribution given a graph clearly violates the Gaussian assumption; 2) the adopted inner-product decoder restricts its generative model flexibility. While recent study tries to address the first problem by changing the prior distribution but does not show much practical success [27], the latter one is not well-studied yet to the best of our knowledge.

Inspired by recently developed semi-implicit variational inference (SIVI) [113] and normalizing flow (NF) [85, 54, 78], which offer the interesting combination of flexible posterior distribution and effective optimization, we propose a hierarchical variational graph framework for node embedding of graph structured data, notably increasing the expressiveness of the posterior distribution for each node in the latent space. SIVI enriches mean-field variational inference with a flexible (implicit) mixing distribution. NF transforms a simple Gaussian random variable through a sequence of invertible differentiable functions with tractable Jacobians. While NF restricts the mixing

*Reprinted with permission from "Semi-Implicit Graph Variational Auto-Encoders" by A. Hasanzadeh, E. Hajiramezanali, K. Narayanan, N. Duffield, M. Zhou, and X. Qian. In Advances in Neural Information Processing Systems, 2019. Copyright 2019 by authors.

distribution in the hierarchy to have an explicit probability density function, SIVI does not impose such a constraint. Both SIVI and NF can model complex posterior distributions, which will help when the underlying true embedded node distribution exhibits heavy tails and/or multiple modes. We further argue that the graph structure cannot be fully exploited by the posterior distribution from the trivial combination of SIVI and/or NF with VGAE, if not integrating graph neighborhood information. On the other hand, it does not address the flexibility of the generative model as stated as the second VGAE problem.

To address the aforementioned issues, instead of explicitly choosing the posterior distribution family in previous works [55, 27], our hierarchical variational framework adopts a stochastic generative node embedding model that can learn implicit posteriors while maintaining simple optimization. Specifically, we innovate a semi-implicit hierarchical construction to model the posterior distribution to best fit both the graph topology and node attributes given graphs. With SIVI, even if the posterior is not tractable, its density can be evaluated with Monte Carlo estimation, enabling efficient model inference on top of highly enhanced model flexibility/expressive power. Our semi-implicit graph variational auto-encoder (SIG-VAE) can well model heavy tails, skewness, multimodality, and other characteristics that are exhibited by the posterior but failed to be captured by existing VGAEs. Furthermore, a Bernoulli-Poisson link function [119] is adopted in the decoder of SIG-VAE to increase the flexibility of the generative model and better capture graph properties of real-world networks that are often sparse. SIG-VAE facilitates end-to-end learning for various graph analytic tasks evaluated in our experiments. For link prediction, SIG-VAE consistently outperforms state-of-the-art methods by a large margin. It is also comparable with state-of-the-arts when modified to perform two additional tasks, node classification and graph clustering, even though node classification is more suitable to be solved by supervised learning methods. We further show that the new decoder is able to generate sparse random graphs whose statistics closely resemble those of real-world graph data. These results clearly demonstrate the great practical values of SIG-VAE.

3.2 Related works

Variational graph auto-encoders (VGAE), proposed by Kipf and Welling [55], embed each node to a random variable in the latent space. VGAE, by extending the use of VAEs to graph structured data, is shown to be capable of learning interpretable latent representations for undirected graphs and getting competitive results in the link prediction task. However, the Gaussian assumption imposed on the variational distribution restricts the model flexibility when the true posterior distribution given a graph clearly violates the assumption. It also suffers from underestimating the variance of the posterior, which is a well-known issue of vanilla VAEs.

To better model graph data using variational distributions in VGAEs, Davidson et al. [27] proposes the hyperspherical VGAE ($\mathcal{S}$-VGAE), in which, instead of the Gaussian assumption for the posterior, the von Mises-Fisher distribution has been deployed. This assumption is not well-suited for all classes of graphs. For example, it has been proven that graphs with hierarchical tree-like structure have hyperbolic latent structures [75] which clearly cannot be represented well in a hyperspherical space. While $\mathcal{S}$-VGAE outperforms vanilla VGAE in some graphs including Cora and Citeseer in terms of link prediction accuracy, its performance will be degraded for more complex graphs such as Pubmed. On the other hand, changing the prior is not going to change the flexibility and optimal solution of the generative model, but will affect the tightness of the ELBO and hence how well the generative model parameters can be inferred. This shows the necessity to develop a variational graph auto-encoders that not only is capable of inferring more flexible posteriors to represent a broader range of graphs, but also is able to have more flexible decoder especially for the real-world sparse graphs.

3.3 Background

3.3.1 Variational graph auto-encoder (VGAE).

Many node embedding methods derive deterministic latent representations [38, 43, 17]. By expanding the variational auto-encoder (VAE) notion to graphs, [55] propose to solve the following problem by embedding the nodes to Gaussian random vectors in the latent space.

Problem 1. Given a graph $G = (\mathcal{V}, \mathcal{E})$ with the adjacency matrix $\mathbf{A}$ and M-dimensional node attributes $\mathbf{X} \in \mathbb{R}^{N \times M}$, find the probability distribution of the latent representation of nodes $\mathbf{Z} \in \mathbb{R}^{N \times \mathscr{L}}$, i.e., $p(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A})$.

Finding the true posterior, $p(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A})$, is often difficult and intractable. In [55], it is approximated by a Gaussian distribution, $q(\mathbf{Z} \,|\, \psi) = \prod_{i=1}^{N} q_i(\mathbf{z}_i \,|\, \psi_i)$ and $q_i(\mathbf{z}_i \,|\, \psi_i) = \mathcal{N}(\mathbf{z}_i \,|\, \psi_i)$ with $\psi_i = \{\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2)\}$. Here, $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are l-dimensional mean and standard deviation vectors corresponding to node i, respectively. The parameters of $q(\mathbf{Z} \,|\, \psi)$, i.e., $\psi = \{\psi_i\}_{i=1}^{N}$, are modeled and learned using two graph convolutional neural networks (GCNs) [56]. More precisely, $\boldsymbol{\mu} = \mathbf{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A})$, $\log \boldsymbol{\sigma} = \mathbf{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$ and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are matrices of $\boldsymbol{\mu}_i$'s and $\boldsymbol{\sigma}_i$'s, respectively. Given $\mathbf{Z}$, the decoder in VGAE is a simple inner-product decoder as $p(A_{i,j} = 1 \,|\, \mathbf{z}_i, \mathbf{z}_j) = \text{sigmoid}(\mathbf{z}_i \, \mathbf{z}_j^T)$.

The parameters of the model are found by optimizing the well known evidence lower bound (ELBO) [49, 11, 12, 102]: $\mathcal{L} = \mathbb{E}_{q(\mathbf{Z} \,|\, \psi)}[p(\mathbf{A} \,|\, \mathbf{Z})] - \text{KL}[q(\mathbf{Z} \,|\, \psi) \,||\, p(\mathbf{Z})]$. Note that $q(\mathbf{Z} \,|\, \psi)$ here is equivalent to $q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A})$. Despite promising results shown by VGAE, a well-known issue in variational inference is underestimating the variance of the posterior. The reason behind this is the mismatch between the representation power of the variational family to which q is restricted and the complexity of the true posterior, in addition to the use of KL divergence, which is asymmetric, to measure how different q is from the true posterior.

3.3.2 Semi-implicit variational inference (SIVI).

To well characterize the posterior while maintaining simple optimization, semi-implicit variational inference (SIVI) has been proposed by [113], which is also related to the hierarchical variational inference [84] and auxiliary deep generative models [65]; see [113] for more details about their connections and differences. It has been shown that SIVI can capture complex posterior distributions like multimodal or skewed distributions, which can not be captured by a vanilla VI due to its restricted exponential family assumption over both the prior and posterior in the latent space. SIVI assumes that ψ, the parameters of the posterior, are drawn from an implicit distribution rather than being analytic. This hierarchical construction enables flexible mixture modeling and allows to have more complex posteriors while maintaining simple optimization for model inference. More

specifically, $\mathbf{Z} \sim q(\mathbf{Z} \mid \psi)$ and $\psi \sim q_\phi(\psi)$ with ϕ denoting the distribution parameters to be inferred. Marginalizing ψ out leads to the random variables $\mathbf{Z}$ drawn from a distribution family $\boldsymbol{\mathcal{H}}$ indexed by variational parameters ϕ, expressed as

$$\boldsymbol{\mathcal{H}} = \left\{ h_{\boldsymbol{\phi}}(\mathbf{Z}) \ : \ h_{\boldsymbol{\phi}}(\mathbf{Z}) = \int_{\psi} q(\mathbf{Z} \mid \psi) q_{\boldsymbol{\phi}}(\psi) \, d\psi \right\}. \tag{3.1}$$

The importance of semi-implicit formulation is that while the original posterior $q(\mathbf{Z} \mid \psi)$ is explicit and analytic, the marginal distribution, $h_{\boldsymbol{\phi}}(\mathbf{Z})$ is often implicit. Note that, if $q_{\boldsymbol{\phi}}$ equals a delta function, then $h_{\boldsymbol{\phi}}$ is an explicit distribution. Unlike regular variational inference that assumes independent latent dimensions, semi-implicit does not impose such a constraint. This enables the semi-implicit variational distributions to model very complex multivariate distributions.

Since the marginal probability density function $h_{\boldsymbol{\phi}}(\mathbf{Z})$ is often intractable, SIVI derives a lower bound for ELBO, as follows, to optimize the variational parameters.

$$\begin{aligned}
\mathcal{L} = \mathbb{E}_{\mathbf{Z} \sim h_{\boldsymbol{\phi}}(\mathbf{Z})} \Big[\log \frac{p(\mathbf{Y}, \mathbf{Z})}{h_{\boldsymbol{\phi}}(\mathbf{Z})} \Big] &= -\mathrm{KL}(\mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi)}[q(\mathbf{Z} \mid \psi)] \mid\mid p(\mathbf{Z} \mid \mathbf{Y})) + \log p(\mathbf{Y}) \\
&\geq -\mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi)} \mathrm{KL}(q(\mathbf{Z} \mid \psi) \mid\mid p(\mathbf{Z} \mid \mathbf{Y})) + \log p(\mathbf{Y}) \\
&= \mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi)} \Big[\mathbb{E}_{\mathbf{Z} \sim q(\mathbf{Z} \mid \psi)} \Big[\log \left(\frac{p(\mathbf{Y}, \mathbf{Z})}{q(\mathbf{Z} \mid \psi)} \right) \Big] \Big] = \underline{\mathcal{L}}(q(\mathbf{Z} \mid \psi), q_{\boldsymbol{\phi}}(\psi)),
\end{aligned} \tag{3.2}$$

where $\mathbf{Y}$ is the observations. The inequality $\mathbb{E}_\psi \mathrm{KL}(q(\mathbf{Z} \mid \psi) || p(\mathbf{Z})) \geq \mathrm{KL}(\mathbb{E}_\psi[q(\mathbf{Z}|\psi)] || p(\mathbf{Z}))$ has been used to derive $\underline{\mathcal{L}}$. Optimizing this lower bound, however, could drive the mixing distribution $q_{\boldsymbol{\phi}}(\psi)$ towards a point mass density. To address the degeneracy issue, SIVI adds a nonnegative regularization term, leading to a surrogate ELBO that is asymptotically exact [113]. We will further discuss this in the Appendix A.

3.3.3 Normalizing flow (NF).

NF [78] also enriches the posterior distribution families. Compared to SIVI, NF imposes explicit density functions for the mixing distributions in the hierarchy while SIVI only requires $q_{\boldsymbol{\phi}}$ to be

reparameterizable. This makes SIVI more flexible, especially when using it for graph analytics as explained in the next section, since the SIVI posterior can be generated by transforming random noise using any flexible function, for example a neural network.

3.4 Baselines: variational inference with VGAE

Before presenting our semi-implicit graph variational auto-encoder (SIG-VAE), we first introduce two baseline methods that directly combines SIVI and NF with VGAE.

3.4.1 SIVI-VGAE

To address **Problem 1** while well characterizing the posterior with modeling flexibility in the VGAE framework, the naive solution is to take the semi-implicit variational distribution in SIVI for modeling latent variables in VGAE, following the hierarchical formulation

$$\mathbf{Z} \sim q(\mathbf{Z} \,|\, \psi), \qquad \psi \sim q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A}), \tag{3.3}$$

by introducing the implicit prior distribution parametrized by ψ, which can be sampled from the reparametrizable $q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A})$. Such a hierarchical semi-implicit construct not only leads to flexible mixture modeling of the posterior but also enables efficient model inference, for example, with ϕ being parameterized by deep neural networks. In this framework, the features from multiple layers of GNNs can be aggregated and then transformed via multiple fully connected layers after being concatenated by random noise to derive the posterior distribution for each node separately. More specifically, SIVI-VGAE injects random noise at C different stochastic fully connected layers for each node independently:

$$\mathbf{h}_u = \text{GNN}_u(\mathbf{A}, \text{CONCAT}(\mathbf{X}, \mathbf{h}_{u-1})), \quad \text{for } u = 1, \ldots, L, \ \mathbf{h}_0 = \mathbf{0}$$

$$\boldsymbol{\ell}_t^{(i)} = \boldsymbol{T}_t(\boldsymbol{\ell}_{t-1}^{(i)}, \boldsymbol{\epsilon}_t, \mathbf{h}_L^{(i)}), \quad \text{where } \boldsymbol{\epsilon}_t \sim q_t(\boldsymbol{\epsilon}) \text{ for } t = 1, \ldots, C, \ \boldsymbol{\ell}_0^{(i)} = \mathbf{0}$$

$$\boldsymbol{\mu}_i(\mathbf{A}, \mathbf{X}) = \boldsymbol{g}_\mu(\boldsymbol{\ell}_C^{(i)}, \mathbf{h}_L^{(i)}), \ \boldsymbol{\Sigma}_i(\mathbf{A}, \mathbf{X}) = \boldsymbol{g}_\Sigma(\boldsymbol{\ell}_C^{(i)}, \mathbf{h}_L^{(i)}),$$

$$q(\mathbf{Z} \,|\, \mathbf{A}, \mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \textstyle\prod_{i=1}^N q(\mathbf{z}_i \,|\, \mathbf{A}, \mathbf{X}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad q(\mathbf{z}_i \,|\, \mathbf{A}, \mathbf{X}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{A}, \mathbf{X}), \boldsymbol{\Sigma}_i(\mathbf{A}, \mathbf{X})),$$

where $\boldsymbol{T}_t, \boldsymbol{g}_\mu$, and $\boldsymbol{g}_\sigma$ are all deterministic neural networks, i is the node index, L is the number of GNN layers, and $\boldsymbol{\epsilon}_t$ is random noise drawn from the distribution q_t. Note that in the equations above, GNN is any type of existing graph neural networks, such as graph convolutional neural network (GCN) [56], GCN with Chebyshev filters [28], GraphSAGE [43], jumping knowledge (JK) networks [110], and graph isomorphism network (GIN) [111]. Given the GNN_L output $\mathbf{h}_L$, $\boldsymbol{\mu}_i(\mathbf{A}, \mathbf{X})$ and $\boldsymbol{\Sigma}_i(\mathbf{A}, \mathbf{X})$ are now random variables rather than following vanilla VAE to assume deterministic values.

In this way, however, the constructed implicit distributions may not capture the dependency between neighboring nodes completely. Note that we consider SIVI-VGAE as a naive version of our proposed SIG-VAE (and call it as *naive SIG-VAE* in the rest of the dissertation), which is specifically designed with neighborhood sharing to capture complex dependency structures in networks, as detailed in the next section. Please also note that the first layer of SIVI can be integrated with NF rather than simple Gaussian. We leave that for future study.

3.4.2 NF-VGAE

It is also possible to enable VGAE model flexibility by other existing variational inference methods, for example using NF. However, NF requires deterministic transform functions whose Jacobians shall be easy to compute, which limits the flexibility when considering complex dependency structures in graph analytic tasks. We indeed have constructed a non-Gaussian VGAE, i.e. NF-based variational graph auto-encoder (NF-VGAE) as follows

$$\mathbf{h}_u = \text{GNN}_u(\mathbf{A}, \text{CONCAT}(\mathbf{X}, \mathbf{h}_{u-1})), \quad \text{for } u = 1, \ldots, L, \ \mathbf{h}_0 = \mathbf{0} \tag{3.4}$$

$$\boldsymbol{\mu}(\mathbf{A}, \mathbf{X}) = \text{GNN}_\mu(\mathbf{A}, \text{CONCAT}(\mathbf{X}, \mathbf{h}_L)), \quad \boldsymbol{\Sigma}(\mathbf{A}, \mathbf{X}) = \text{GNN}_\Sigma(\mathbf{A}, \text{CONCAT}(\mathbf{X}, \mathbf{h}_L)),$$

$$q_0(\mathbf{Z}^{(0)} \,|\, \mathbf{A}, \mathbf{X}) = \textstyle\prod_{i=1}^N q_0(\mathbf{z}_i^{(0)} \,|\, \mathbf{A}, \mathbf{X}), \quad \text{with} \quad q_0(\mathbf{z}_i^{(0)} \,|\, \mathbf{A}, \mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2)),$$

$$q_K(\mathbf{Z}^{(K)} \,|\, \mathbf{A}, \mathbf{X}) = \textstyle\prod_{i=1}^N q_0(\mathbf{z}_i^{(K)} | \mathbf{A}, \mathbf{X}), \quad \ln(q_K(\mathbf{z}_i^{(K)} \,|\, -)) = \ln(q_0(\mathbf{z}_i^{(0)})) - \sum_k \ln|\det \frac{\partial f_k}{\partial \mathbf{z}_i^{(k)}}|,$$

where the posterior distribution $q_K(\mathbf{Z}^{(K)}|\mathbf{A}, \mathbf{X})$ is obtained by successively transforming a Gaussian random variable $\mathbf{Z}^{(0)}$ with distribution q_0 through a chain of K invertible differentiable transfor-

mations $f_k : \mathbb{R}^d \to \mathbb{R}^d$. We will further discuss this in the Appendix A. NF-VGAE is a two-step inference method that 1) starts with Gaussian random variables and then 2) transforms them through a series of invertible mappings. We emphasize again that in NF-VGAE, the **GNN** output layers are deterministic without neighborhood distribution sharing due to the deterministic nature of the initial density parameters in q_0.

3.5 Semi-implicit graph variational auto-encoder (SIG-VAE)

While the above two models are able to approximate more flexible and complex posterior, such trivial combinations may fail to fully exploit graph dependency structure because they are not capable of propagating uncertainty between neighboring nodes. To enable effective uncertainty propagation, which is the essential factor to capture complex posteriors with graph data, we develop a *carefully* designed generative model, SIG-VAE, to better integrate variational inference and VGAE with a natural neighborhood sharing scheme.

To have tractable posterior inference, we construct SIG-VAE using a hierarchy of multiple stochastic layers. Specifically, the first stochastic layer $q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A})$ is reparameterizable and has an analytic probability density function. The layers added after are reparameterizable and computationally efficient to sample from. More specifically, we adopt a hierarchical encoder in SIG-VAE that injects random noise at L different stochastic layers:

$$\mathbf{h}_u = \text{GNN}_u(\mathbf{A}, \text{CONCAT}(\mathbf{X}, \boldsymbol{\epsilon}_u, \mathbf{h}_{u-1})), \quad \text{where } \boldsymbol{\epsilon}_u \sim q_u(\boldsymbol{\epsilon}) \text{ for } u = 1, \ldots, L, \ \mathbf{h}_0 = \mathbf{0} \tag{3.5}$$

$$\boldsymbol{\mu}(\mathbf{A}, \mathbf{X}) = \text{GNN}_\mu(\mathbf{A}, \text{CONCAT}(\mathbf{X}, \mathbf{h}_L)), \quad \boldsymbol{\Sigma}(\mathbf{A}, \mathbf{X}) = \text{GNN}_\Sigma(\mathbf{A}, \text{CONCAT}(\mathbf{X}, \mathbf{h}_L)), \tag{3.6}$$

$$q(\mathbf{Z} \,|\, \mathbf{A}, \mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \textstyle\prod_{i=1}^N q(\mathbf{z}_i \,|\, \mathbf{A}, \mathbf{X}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad q(\mathbf{z}_i \,|\, \mathbf{A}, \mathbf{X}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{A}, \mathbf{X}), \boldsymbol{\Sigma}_i(\mathbf{A}, \mathbf{X})).$$

Note that in the equations above $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are random variables and thus $q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A})$ is not necessarily Gaussian after marginalization; $\boldsymbol{\epsilon}_u$ is N-dimensional random noise drawn from a distribution q_u; and q_u is chosen such that the samples drawn from it are the same type as $\mathbf{X}$, for example if $\mathbf{X}$ is categorical, Bernoulli is a good choice for q_u. By concatenating the random noise and node attributes, the output of GNNs are random variables rather than deterministic vectors. Their expressive power

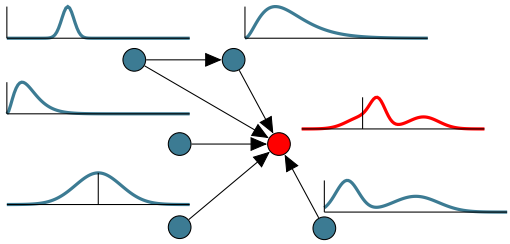

Figure 3.1: SIG-VAE diffuses the neighboring nodes' distributions, which is more informative than sharing deterministic features, to infer each node's latent distribution.

is inherited in SIG-VAE to go beyond Gaussian, exponential family, or von Mises-Fisher [27] posterior distributions for the derived latent representations.

In SIG-VAE, when inferring each node's latent posterior, we incorporate the distributions of the neighboring nodes, better capturing graph dependency structure than sharing deterministic features from GNNs. More specifically, the input to our model at stochastic layer u is CONCAT$(\mathbf{X}, \boldsymbol{\epsilon}_u)$ so that the outputs of the subsequent stochastic layers give mixing distributions by integrating information from neighboring nodes (Figure 3.1). The flexibility of SIG-VAE directly working on the stochastic distribution parameters in (5-6) allows neighborhood sharing to achieve better performance in graph analytic tasks. We argue that the uncertainty propagation in our *carefully* designed SIG-VAE, which is the an outcome of using GNNs and adding noise in the input in equations (5-6), is the key factor in capturing more faithful and complex posteriors. Note that (5) is different from the NF-VAE construction (3), where the GNN output layers are deterministic. Through experiments, we show that this uncertainty neighborhood sharing is key for SIG-VAE to achieve superior graph analysis performance.

We further argue that increasing the flexibility of variational inference is not enough to better model real-world graph data as the optimal solution of the generative model does not change. In SIG-VAE, the Bernoulli-Poisson link [119] is adopted for the decoder to further increase the expressiveness of the generative model. Potential extensions with other decoders can be integrated with SIG-VAE if needed. Let $A_{i,j} = \delta(m_{ij} > 0)$, $m_{ij} \sim \text{Poisson}\big(\exp(\sum_{k=1}^{l} r_k z_{ik}\, z_{jk})\big)$, and

hence

$$p(\mathbf{A} \mid \mathbf{Z}, \mathbf{R}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{i,j} \mid \mathbf{z}_i, \mathbf{z}_j, \mathbf{R}), \quad p(A_{i,j} = 1 \mid \mathbf{z}_i, \mathbf{z}_j, \mathbf{R}) = 1 - e^{-\exp\left(\sum_{k=1}^{\mathscr{L}} r_k z_{ik}\ z_{jk}\right)}, \tag{3.7}$$

where $\mathbf{R} \in \mathbb{R}_{+}^{\mathscr{L} \times \mathscr{L}}$ is a diagonal matrix with diagonal elements r_k.

3.5.1 Inference

To derive the ELBO for model inference in SIG-VAE, we must take into account the fact that ψ has to be drawn from a distribution. Hence, the ELBO moves beyond the simple VGAE as

$$\begin{aligned} \mathcal{L} &= -\mathrm{KL}(\mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})}[q(\mathbf{Z} \mid \psi)] \,||\, p(\mathbf{Z})) + \mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})}[\mathbb{E}_{\mathbf{Z} \sim q(\mathbf{Z} \mid \psi)}[\log p(\mathbf{A} \mid \mathbf{Z})]] \\ &= \mathbb{E}_{\mathbf{Z} \sim h_{\boldsymbol{\phi}}(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})}\Big[\log \frac{p(\mathbf{A} \mid \mathbf{Z}) p(\mathbf{Z})}{h_{\boldsymbol{\phi}}(\mathbf{Z} \mid \mathbf{X}, \mathbf{A})}\Big]. \end{aligned} \tag{3.8}$$

Direct optimization of the ELBO in SIVI is not tractable [113], so the Monte Carlo estimation of the ELBO, $\mathcal{L}$, is prohibited. To address this issue, SIVI derives a lower bound for the ELBO and optimizes this lower bound instead of optimizing the ELBO itself, which is tractable and asymptotically equals to the ELBO. SIG-VAE requires $q(\mathbf{Z} \mid \psi)$ to be explicit, and also requires it to either be reparameterizable or the ELBO under $q(\mathbf{Z} \mid \psi)$ to be analytic, while $q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})$ is required to be reparameterizable but not necessarily explicit. This captures the idea that combining an explicit $q(\mathbf{Z} \mid \psi)$ with an implicit $q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})$ is as powerful as needed, but makes the computation tractable.

Following Yin and Zhou [113], we can derive a lower bound for the ELBO as follows

$$\begin{aligned} \underline{\mathcal{L}} &= \mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})}\Big[\mathbb{E}_{\mathbf{Z} \sim q(\mathbf{Z} \mid \psi)}\Big[\log\left(\frac{p(\mathbf{A}|\mathbf{Z}) p(\mathbf{Z})}{q(\mathbf{Z} \mid \psi)}\right)\Big]\Big] \\ &= -\mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})}[\mathrm{KL}(q(\mathbf{Z} \mid \psi) \,||\, p(\mathbf{Z}))] + \mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})}\big[\mathbb{E}_{\mathbf{Z} \sim q(\mathbf{Z} \mid \psi)}[\log p(\mathbf{A} \mid \mathbf{Z})]\big] \leq \mathcal{L}. \end{aligned}$$

This can be proved based on the first theorem in Yin and Zhou [113], which shows

$$\mathrm{KL}(\mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})}[q(\mathbf{Z} \mid \psi)] \,||\, p(\mathbf{Z})) \leq \mathbb{E}_{\psi \sim q_{\boldsymbol{\phi}}(\psi \mid \mathbf{X}, \mathbf{A})}[\mathrm{KL}(q(\mathbf{Z} \mid \psi) \,||\, p(\mathbf{Z}))].$$

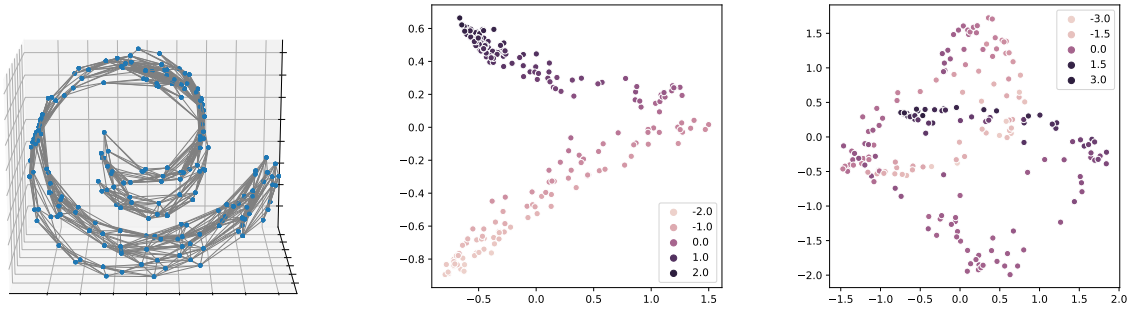

Figure 3.2: Swiss roll graph (**Left**) and its latent representation using SIG-VAE (**Middle**) and VGAE (**Right**). The latent representations (middle and right) are heat maps in $\mathbb{R}^3$. We expect that the embedding of the Swiss roll graph with inner-product decoder to be a curved plane in $\mathbb{R}^3$, which is clearly captured better by SIG-VAE.

Unlike $\mathcal{L}$, a Monte Carlo estimation of $\underline{\mathcal{L}}$ only requires $q_{\boldsymbol{\phi}}(\mathbf{Z} \,|\, \psi)$ to have an analytic density functions and $q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A})$ to be convenient to sample from.

Directly optimizing $\underline{\mathcal{L}}$ without early stopping could lead to a point mass density as $q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A})$. This degenerates SIG-VAE to the vanilla VGAE. To avoid degeneracy, a regularization term can be added to $\underline{\mathcal{L}}$. Assume that K samples are drawn from $q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A})$ denoted by $\{\psi^{(i)}\}_{i=1}^{K}$. We define a regularized lower bound as $\underline{\mathcal{L}}_K = \underline{\mathcal{L}} + B_K$ where

$$B_K = \mathbb{E}_{\psi, \psi^{(1)}, \ldots, \psi^{(K)} \sim q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A})}[\mathrm{KL}(q(\mathbf{A} \,|\, \psi) \,||\, \tilde{h}_K(\mathbf{Z}))],$$

and

$$\tilde{h}_K(\mathbf{Z})) = \frac{q_{\boldsymbol{\phi}}(\psi \,|\, \mathbf{X}, \mathbf{A}) + \sum_{k=1}^{K} q_{\boldsymbol{\phi}}(\psi^{(k)} \,|\, \mathbf{X}, \mathbf{A})}{K+1}.$$

It has been proved by Molchanov et al. [69] that $\underline{\mathcal{L}}_K$ is a monotonic lower bound of the ELBO, satisfying $\underline{\mathcal{L}}_K \leq \underline{\mathcal{L}}_{K+1} \leq \underline{\mathcal{L}}$. Therefore, setting K to zero means that $\underline{\mathcal{L}}_0 = \underline{\mathcal{L}}$, and as K goes to infinity $\underline{\mathcal{L}}$ converges to the exact ELBO, i.e., $\lim_{K \to \infty} \underline{\mathcal{L}}_K = \mathcal{L}$.

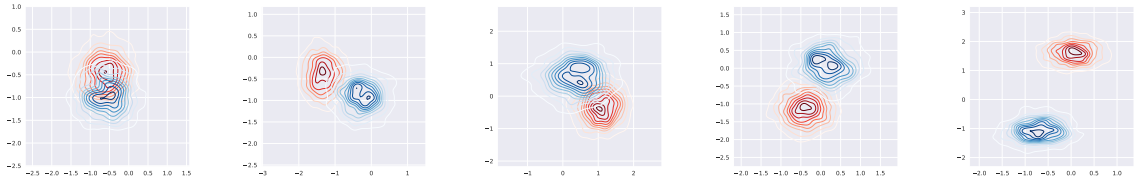

Figure 3.3: Latent representation distributions of five example nodes from the Swiss roll graph using SIG-VAE (**Blue**) and VGAE (**Red**). SIG-VAE clearly infers more complex distributions that can be multi-modal, skewed, and with sharp and steep changes. This helps SIG-VAE to better represent the nodes in the latent space.

3.6 Experiments

We test the performances of SIG-VAE on different graph analytic tasks: 1) interpretability of SIG-VAE compared to VGAE, 2) link prediction in various real-world graph datasets including graphs with node attributes and without node attributes, 3) graph generation, 4) node classification in the citation graphs with labels. In all of the experiments, GCN [56] is adopted for all the GNN modules in SIG-VAE, Naive SIG-VAE, and NF-VGAE, implemented in Tensorflow [1]. The PyGSP package [29] is used to generate synthetic graphs. Implementation details for all the experiments, together with graph data statistics, can be found in the Appendix A.

3.6.1 Interpretable latent representations

We first demonstrate the expressiveness of SIG-VAE by illustrating the approximated variational distributions of node latent representations. We show that SIG-VAE captures the graph structure better and has a more interpretable embedding than VGAE on a generated Swiss roll graph with 200 nodes and 1244 edges (Figure 3.2). In order to provide a fair comparison, both models share an identical implementation with the inner-product decoder and same number of parameters. We simply consider the identity matrix $\mathbf{I}_N$ as node attributes and choose the latent space dimension to be three in this experiment. This graph has a simple plane like structure. As the inner-product decoder assumes that the information is embedded in the *angle* between latent vectors, we expect that the node embedding to map nodes of the Swiss roll graph into a curve in the latent space. As

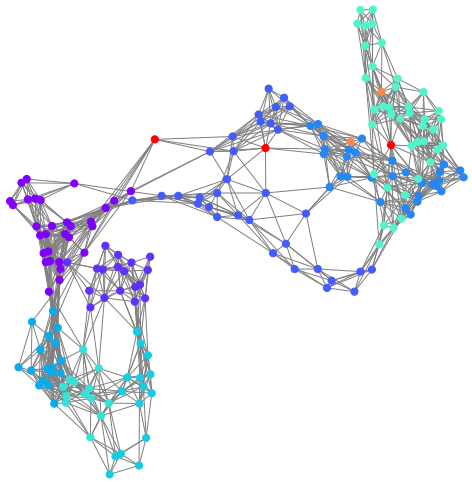

Figure 3.4: The nodes with multi-modal posteriors (red nodes) reside between different communities in Swiss Roll graph.

we can see in Figure 3.2, SIG-VAE derives a clearly more interpretable planar latent structure than VGAE.

We also show the posterior distributions of five randomly selected nodes from the graph in Figure 3.3. As we can see, SIG-VAE is capable of inferring complex distributions. The inferred distributions can be multi-modal, skewed, non-symmetric, and with sharp and steep changes. These complex distributions help the model to get a more realistic embedding capturing the intrinsic graph structure. To explain why multi-modality may arise, we used Asynchronous Fluid [79] to visualize the Swiss Roll graph by highlighting detected communities with different colors in Figure 3.4. Note that we used a different layout from the one in Figure 3.2(a) to better visualize the communities in the graph. The three red (two orange) nodes are the nodes with multi-modal (skewed) distributions in Figure 3.3. These nodes with multi-modal posteriors reside between different communities; hence, with a probability, they could be assigned to multiple communities. The Appendix A contains additional results and discussions with a torus graph, with similar observations.

3.6.2 Accurate link prediction

We further conduct extensive experiments for link prediction with various real-world graph datasets. Our results show that SIG-VAE significantly outperforms well-known baselines and

Table 3.1: Link prediction performance in networks with node attributes.

Method	**Cora**		**Citeseer**		**Pubmed**	
	AUC	AP	AUC	AP	AUC	AP
SC [93]	84.6±0.01	88.5±0.00	80.5±0.01	85.0±0.01	84.2±0.02	87.8±0.01
DW [81]	83.1±0.01	85.0±0.00	80.5±0.02	83.6±0.01	84.4±0.00	84.1±0.00
GAE [55]	91.0±0.02	92.0±0.03	89.5±0.04	89.9±0.05	96.4±0.00	96.5±0.00
VGAE [55]	91.4±0.01	92.6±0.01	90.8±0.02	92.0±0.02	94.4±0.02	94.7±0.02
$\mathcal{S}$-VGAE [27]	94.10±0.1	94.10±0.3	94.70±0.2	95.20±0.2	96.00±0.1	96.00±0.1
SEAL [117]	90.09±0.1	83.01±0.3	83.56±0.2	77.58±0.2	96.71±0.1	90.10±0.1
G2G [13]	92.10±0.9	92.58±0.8	95.32±0.7	95.57±0.7	94.28±0.3	93.38±0.5
NF-VGAE	92.42±0.6	93.08±0.5	91.76±0.3	93.04±0.8	96.59±0.3	96.68±0.4
Naive SIG-VAE	93.97±0.5	93.29±0.4	94.25±0.8	93.60±0.9	96.53±0.7	96.01±0.5
SIG-VAE (IP)	**94.37**±0.1	**94.41**±0.1	**95.90**±0.1	**95.46**±0.1	**96.73**±0.1	**96.67**±0.1
SIG-VAE	**96.04**±0.04	**95.82**±0.06	**96.43**±0.02	**96.32**±0.02	**97.01**±0.07	**97.15**±0.04

state-of-the-art methods in all benchmark datasets. We consider two types of datasets, i.e., datasets with node attributes and datasets without attributes. We preprocess and split the datasets as done in [55] with validation and test sets containing 5% and 10% of network links, respectively. We learn the model parameters for 3500 epochs with the learning rate 0.0005 and the validation set used for early stopping. The latent space dimension is set to 16. The hyperparameters of SIG-VAE, Naive SIG-VAE, and NF-VGAE are the same for all the datasets. For fair comparison, all methods have the similar number of parameters as the default VGAE. The Appendix A contains further implementation details. We measure the performance by average precision (AP) and area under the ROC curve (AUC) based on 10 runs on a test set of previously removed links in these graphs.

With node attributes. We consider three graph datasets with node attribbutes—Citeseer, Cora, and Pubmed [88]. The number of node attributes for these dataset are 3703, 1433, and 500 respectively. Other statistics of the datasets are summarized in the Table A.1. We compare the results of SIG-VAE, Naive SIG-VAE, and NF-VGAE with six state-of-the-art methods, including spectral clustering (SC), DeepWalk (DW) [81] , GAE [55], VGAE [55], $\mathcal{S}$-VGAE [27], and SEAL [117]. The inner-product decoder is also adopted in SIG-VAE to clearly demonstrate the advantages of the semi-implicit hierarchical variational distribution for the encoder.

Table 3.2: AUC and AP of link prediction in networks without node attributes. * indicates that the numbers are reported from [117]. The Appendix A contains the complete result tables with standard deviation values.

Metrics	Data	MF*	SBM*	N2V*	LINE*	SC*	GAE	VGAE*	SEAL*	G2G	**NF-VGAE**	**N-SIG-VAE**	**SIG-VAE**(IP)	**SIG-VAE**
AUC	**USAir**	94.08	94.85	91.44	81.47	74.22	93.09	89.28	97.09	92.17	95.74	94.22	**97.56**	94.52
	NS	74.55	92.30	91.52	80.63	89.94	93.14	94.04	97.71	98.18	98.38	98.00	**98.75**	**99.17**
	Yeast	90.28	91.41	93.67	87.45	93.25	93.74	93.88	97.20	97.34	97.86	93.36	**98.11**	**98.32**
	Power	50.63	66.57	76.22	55.637	91.78	72.21	71.20	84.18	91.35	94.61	93.67	**95.04**	**96.23**
	Router	78.03	85.65	65.46	67.15	68.79	55.73	61.51	95.68	85.98	93.56	92.66	**95.94**	**96.13**
AP	**USAir**	94.36	95.08	89.71	79.70	78.07	95.14	89.27	95.70	90.22	96.27	94.48	**97.50**	94.95
	NS	78.41	92.13	94.28	85.17	90.83	95.26	95.83	98.12	97.43	98.52	97.83	**98.53**	**99.24**
	Yeast	92.01	92.73	94.90	90.55	94.63	95.34	95.19	97.95	97.83	98.18	94.24	**97.97**	**98.41**
	Power	53.50	65.48	81.49	56.66	91.00	77.13	75.91	86.69	92.29	95.76	93.80	**96.50**	**97.28**
	Router	82.59	84.67	68.66	71.92	73.53	67.50	70.36	95.66	86.28	95.88	92.80	**94.94**	**96.86**

We use the same hyperparameters for the competing methods as stated in [117, 55, 27]. As we can see in Table 3.1, SIG-VAE shows significant improvement in terms of both AUC and AP over state-of-the-art methods. Note the standard deviation of SIG-VAE is also smaller compared to other methods, indicating stable semi-implicit variational inference. Compared to the baseline VGAE, more flexible posterior in three proposed methods SIGVAE (with both inner-product and Bernoulli-Poisson link decoders), Naive SIG-VAE, and NF-VGAE can clearly improve the link prediction accuracy. This suggests that the Gaussian assumption does not hold for these graph structured data. The performance improvement of SIG-VAE with inner-product decoder (IP) over Naive SIG-VAE and NF-VGAE clearly demonstrates the advantages of neighboring node sharing, especially in the smaller graphs. Even for the large graph Pubmed, on which VGAE performs similar to $\mathcal{S}$-VGAE, our SIG-VAE still achieves the highest link prediction accuracy, showing the importance of all modeling components in the proposed method including non-Gaussian posterior, using neighborhood distribution, and the sparse Bernoulli-Poisson link decoder.

Without node attributes. We further consider five graph datasets without node attributes—USAir, NS [74], Router [90], Power [103] and Yeast [100]. The data statistics are summarized in the Table A.1. We compare the performance of our models with seven competing state-of-the-art methods including matrix factorization (MF), stochastic block model (SBM) [3], node2vec (N2V) [38], LINE [92], spectral clustering (SC), VGAE [55], $\mathcal{S}$-VGAE [27], and SEAL [117].

For baseline methods, we use the same hyperparameters as stated in Zhang et al. [117].

Table 3.3: Graph generation performance. The closest results to the original graph is highlighted in boldface.

Detasets	Orignial Graph		VGAE		**SIG-VAE** (IP)		**SIG-VAE**	
	Dens.	Clus.	Dens.	Clus.	Dens.	Clus.	Dens.	Clus.
Cora	0.00143	0.24	0.1178	0.49	0.1178	0.49	**0.00147**	**0.25**
Citeseer	0.0008	0.14	0.09	0.45	0.26	0.42	**0.0008**	**0.16**
USAir	0.038	0.62	0.18	0.40	0.21	**0.56**	**0.043**	0.45
NS	0.002	0.63	0.36	0.47	0.26	0.42	**0.02**	**0.49**
Router	0.0004	0.01	0.16	0.49	0.16	0.49	**0.0010**	**0.09**

For datasets without node attributes, we use a two-stage learning process for SIG-VAE. First, the embedding of each node is learned in the 128-dimensional latent space while injecting 5-dimensional Bernoulli noise to the system. Then the learned embedding is taken as node features for the second stage to learn 16 dimensional embedding while injecting 64-dimensional noise to SIG-VAE. Through empirical experiments, we found that this two-stage learning converges faster than end-to-end learning. We follow the same procedure for Naive SIG-VAE and NF-VGAE.

As we can see in Table 3.2, SIG-VAE again shows the consistent superior performance compared to the competing methods, especially over the baseline VGAE, in both AUC and AP. It is interesting to note that, while the proposed Berhoulli-Poisson decoder works well for sparser graphs, especially NS and Router datasets, SIG-VAE with inner-product decoder shows superior performance for the USAir graph which is much denser. Compared to the baseline VGAE, both Naive SIG-VAE and NF-VGAE improve the results with a large margin in both AUC and AP, showing the benefits of more flexible posterior. Comparing SIG-VAE with two other flexible inference methods shows not only SIG-VAE is not restricted to the Gaussian assumption, which is not a good fit for link prediction with the inner-product decoder [27], but also it is able to model flexible posterior considering graph topology. The results for the link prediction of the Power graph clearly magnifies this fact as SIG-VAE improves the accuracy by 34% compared to VGAE. The Appendix A contains the results with standard deviation values over different runs, showing the stability again.

Ablation studies have also been run to evaluate SIG-VAE with inner-product decoder in link

Table 3.4: Summary of results in terms of classification accuracy (in %).

Method	**Cora**	**Citeseer**	**Pubmed**
ManiReg [8]	59.5	60.1	70.7
SemiEmb [104]	59.0	59.6	71.1
LP [121]	68.0	45.3	63.0
DeepWalk [81]	67.2	43.2	65.3
ICA [62]	75.1	69.1	73.9
Planetoid [112]	75.7	64.7	77.2
GCN [56]	**81.5**	70.3	79.0
SIG-VAE	79.7	**70.4**	**79.3**

prediction for citation graphs without using node attributes. The [AUC, AP] are [91.14, 90.99] for Cora and [88.72, 88.24] for Citeseer, lower than the values from SIG-VAE with attributes in Table 3.1 but are still competitive against existing methods (even with node attributes), showing the ability of SIG-VAE of utilizing graph structure. While some of the methods, like SEAL, work well for graphs without node attributes and some of others, like VGAE, get good performance for graphs with node attributes, SIG-VAE consistently achieves superior performance in both types of datasets. This is due to the fact that SIG-VAE can learn implicit distributions for nodes, which are very powerful in capturing graph structure even without any node attributes.

3.6.3 Graph generation

To further demonstrate the flexibility of SIG-VAE as a generative model, we have used the inferred embedding representations of different graph datasets with and without node attributes to generate new graphs. Results are summarized in Table 3.3. The SIG-VAE results are much closer to the real-world graph in terms of both graph density and average clustering for very sparse graphs. For example, SIG-VAE infers network parameters for Cora whose density and average clustering coefficients are 0.00143 and 0.24, respectively. Using the inferred posterior and learned decoder, a new graph is generated with corresponding r_k to see if its graph statistics are close to the original ones. Please note that we have shrunk inferred r_k's smaller than 0.01 to 0. The density and average clustering coefficients of this generated graph based on SIG-VAE are 0.00147 and 0.25, respectively,

Table 3.5: Graph clustering performance in citation networks with label.

Method	Cora		Citeseer	
	NMI	ACC	NMI	ACC
VGAE	0.43	59.2	0.20	51.5
SIG-VAE	**0.58**	**68.8**	**0.34**	**57.4**

which are very close to the original graph. We also generate new graphs based on SIG-VAE with the inner-product decoder and VGAE. The density and average clustering coefficients of the generated graphs based on SIG-VAE (IP) and VGAE are same, i.e. 0.1178 and 0.49, respectively, showing the inner-product decoder may not be a good choice for sparse graphs.

For the USAir dataset, which is much dense compare to othe graphs, the average clustering coefficient of SIG-VAE with inner-product decoder is closer to the read-world graph. This can be describe the better link prediction results of SIG-VAE (IP) for USAir dataset. On the other hand, the generated graph by SIG-VAE with the Bernoulli-Poisson link decoder is much sparser as its density is very closer to the read-world graph. This shows the benefit of the proposed decoder to improve the flexibility of the generative model.

3.6.4 Node classification and graph clustering

We also have applied SIG-VAE for node classification on citation graphs with labels by modifying the loss function to include graph reconstruction and semi-supervised classification terms. Results are summarized in Table 3.4. Our model exhibits strong generalization properties, highlighted by its competitive performance compared to the state-of-the-art methods, despite not being trained specifically for this task. To show the robustness of SIG-VAE to missing edges, we randomly removed 10, 20, 50 and 70 (%) edges while keeping node attributes. The mean accuracy of 10 run for Cora (2 layers [32,16]) are 79.5, 78.7, 75.3 and 60.6, respectively.

SIG-VAE can be applied in the other application including graph clustering. We first tried SIG-VAE for getting low-dimential feature space and then apply Gaussian mixture clustering (GMM) on citation graphs with labels including Cora and Citeseer and compare its results with VGAE.

We consider same number of parameters and GCN layer for both model. Results are summarized in Table 3.5. We report the normalized mutual information (NMI) and unsupervised clustering accuracy (ACC) of 10 runs. The decoders for both methods are inner-product decoder.

SIG-VAE has demonstrated state-of-the-art performances in link prediction and comparable results on other tasks, clearly showing the potential of SIG-VAE on different graph analytic tasks.

4. BAYESIAN GRAPH NEURAL NETWORKS*

4.1 Introduction

Graph neural networks (GNNs), and its numerous variants, have shown to be successful in graph representation learning by extracting high-level features for nodes from their topological neighborhoods. GNNs have boosted the state-of-the-art performance in a variety of graph analytic tasks, such as semi-supervised node classification and link prediction [56, 55, 46, 40]. Despite their successes, GNNs have two major limitations: 1) they cannot go very deep due to *over-smoothing* and *over-fitting* phenomena [61, 56]; 2) the current implementations of GNNs do not provide uncertainty quantification (UQ) of output predictions.

There exist a variety of methods to address these problems. For example, DropOut [91] is a popular regularisation technique with deep neural networks (DNNs) to avoid over-fitting, where network units are randomly masked during training. In GNNs, DropOut is realized by randomly removing the node features during training [87]. Often, the procedure is independent of the graph topology. However, empirical results have shown that, due to the nature of Laplacian smoothing in GNNs, graph convolutions have the over-smoothing tendency of mixing representations of adjacent nodes so that, when increasing the number of GNN layers, all nodes' representations will converge to a stationary point, making them unrelated to node features [61]. While it has been shown in Kipf and Welling [56] that DropOut alone is ineffectual in preventing over-fitting, partially due to over-smoothing, the combination of DropEdge, in which a set of edges are randomly removed from the graph, with DropOut has recently shown potential to alleviate these problems [87].

On the other hand, with the development of efficient posterior computation algorithms, there have been successes in learning with uncertainty by Bayesian extensions of traditional deep network architectures, including convolutional neural networks (CNNs). However, for GNNs, deriving their Bayesian extensions is more challenging due to their irregular neighborhood connection structures.

*Reprinted with permission from "Bayesian graph neural networks with adaptive connection sampling" by A. Hasanzadeh, E. Hajiramezanali, S. Boluki, M. Zhou, N. Duffield, K. Narayanan, and X. Qian. In International conference on machine learning, pp. 4094-4104. PMLR, 2020. Copyright 2020 by the authors.

In order to account for uncertainty in GNNs, Zhang et al. [118] present a Bayesian framework where the observed graph is viewed as a realization from a parametric family of random graphs. This allows joint inference of the graph and the GNN weights, leading to resilience to noise or adversarial attacks. Besides its prohibitive computational cost, the choice of the random graph model is important and can be inconsistent for different problems and datasets. Furthermore, the posterior inference in the current implementation only depends on the graph topology, but cannot consider node features.

In this work, we introduce a general stochastic regularization technique for GNNs by adaptive connection sampling—Graph DropConnect (GDC). We show that existing GNN regularization techniques such as DropOut [91], DropEdge [87], and node sampling [20] are special cases of GDC. GDC regularizes neighborhood aggregation in GNNs at each channel, separately. This prevents connected nodes in graph from having the same learned representations in GNN layers; hence better improvement without serious over-smoothing can be achieved. Furthermore, adaptively learning the connection sampling or drop rate in GDC enables better stochastic regularization given graph data for target graph analytic tasks. In fact, our ablation studies show that only learning the DropEdge rate, without any DropOut, already substantially improves the performance in semi-supervised node classification with GNNs. By probabilistic modeling of the *connection* drop rate, we propose a hierarchical beta-Bernoulli construction for Bayesian learnable GDC, and derive the solution with both continuous relaxation and direct optimization with Augment-REINFORCE-Merge (ARM) gradient estimates. With the naturally enabled UQ and regularization capability, our learnable GDC can help address both over-smoothing and UQ challenges to further push the frontier of GNN research.

We further prove that adaptive connection sampling of GDC at each channel can be considered as random aggregation and diffusion in GNNs, with a similar Bayesian approximation interpretation as in Bayesian DropOut for CNNs [32]. Specifically, Monte Carlo estimation of GNN outputs can be used to evaluate the predictive posterior uncertainty. An important corollary of this formulation is that any GNN with neighborhood sampling, such as GraphSAGE [43], could be considered as its

corresponding Bayesian approximation.

4.2 Background

4.2.1 Bayesian neural networks

Bayesian neural networks (BNNs) aim to capture model uncertainty of DNNs by placing prior distributions over the model parameters to enable posterior updates during DNN training. It has been shown that these Bayesian extensions of traditional DNNs can be robust to over-fitting and provide appropriate prediction uncertainty estimation [33, 14]. Often, the standard Gaussian prior distribution is placed over the weights. With random weights $\{\mathbf{W}^{(l)}\}_{l=1}^{L}$, the output prediction given an input $\mathbf{x}$ can be denoted by $\widehat{\mathbf{f}}\big(\mathbf{x}, \{\mathbf{W}^{(l)}\}_{l=1}^{L}\big)$, which is now a random variable in BNNs, enabling uncertainty quantification (UQ).

The key difficulty in using BNNs is that Bayesian inference is computationally intractable. There exist various methods that approximate BNN inference, such as Laplace approximation [66], sampling-based and stochastic variational inference [77, 86, 42, 25], Markov chain Monte Carlo (MCMC) [73], and stochastic gradient MCMC [64]. However, their computational cost is still much higher than the non-Bayesian methods, due to the increased model complexity and slow convergence [33].

4.2.2 DropOut as Bayesian approximation

Dropout is commonly used in training many deep learning models as a way to avoid over-fitting. Using dropout at test time enables UQ with Bayesian interpretation of the network outputs as Monte Carlo samples of its predictive distribution [33]. Various dropout methods have been proposed to multiply the output of each neuron by a random mask drawn from a desired distribution, such as Bernoulli [47, 91] and Gaussian [53, 91]. Bernoulli dropout and its extensions are the most commonly used in practice due to their ease of implementation and computational efficiency in existing deep architectures.

4.2.3 Over-smoothing and over-fitting in GNNs

It has been shown that graph convolution in graph convolutional neural networks (GCNs) [56] is simply a special form of Laplacian smoothing, which mixes the features of a node and its nearby neighbors. Such diffusion operations lead to similar learned representations when the corresponding nodes are close topologically with similar features, thus greatly improving node classification performance. However, it also brings potential concerns of *over-smoothing* [61]. If a GCN is deep with many convolutional layers, the learned representations may be over-smoothed and nodes with different topological and feature characteristics may become indistinguishable. More specifically, by repeatedly applying Laplacian smoothing many times, the node representations within each connected component of the graph will converge to the same values.

Moreover, GCNs, like other deep models, may suffer from *over-fitting* when we utilize an over-parameterized model to fit a distribution with limited training data, where the model we learn fits the training data very well but generalizes poorly to the testing data.

4.2.4 Stochastic regularization and reduction for GNNs

Quickly increasing model complexity and possible over-fitting and over-smoothing when modeling large graphs, as empirically observed in the GNN literature, have been conjectured for the main reason of limited performance from deep GNNs [56, 87]. Several stochastic regularization and reduction methods in GNNs have been proposed to improve the deep GNN performance. For example, *stochastic regularization techniques*, such as DropOut [91] and DropEdge [87], have been used to prevent over-fitting and over-smoothing in GNNs. Sampling-based *stochastic reduction by* random walk neighborhood sampling [43] and node sampling [20] has been deployed in GNNs to reduce the size of input data and thereafter model complexity. Next, we review each of these methods and show that they can be formulated in our proposed adaptive connection sampling framework.

Denote the output of the lth hidden layer in GNNs by $\mathbf{H}^{(l)} = [\mathbf{h}_0^{(l)}, \ldots, \mathbf{h}_n^{(l)}]^T \in \mathbb{R}^{n \times f_l}$ with n being the number of nodes and f_l being the number of output features at the lth layer. Assume

$\mathbf{H}^{(0)} = \mathbf{X} \in \mathbb{R}^{n \times f_0}$ is the input matrix of node attributes, where f_0 is the number of nodes features. Also, assume that $\mathbf{W}^{(l)} \in \mathbb{R}^{f_l \times f_{l+1}}$ and $\sigma(\cdot)$ are the GNN parameters at the lth layer and the corresponding activation function, respectively. Moreover, $\mathcal{N}(v)$ denotes the neighborhood of node v; $\hat{\mathcal{N}}(v) = \mathcal{N}(v) \cup \{v\}$; and $\mathfrak{N}(.)$ is the normalizing operator, i.e., $\mathfrak{N}(\mathbf{A}) = \mathbf{I}_N + \mathbf{D}^{-1/2}\,\mathbf{A}\,\mathbf{D}^{-1/2}$. Finally, $\odot$ represents the Hadamard product.

4.2.4.1 DropOut

In a GNN layer, DropOut [91] randomly removes output elements of its previous hidden layer $\mathbf{H}^{(l)}$ based on independent Bernoulli random draws with a constant success rate at each training iteration. This can be formulated as follows:

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathfrak{N}(\mathbf{A})(\mathbf{Z}^{(l)} \odot \mathbf{H}^{(l)})\,\mathbf{W}^{(l)}\right), \tag{4.1}$$

where $\mathbf{Z}^{(l)}$ is a random binary matrix, with the same dimensions as $\mathbf{H}^{(l)}$, whose elements are samples of $\mathrm{Bernoulli}(\pi)$. Despite its success in fully connected and convolutional neural networks, DropOut has shown to be ineffectual in GNNs for preventing over-fitting and over-smoothing.

4.2.4.2 DropEdge

DropEdge [87] randomly removes edges from the graph by drawing independent Bernoulli random variables (with a constant rate) at each iteration. More specifically, a GNN layer with DropEdge can be written as follows:

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathfrak{N}(\mathbf{A} \odot \mathbf{Z}^{(l)})\,\mathbf{H}^{(l)}\,\mathbf{W}^{(l)}\right), \tag{4.2}$$

Note that here, the random binary mask, i.e. $\mathbf{Z}^{(l)}$, has the same dimensions as $\mathbf{A}$. Its elements are random samples of $\mathrm{Bernoulli}(\pi)$ where their corresponding elements in $\mathbf{A}$ are non-zero and zero everywhere else. It has been shown that the combination of DropOut and DropEdge reaches the best performance in terms of mitigating overfitting in GNNs.

4.2.4.3 *Node sampling*

To reduce expensive computation in batch training of GNNs, due to the recursive expansion of neighborhoods across layers, Chen et al. [20] propose to relax the requirement of simultaneous availability of test data. Considering graph convolutions as integral transforms of embedding functions under probability measures allows for the use of Monte Carlo approaches to consistently estimate the integrals. This leads to an optimal node sampling strategy, FastGCN, which can be formulated as

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathfrak{N}(\mathbf{A})\,\mathrm{diag}(\mathbf{z}^{(l)})\mathbf{H}^{(l)}\,\mathbf{W}^{(l)}\right), \tag{4.3}$$

where $\mathbf{z}^{(l)}$ is a random vector whose elements are drawn from $\mathrm{Bernoulli}(\pi)$. This, indeed, is a special case of DropOut, as all of the output features for a node are either completely kept or dropped while DropOut randomly removes some of these related output elements associated with the node.

4.3 Graph DropConnect

We propose a general stochastic regularization technique for GNNs—Graph DropConnect (GDC)—by adaptive connection sampling, which can be interpreted as an approximation of Bayesian GNNs.

In GDC, we allow GNNs to draw different random masks for each channel and edge independently. More specifically, the operation of a GNN layer with GDC is defined as follows:

$$\mathbf{H}^{(l+1)}_{:,j} = \sigma\left(\sum_{i=1}^{f_l} \mathfrak{N}(\mathbf{A}\odot\mathbf{Z}^{(l,i,j)})\,\mathbf{H}^{(l)}_{:,i}\,W^{(l)}_{i,j}\right), \qquad \text{for} \quad j = 1,\,\ldots\,,f_{l+1} \tag{4.4}$$

where f_l and f_{l+1} are the number of features at layers l and $l+1$, respectively, and $\mathbf{Z}^{(l,i,j)}$ is a sparse random matrix (with the same sparsity as $\mathbf{A}$) whose non-zero elements are randomly drawn from $\mathrm{Bernoulli}(\pi_l)$. Note that π_l can be different for each layer for GDC instead of assuming the same constant drop rate for all layers in previous methods.

As shown in (4.1), (4.2), and (4.3), DropOut [91], DropEdge [87], and Node Sampling [20] have different sampling assumptions on channels, edges, or nodes, yet there is no clear evidence to favor one over the other in terms of consequent graph analytic performance. In the proposed GDC approach, there is a free parameter $\{\mathbf{Z}^{(l,i,j)} \in \{0,1\}^{n\times n}\}_{i=1}^{f_l}$ to adjust the binary mask for the edges, nodes and channels. Thus the proposed GDC model has one extra degree of freedom to incorporate flexible connection sampling.

The previous stochastic regularization techniques can be considered as special cases of GDC. To illustrate that, we assume $\mathbf{Z}^{(l,i,j)}$ are the same for all $j \in \{1, 2, \ldots, f_{l+1}\}$, thus we can omit the indices of the output elements at layer $l+1$ and rewrite (4.4) as

$$\mathbf{H}^{(l+1)} = \sigma\left(\sum_{i=1}^{f_l} \mathfrak{N}(\mathbf{A} \odot \mathbf{Z}^{(l,i)})\, \mathbf{H}_{:,i}^{(l)}\, \mathbf{W}_{i,:}^{(l)}\right) \tag{4.5}$$

Define $\mathbf{J}_n$ as a $n \times n$ all-one matrix. Let $\mathbf{Z}^{(l,DO)} \in \{0,1\}^{n\times f_l}$, $\mathbf{Z}^{(l,DE)} \in \{0,1\}^{n\times n}$, and $\mathrm{diag}(\mathbf{z}^{(l,NS)}) \in \{0,1\}^{n\times n}$ be the random binary matrices corresponding to the ones adopted in DropOut [91], DropEdge [87], and Node Sampling [20], respectively. The random mask $\{\mathbf{Z}^{(l,i)} \in \{0,1\}^{n\times n}\}_{i=1}^{f_l}$ in GDC become the same as those of the DropOut when $\mathbf{Z}^{(l,i)} = \mathbf{J}_n\, \mathrm{diag}(\mathbf{Z}_{:,i}^{(l,DO)})$, the same as those of DropEdge when $\{\mathbf{Z}^{(l,i)}\}_{i=1}^{f_l} = \mathbf{Z}^{(l,DE)}$, and the same as those of node sampling when $\{\mathbf{Z}^{(l,i)}\}_{i=1}^{f_l} = \mathbf{J}_n \mathrm{diag}(\mathbf{z}^{(l,NS)})$.

4.3.1 GDC as Bayesian approximation

In GDC, random masking is applied to the adjacency matrix of the graph to regularize the aggregation steps at each layer of GNNs. In existing Bayesian neural networks, the model parameters, i.e. $\mathbf{W}^{(l)}$, are considered random to enable Bayesian inference based on predictive posterior given training data [34, 14]. Here, we show that connection sampling in GDC can be transformed from the output feature space to the parameter space so that it can be considered as appropriate Bayesian extensions of GNNs.

First, we rewrite equation 4.5 to have a node-wise view of a GNN layer with GDC. More specifically,

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\frac{1}{c_v}\Big(\sum_{u\in\hat{\mathcal{N}}(v)} \mathbf{z}^{(l,v,u)} \odot \mathbf{h}_u^{(l)}\Big)\,\mathbf{W}^{(l)}\right), \tag{4.6}$$

where c_v is a constant derived from the degree of node v, and $\mathbf{z}^{(l,v,u)} \in \{0,1\}^{1\times f_l}$ is the mask row vector corresponding to connection between nodes v and u in three dimensional tensor $\mathcal{Z}^{(l)} = [\mathbf{Z}_1^{(l)}, \ldots, \mathbf{Z}_{f_l}^{(l)}]$. For brevity and without loss of generality, we ignore the constant c_v in the rest of this section. We can rewrite and reorganize equation 4.6 to transform the randomness from sampling to the parameter space as

$$\begin{aligned}\mathbf{h}_v^{(l+1)} &= \sigma\left(\Big(\sum_{u\in\hat{\mathcal{N}}(v)} \mathbf{h}_u^{(l)}\,\mathrm{diag}(\mathbf{z}^{(l,v,u)})\Big)\,\mathbf{W}^{(l)}\right)\\ &= \sigma\left(\sum_{u\in\hat{\mathcal{N}}(v)} \mathbf{h}_u^{(l)}\,\Big(\mathrm{diag}(\mathbf{z}^{(l,v,u)})\,\mathbf{W}^{(l)}\Big)\right).\end{aligned} \tag{4.7}$$

Define $\mathbf{W}^{(l,v,u)} := \mathbf{z}^{(l,v,u)}\,\mathbf{W}^{(l)}$. We have:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{u\in\hat{\mathcal{N}}(v)} \mathbf{h}_u^{(l)}\,\mathbf{W}^{(l,v,u)}\right). \tag{4.8}$$

$\mathbf{W}^{(l,v,u)}$, which pairs the corresponding weight parameter with the edge in the given graph. The operation with GDC in equation 4.8 can be interpreted as learning different weights for each of the message passing along edges $e = (u, v) \in \mathcal{E}$ where $\mathcal{E}$ is the union of edge set of the input graph and self-loops for all nodes.

Following the variational interpretation in Gal et al. [34], GDC can be seen as an approximating distribution $q_\theta(\boldsymbol{\omega})$ for the posterior $p(\boldsymbol{\omega} \mid \mathbf{A}, \mathbf{X})$ when considering a set of random weight matrices $\boldsymbol{\omega} = \{\boldsymbol{\omega}_e\}_{e=1}^{|\mathcal{E}|}$ in the Bayesian framework, where $\boldsymbol{\omega}_e = \{\mathbf{W}_e^{(l)}\}_{l=1}^{L}$ is the set of random weights for the eth edge, $|\mathcal{E}|$ is the number of edges in the input graph, and θ is the set of variational parameters. The Kullback–Leibler (KL) divergence $\mathrm{KL}(q_\theta(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$ is considered in training as a regularisation term, which ensures that the approximating $q_\theta(\boldsymbol{\omega})$ does not deviate too far from the prior distribution. To be able to evaluate the KL term analytically, the discrete quantised Gaussian can be adopted as the

prior distribution as in Gal et al. [34]. Further with the factorization $q_\theta(\boldsymbol{\omega})$ over L layers and $|\mathcal{E}|$ edges such that $q_\theta(\boldsymbol{\omega}) = \prod_l \prod_e q_{\theta_l}(\mathbf{W}_e^{(l)})$ and letting $q_{\theta_l}(\mathbf{W}_e^{(l)}) = \pi_l\, \delta(\mathbf{W}_e^{(l)} - 0) + (1-\pi_l)\, \delta(\mathbf{W}_e^{(l)} - \mathbf{M}^{(l)})$, where $\theta_l = \{\mathbf{M}^{(l)}, \pi_l\}$, the KL term can be written as $\sum_{l=1}^{L} \sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}(q_{\theta_l}(\mathbf{W}_e^{(l)}) \,||\, p(\mathbf{W}_e^{(l)}))$ and approximately

$$\mathrm{KL}(q_{\theta_l}(\mathbf{W}_e^{(l)}) \,||\, p(\mathbf{W}_e^{(l)})) \propto \frac{(1-\pi_l)}{2} \,||\, \mathbf{M}^{(l)}||^2 - \mathcal{H}(\pi_l),$$

where $\mathcal{H}(\pi_l)$ is the entropy of a Bernoulli random variable with the success rate π_l.

Since the entropy term does not depend on network weight parameters $\mathbf{M}^{(l)}$, it can be omitted when π_l is not optimized. But we learn π_l in GDC, thus the entropy term is important. Minimizing the KL divergence with respect to the drop rate π_l is equivalent to maximizing the entropy of a Bernoulli random variable with probability $1 - \pi_l$. This pushes the drop rate towards 0.5, which may not be desired in some cases where higher/lower drop rate probabilities are more appreciated.

4.3.2 Variational inference for GDC

Let's denote $\mathbf{Z}^{(l)} = \{\mathbf{z}_e^{(l)}\}_{e=1}^{|\mathcal{E}|}$ and $\boldsymbol{\omega}^{(l)} = \{\mathbf{W}_e^{(l)}\}_{e=1}^{|\mathcal{E}|}$. For inference of this approximating model with GDC, we assume a factorized variational distribution $q(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}) = q(\boldsymbol{\omega}^{(l)} \,|\, \mathbf{Z}^{(l)})\, q(\mathbf{Z}^{(l)})$. Let the prior distribution $p(\mathbf{W}_e^{(l)} \,|\, \mathbf{z}_e^{(l)} = 1)$ be a discrete quantised Gaussian, $p(\mathbf{W}_e^{(l)} \,|\, \mathbf{z}_e^{(l)} = 0)$ be $\delta(\mathbf{W}_e^{(l)} - 0)$, and $p(\boldsymbol{\omega}^{(l)} \,|\, \mathbf{Z}^{(l)}) = \prod_{e=1}^{\mathcal{E}} p(\mathbf{W}_e^{(l)} \,|\, \mathbf{z}_e^{(l)})$. Therefore, the KL term can be written as $\sum_{l=1}^{L} \mathrm{KL}(q(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}) \,||\, p(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}))$, with

$$\mathrm{KL}\left(q(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}) \,||\, p(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)})\right) \propto \frac{|\mathcal{E}|(1-\pi_l)}{2} ||\mathbf{M}^{(l)}||^2 + \sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}\left(q(\mathbf{z}_e^{(l)}) \,||\, p(\mathbf{z}_e^{(l)})\right).$$

The KL term consists of the common weight decay in the non-Bayesian GNNs with the additional KL term $\sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}(q(\mathbf{z}_e^{(l)}) \,||\, p(\mathbf{z}_e^{(l)}))$ acting as a regularization term for $\mathbf{z}_e^{(l)}$. In this GDC framework, the variational inference loss, for node classification for example, can be written as

$$\begin{aligned} \mathcal{L}(\{\mathbf{M}^{(l)}, \pi_l\}_{l=1}^{L}) = \quad & \mathbb{E}_{q(\{\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}\}_{l=1}^{L})}[\log P(Y_o | X, \{\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}\}_{l=1}^{L})] \\ & - \sum_{l=1}^{L} \mathrm{KL}(q(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)}) \,||\, p(\boldsymbol{\omega}^{(l)}, \mathbf{Z}^{(l)})), \end{aligned} \tag{4.9}$$

where Y_o denotes the collection of the available labels for the observed nodes. The optimization of equation 4.9 with respect to the weight matrices can be done by a Monte Carlo sample, i.e. sampling a random GDC mask and calculating the gradients with respect to $\{\mathbf{M}^{(l)}\}_{l=1}^{L}$ with stochastic gradient descent. It is easy to see that if $\{\pi_l\}_{l=1}^{L}$ are fixed, implementing our GDC is as simple as using common regularization terms on the neural network weights.

We aim to optimize the drop rates $\{\pi_l\}_{l=1}^{L}$ jointly with the weight matrices. This clearly provides more flexibility as all the parameters of the approximating posterior will be learned from the data instead of being fixed *a priori* or treated as hyper-parameters, often difficult to tune. However, the optimization of equation 4.9 with respect to the drop rates is challenging. Although the KL term is not a function of the random masks, the commonly adopted reparameterization techniques [86, 52] are not directly applicable here for computing the expectation in the first term since the drop masks are binary. Moreover, score-function gradient estimators, such as REINFORCE [105, 31], possess high variance. One potential solution is continuous relaxation of the drop masks. This approach has lower variance at the expense of introducing bias. Another solution is the direct optimization with respect to the discrete variables by the recently developed Augment-REINFORCE-Merge (ARM) method [114, 115], which has been used in BNNs [14] and information retrieval [26, 25]. In the next section, we will discuss in detail about our GDC formulation with more flexible beta-Bernoulli prior construction for adaptive connection sampling and how we solve the joint optimization problem for training GNNs with adaptive connection sampling.

4.4 Variational beta-Bernoulli GDC

The sampling or drop rate in GDC can be set as a constant hyperparameter as commonly done in other stochastic regularization techniques. In this work, we further enrich GDC with an adaptive sampling mechanism, where the drop rate is directly learned together with GNN parameters given graph data. In fact, in the Bayesian framework, such a hierarchical construct may increase the model expressiveness to further improve prediction and uncertainty estimation performance, as we will show empirically in Section 4.7.

Note that in this section, for brevity and simplicity we do the derivations for one feature

dimension only, i.e. $f_l = 1$. Extending to multi-dimensional features is straightforward as we assume the drop masks are independent across features. Therefore, we drop the feature index in our notations. Inspired by the beta-Bernoulli process [94], whose marginal representation is also known as the Indian Buffet Process (IBP) [35], we impose a beta-Bernoulli prior to the binary random masks as

$$\begin{aligned} a_e^{(l)} &= z_e^{(l)} a_e, \quad z_e^{(l)} \sim \text{Bernoulli}(\pi_l), \\ \pi_l &\sim \text{Beta}(c/L,\, c(L-1)/L), \end{aligned} \tag{4.10}$$

where a_e denotes an element of the adjacency matrix $\mathbf{A}$ corresponding to an edge e, and $\hat{a}_e^{(l)}$ an element of the matrix $\hat{\mathbf{A}}^{(l)} = \mathbf{A} \odot \mathbf{Z}^{(l)}$. Such a formulation is known to be capable of enforcing sparsity in random masks [120, 39], which has been shown to be necessary for regularizing deep GNNs as discussed in DropEdge [87].

With this hierarchical beta-Bernoulli GDC formulation, inference based on Gibbs sampling can be computationally demanding for large datasets, including graph data [46]. In the following, we derive efficient variational inference algorithm(s) for learnable GDC.

To perform variational inference for GDC random masks and the corresponding drop rate at each GNN layer together with weight parameters, we define the variational distribution as $q(\mathbf{Z}^{(l)}, \pi_l) = q(\mathbf{Z}^{(l)} \,|\, \pi_l)\, q(\pi_l)$. We define $q(\pi_l)$ to be Kumaraswamy distribution [58]; as an alternative to the beta prior factorized over lth layer

$$q(\pi_l; a_l, b_l) = a_l b_l \pi_l^{a_l - 1} (1 - \pi_l^{a_l})^{b_l - 1}, \tag{4.11}$$

where a_l and b_l are greater than zero. Knowing π_l the edges are independent, thus we can rewrite $q(\mathbf{Z}^{(l)} \,|\, \pi_l) = \prod_{e=1}^{|\mathcal{E}|} q(z_e^{(l)} \,|\, \pi_l)$. We further put a Bernoulli distribution with parameter π_l over

$q(\mathbf{z}_e^{(l)}|\pi_l)$. The KL divergence term can be written as

$$\mathrm{KL}\left(q(\mathbf{Z}^{(l)}, \pi_l) \,||\, p(\mathbf{Z}^{(l)}, \pi_l)\right) = \sum_{e=1}^{|\mathcal{E}|} \mathrm{KL}\left(q(z_e^{(l)} \,|\, \pi_l) \,||\, p(z_e^{(l)} \,|\, \pi_l)\right) + \mathrm{KL}\left(q(\pi_l) \,||\, p(\pi_l)\right).$$

While the first term is zero due to the identical distributions, the second term can be computed in closed-form as

$$\mathrm{KL}\left(q(\pi_l) \,||\, p(\pi_l)\right) = \frac{a_l - c/L}{a_l}\left(-\gamma - \Psi(b_l) - \frac{1}{b_l}\right) + \log\frac{a_l b_l}{c/L} - \frac{b_l - 1}{b_l}, \tag{4.12}$$

where γ is the Euler-Mascheroni constant and $\Psi(\cdot)$ is the digamma function.

The gradient of the KL term in equation 4.9 can easily be calculated with respect to the drop parameters. However, as mentioned in the previous section, due to the discrete nature of the random masks, we cannot directly apply reparameterization technique to calculate the gradient of the first term in equation 4.9 with respect to the drop rates (parameters). One way to address this issue is to replace the discrete variables with a continuous approximation. We impose a concrete distribution relaxation [48, 34] for the Bernoulli random variable $z_e^{(l)}$, leading to an efficient optimization by sampling from simple sigmoid distribution which has a convenient parametrization

$$\tilde{z}_e^{(l)} = \mathrm{sigmoid}\left(\frac{1}{t}\big(\log\big(\frac{\pi_l}{1-\pi_l}\big) + \log\big(\frac{u}{1-u}\big)\big)\right), \tag{4.13}$$

where $u \sim \mathrm{Unif}[0, 1]$ and t is temperature parameter of relaxation. We can then use stochastic gradient variational Bayes to optimize the variational parameters a_l and b_l.

Although this approach is simple, the relaxation introduces bias. Our other approach is to directly optimize the variational parameters using the original Bernoulli distribution in the formulation as in Boluki et al. [14]. We can calculate the gradient of the variational loss with respect to $\boldsymbol{\alpha} = \{\mathrm{logit}(1 - \pi_l)\}_{l=1}^{L}$ using ARM estimator , which is unbiased and has low variance, by performing two forward passes as

$$\nabla_{\boldsymbol{u}}\mathcal{L}(\alpha) = \mathbb{E}_{u\sim\prod_{l=1}^{L}\prod_{e=1}^{|\mathcal{E}|}\mathrm{Unif}[0,1](u_e^{(l)})}\Big[\big(\mathcal{L}(\{\mathbf{M}^{(l)}\}_{l=1}^{L}1_{[\boldsymbol{u}>\sigma(-\boldsymbol{\alpha})]}) - \mathcal{L}(\{\mathbf{M}^{(l)}\}_{l=1}^{L},1_{[\boldsymbol{u}<\sigma(\boldsymbol{\alpha})]})\big)\big(\boldsymbol{u}-\frac{1}{2}\big)\Big],$$

where $\mathcal{L}(\{\mathbf{M}^{(l)}\}_{l=1}^{L},1_{[\boldsymbol{u}<\sigma(\boldsymbol{\alpha})]})$ denotes the loss obtained by setting $\mathbf{Z}^{(l)} = 1_{[\boldsymbol{u}^{(l)}<\sigma(\boldsymbol{\alpha}_l)]} := \big(1_{[u_1^{(l)}<\sigma(\alpha_l)]},\dots,1_{[u_{|\mathcal{E}|}^{(l)}<\sigma(\alpha_l)]}\big)$ for $l=1,\dots,L$. The gradient with respect to $\{a_l,b_l\}_{l=1}^{L}$ can then be calculated by using the chain rule and the reparameterization for $\pi_l = (1-u^{\frac{1}{b_l}})^{\frac{1}{a_l}}, u\sim\mathrm{Unif}[0,1]$.

It is worth noting that although the beta-Bernoulli DropConnect with ARM is expected to provide better performance due to the more accurate gradient estimates, it has slightly higher computational complexity as it requires two forward passes.

4.5 Connection to random walk sampling

Various types of random walk have been used in graph representation learning literature to reduce the size of input graphs. In GNNs, specifically in GraphSAGE [43], random walk sampling has been deployed to reduce the model complexity for very large graphs. One can formulate a GNN layer with random walk sampling as follows:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\Big(\sum_{u\in\hat{\mathcal{N}}(v)}(z_{vu}^{(l)}\,|\,\mathbf{Z}^{(l-1)})\,\mathbf{h}_u^{(l)}\Big)\,\mathbf{W}^{(l)}\right). \tag{4.14}$$

Here, $\mathbf{Z}^{(l)}$ is the same as the one in DropEdge except that it is dependent on the masks from the previous layer. This is due to the fact that random walk samples for each node are connected subgraphs.

In this setup, we can decompose the variational distribution of the GDC formulation in an autoregressive way. Specifically, here we have $q(z_{vu}^{(l)}|\mathbf{Z}^{(l-1)}) = \mathrm{Bernoulli}(\pi_l)1_{\sum_{u\in\hat{\mathcal{N}}(v)}z_{vu}^{(l-1)}>0}$. With fixed Bernoulli parameters, we can calculate the gradients for the weight matrices with Monte Carlo integration. Learning Bernoulli parameters is challenging and does not allow direct application of ARM due to the autoregressive structure of the variational posterior. We leave sequential ARM for future study.

Corollary 1. *Any graph neural network with random walk sampling, such as GraphSAGE, is an*

approximation of a Bayesian graph neural network as long as outputs are calculated using Monte Carlo sampling.

4.6 Sampling complexity

The number of random samples needed for variational inference in GDC, equation 4.4, at each layer of a GNN is $|\mathcal{E}| \times f_l \times f_{l+1}$. This number would reduce to $|\mathcal{E}| \times f_l$ in the constrained version of GDC as shown in equation 4.5. These numbers, potentially, could be very high specially if the size of the graph or the number of filters are large, which could increase the space complexity and computation time. To circumvent this issue, we propose to draw a single sample for a block of features as oppose to drawing a new sample for every single feature. This would reduce the number of required samples to $|\mathcal{E}| \times \mathrm{nb}$ with nb being the number of blocks. In our experiments, we have one block in the first layer and two blocks in layers after that. In our experiments, we keep the order of features the same as the original input files, and divide them into nb groups with the equal number of features.

While in our GDC formulation, as shown in equation 4.4 and equation 4.5, the normalization $\mathfrak{N}(\cdot)$ is applied after masking, one can multiply the randomly drawn mask with the pre-computed normalized adjacency matrix. This relaxation reduces the computation time and has negligible effect on the performance based on our experiments. An extension to the GDC sampling strategy is asymmetric sampling where the mask matrix $\mathbf{Z}$ could be asymmetric. This would increase the number of samples by a factor of two; however it increases the model flexibility. In our experiments, we have used asymmetric masks and multiplied the mask with the normalized adjacency matrix.

4.7 Numerical results

We test the performance of our adaptive connection sampling framework, learnable GDC, on semi-supervised node classification using real-world citation graphs. In addition, we compare the uncertainty estimates of predictions by Monte Carlo beta-Bernoulli GDC and Monte Carlo Dropout. We also show the performance of GDC compared to existing methods in alleviating the issue of over-smoothing in GNNs. Furthermore, we investigate the effect of the number of blocks on the

Table 4.1: Semi-supervised node classification accuracy of GCNs with our adaptive connection sampling and baseline methods.

Method	**Cora**		**Citeseer**		**Cora-ML**	
	2 layers	4 layers	2 layers	4 layers	2 layers	4 layers
GCN-DO	80.98±0.48	78.24±2.40	70.44±0.39	64.38±0.90	83.45±0.73	81.51±1.01
GCN-DE	78.36±0.92	73.40±2.07	70.52±0.75	57.14±0.90	83.30±1.37	68.89±3.37
GCN-DO-DE	80.58±1.19	79.20±1.07	70.74±1.23	64.84±0.98	83.61±0.83	81.21±1.53
GCN-BBDE	**81.58**±0.49	**80.42**±0.25	**71.46**±0.55	**68.58**±0.88	**84.62**±1.70	**84.73**±0.52
GCN-BBGDC	**81.80**±0.99	**82.20**±0.92	**71.72**±0.48	**70.00**±0.36	**85.43**±0.70	**85.52**±0.83

performance of GDC. We have also investigated learning separate drop rates for every edge in the network, i.e. *local* GDC, which is included in the supplementary materials.

4.7.1 Semi-supervised node classification

4.7.1.1 Datasets and implementation details

We conducted extensive experiments for semi-supervised node classification with real-world citation datasets. We consider Cora, Citeseer and Cora-ML datasets, and preprocess and split them same as Kipf and Welling [56] and Bojchevski and Gunnemann [13]. We train beta-Bernoulli GDC (BBGDC) models for 2000 epochs with a learning rate of 0.005 and a validation set used for early stopping. All of the hidden layers in our implemented GCNs have 128 dimensional output features. We use 5×10^{-3}, 10^{-2}, and 10^{-3} as L2 regularization factor for Cora, Citeseer, and Cora-ML, respectively. For the GCNs with more than 2 layers, we use warm-up during the first 50 training epochs to gradually impose the beta-Bernoulli KL term in the objective function. The temperature in the concrete distribution is set to 0.67. For a fair comparison, the number of hidden units are the same in the baselines and their hyper-parameters are hand-tuned to achieve their best performance. Performance is reported by the average accuracy with standard deviation based on 5 runs on the test set. The dataset statistics as well as more implementation details are included in the supplementary materials.

Table 4.2: Accuracy of ARM optimization-based variants of our proposed method in semi-supervised node classification.

Method	**Cora** (4 layers)	**Citeseer** (4 layers)
GCN-BDE-ARM	79.95±0.79	67.90±0.15
GCN-BBDE-ARM	81.74±0.75	69.82±0.58
GCN-BBGDC-ARM	82.46±0.26	70.52±0.44

4.7.1.2 Discussion

Table 4.1 shows that BBGDC outperforms the state-of-the-art stochastic regularization techniques in terms of accuracy in all benchmark datasets. DO and DE in the table stand for DropOut and DropEdge, respectively. Comparing GCN-DO and GCN-DE, we can see that DropEdge alone is less effective than DropOut in overcoming over-smoothing and over-fitting in GCNs. The difference between accuracy of GCN-DO and GCN-DE is more substantial in deeper networks (5% in Cora, 7% in Citeseer, and 13% in Cora-ML), which further proves the limitations of DE. Among the baselines, combination of DO and DE shows the best performance allowing to have deeper models. However, this is not always true. For example in Citeseer, 4-layer GCN shows significant decrease in performance compared to 2-layer GCN.

To show the advantages of learning the drop rates as well as the effect of hierarchical beta-Bernoulli construction, we have also evaluated beta-Bernoulli DropEdge (BBDE) with the concrete approximation, in which the edge drop rate at *each layer* is learned using the same beta-Bernoulli hierarchical construction as GDC. We see that GCN with BBDE, without any DropOut, performs better than both GCNs with DE and DO-DE. By comparing GCN with BBDE and GCN with BBGDC, it is clear that the improvement is not only due to learnable sampling rate but also the increased flexibility of GDC compared to DropEdge. We note that GCN-BBGDC is the only method for which the accuracy improves by increasing the number of layers except in Citeseer.

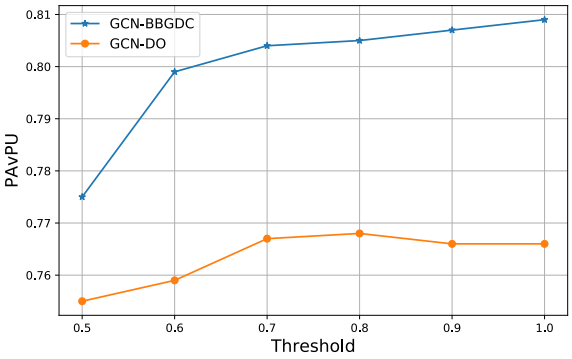

Figure 4.1: Comparison of uncertainty estimates in PAvPU by a 4-layer GCN-BBGDC with 128-dimensional hidden layers and a 4-layer GCN-DO 128-dimensional hidden layers on Cora.

4.7.1.3 Concrete relaxation versus ARM

To investigate the effect of direct optimization of the variational loss with respect to the drop parameters with ARM vs relaxation of the discrete random variables with concrete, we construct three ARM optimization-based variants of our method: Learnable Bernoulli DropEdge with ARM gradient estimator (BDE-ARM) where the edge drop rate of the Bernoulli mask at each layer is directly optimized; beta-Bernoulli DropEdge with ARM (BBDE-ARM); and beta-Bernoulli GDC with ARM (BBGDC-ARM). We evaluate these methods on the 4-layer GCN setups where significant performance improvement compared with the baselines has been achieved by BBDE and GDC with concrete relaxation. Comparing the performance of BBDE-ARM and BBGDC-ARM in Table 4.2 with the corresponding models with concrete relaxation, suggests further improvement when the drop parameters are directly optimized. Moreover, BDE-ARM, which optimizes the parameters of the Bernoulli drop rates, performs better than DO, DE, and DO-DE.

4.7.2 Uncertainty quantification

To evaluate the quality of uncertainty estimates obtained by our model, we use the Patch Accuracy vs Patch Uncertainty (PAvPU) metric introduced in [72]. PAvPU combines $p(\text{accurate}|\text{certain})$, i.e. the probability that the model is accurate on its output given that it is confident on the same, $p(\text{certain}|\text{inaccurate})$, i.e. the probability that the model is uncertain about its output given that

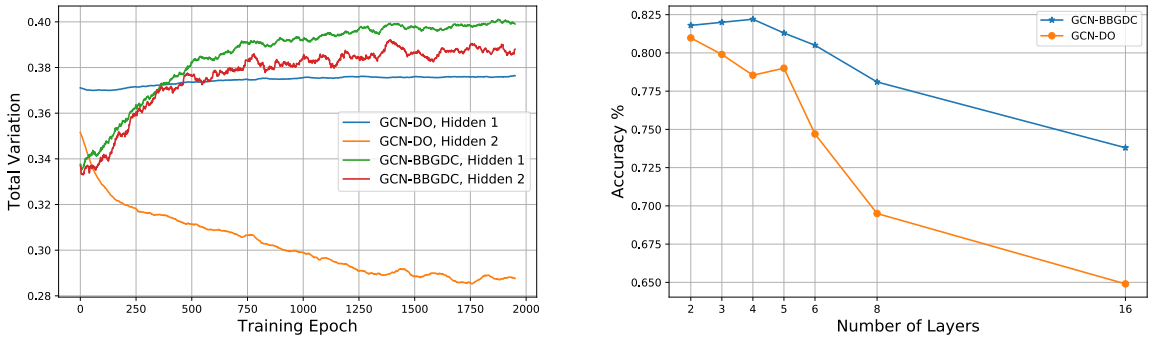

Figure 4.2: From left to right: a) Total variation of the hidden layer outputs during training in a 4-layer GCN-BBGDC with 128-dimensional hidden layers and a 4-layer GCN-DO 128-dimensional hidden layers on Cora; b) Comparison of node classification accuracy for GCNs with a different number of hidden layers using different stochastic regularization methods. All of the hidden layers are 128 dimensional.

it has made a mistake in its prediction, into a single metric. More specifically, it is defined as $\text{PAvPU} = (n_{ac} + n_{iu})/(n_{ac} + n_{au} + n_{ic} + n_{iu})$, where n_{ac} is the number of accurate and certain predictions, n_{au} is the number of accurate and uncertain predictions, n_{ic} is the number of inaccurate and certain predictions, and n_{iu} is the number of inaccurate and uncertain predictions. Higher PAvPU means that certain predictions are accurate and inaccurate predictions are uncertain.

We here demonstrate the results for uncertainty estimates for a 4-layer GCN-DO and a 4-layer GCN-BBGDC with random initialization for semi-supervised node classification on Cora. We have evaluated PAvPU using 20 Monte Carlo samples for the test set where we use predictive entropy as the uncertainty metric. The results are shown in Figure 4.1. It can be seen that our proposed model consistently outperforms GCN-DO on every uncertainty threshold ranging from 0.5 to 1 of the maximum predictive uncertainty. While Figure 4.1 depicts the results based on one random initialization, other initializations show the same trend.

4.7.3 Over-smoothing and over-fitting

To check how GDC helps alleviate over-smoothing in GCNs, we have tracked the total variation (TV) of the outputs of hidden layers during training. TV is a metric used in the graph signal processing literature to measure the smoothness of a signal defined over nodes of a graph [22]. More

Table 4.3: Accuracy of 128-dimensional 4-layer GCN-BBGDC with different number of blocks on Cora in semi-supervised node classification.

Method	2 blocks	16 blocks	32 blocks
GCN-BBGDC	82.2	83.0	83.3

specifically, given a graph with the adjacency matrix $\mathbf{A}$ and a signal $\mathbf{x}$ defined over its nodes, TV is defined as $\mathrm{TV}(\mathbf{x}) = \|\mathbf{x} - (1/|\lambda_{max}|)\mathbf{A}\,\mathbf{x}\|_2^2$, where λ_{max} denotes the eigenvalue of A with largest magnitude. Lower TV shows that the signal on adjacent nodes are closer to each other, indicating possible over-smoothing.

We have compared the TV trajectories of the hidden layer outputs in a 4-layer GCN-BBGDC and a 4-layer GCN-DO normalized by their Frobenius norm, depicted in Figure 4.2(a). It can be seen that, in GCN-DO, while the TV of the first layer is slightly increasing at each training epoch, the TV of the second hidden layer decreases during training. This, indeed, contributed to the poor performance of GCN-DO. On the contrary, the TVs of both first and second layers in GCN-BBGDC is increasing during training. Not only this robustness is due to the dropping connections in GDC framework, but also is related to its learnable drop rates.

With such promising results showing less over-smoothing with BBGDC, we further investigate how our proposed method works in deeper networks. We have checked the accuracy of GCN-BBGDC with a various number of 128-dimensional hidden layers ranging from 2 to 16. The results are shown in Figure 4.2(b). The performance improves up to the GCN with 4 hidden layers and decreases after that. It is important to note that even though the performance drops by adding the 5-th layer, the degree to which it decreases is far less than competing methods. For example, the node classification accuracy with GCN-DO quickly drops to 69.50% and 64.5% with 8 and 16 layers. In addition, we should mention that the performance of GCN-DO only improves from two to three layers. This, indeed, proves GDC is a better stochastic regularization framework for GNNs in alleviating over-fitting and over-smoothing, enabling possible directions to develop deeper GNNs.

4.7.4 Effect of number of blocks

In GDC for every pair of input and output features, a separate mask for the adjacency matrix should be drawn. However, as we discussed in Section 4.6, this demands large memory space. We circumvented this problem by drawing a single mask for a block of features. While we used only two blocks in our experiments presented so far, we here investigate the effect of the number of blocks on the node classification accuracy. The performance of 128-dimensional 4-layer GCN-BBGDC with 2, 16, and 32 blocks are shown in Table 4.3. As can be seen, the accuracy improves as the number of blocks increases. This is due to the fact that increasing the number of blocks increases the flexibility of GDC. The choice of the number of blocks is a factor to consider for the trade off between the performance and memory usage as well as computational complexity.

5. BAYESIAN MULTI-MODAL RELATIONAL

5.1 Introduction

Multi-modal learning tries to fully leverage the information from multiple sources (i.e. different types of omics data in molecular biology) and represents them in a shared embedding space, which is beneficial for many downstream tasks with a limited number of training samples. In biomedical applications, the shared embedding space also enables better understanding of the underlying biological mechanisms by discovering interactions between different types of molecules, which is our focus in this chapter.

Existing multi-omics data integration methods are limited in their applicability. First, most of them attempt to derive low-dimensional embeddings of the input samples and are not designed to infer a multi-partite graph that encodes the interactions across views. In unsupervised setting, matrix factorization based methods, such as Bayesian Canonical Correlation Analysis (BCCA) [57] and Multi-Omics Factor Analysis (MOFA) [5] can achieve the similar goal of cross-view relational learning but often through two-step procedures, in which the factor loading parameters are used for downstream interaction analyses across views. Second, a very recent relational inference for multi-view data integration, BayRel [41], is built on three strict assumptions, which may limit its practical application, including in multi-omics data integration: 1) A graph of dependency between features of each view is available; 2) The input dataset is complete on all views with no missing samples; 3) The samples in different views are well-paired. While the first limitation might be solved by learning a graph using an *ad-hoc* technique, the last two issues are common in many multi-omics data integration problems. Integrated samples commonly have one or more view with various missing patterns. This is mostly due to limitations of experimental designs or compositions from different data platforms. In addition, data might be collected in different laboratories or the sample IDs may not be available due to patient identification or security concerns, leading to unpaired datasets. Apart from these, we might not have access to *a priori* graph structure data in

some view(s) as the nature of data might not be structured, or we only have incomplete or very noisy prior knowledge. For such multi-omics data, leaving out such a view may lose some complementary information while enforcing graph structures may cause degraded performance.

In this work, we propose a new **M**ulti-m**o**dal **Re**lational **L**earning method, MoReL, based on fused Gromov-Wasserstein (FGW) regularization, mitigating the dependency of multi-view learning on the two aforementioned assumptions. The proposed method contains four major contributions: 1) MoReL provides a new Bayesian multi-omics relational learning framework with efficient variational inference and is able to exploit non-linear transformations of data by leveraging deep learning models for either unstructured or graph-structured data; 2) MoReL learns a multi-partite graph across different features from multiple views using a FGW-based decoder, facilitating meaningful biological knowledge discovery from integrative multi-omics data analysis while accounting for arbitrarily permutation and/or transformation caused by processing features with different deep functions across the views; 3) MoReL can flexibly integrate both structured and unstructured heterogeneous views in one framework, in which only confident constraints need to be imposed to improve the model performance; 4) MoReL is able to integrate multiple views with unpaired samples and/or arbitrary sample-missing patterns.

5.2 Related works

Optimal transport. There have been extensive efforts to utilize Gromov-Wasserstein (GW) discrepancy to solve the alignment problems in shape and object matching [67, 68]. A similar attempt has been made recently to investigate its potential for more diverse applications, such as aligning vocabulary sets between different languages [4], and graph matching [23, 95, 108]. Peyré et al. [82] have proposed a fast Sinkhorn projection-based algorithm [24] to compute the entropy-regularized GW distance. Following this direction, Xu et al. [108] have replaced the entropy regularizer with a Bregman proximal term. To further reduce the computational complexity, the recursive GW distance [107] and the sliced GW distance [98] have been proposed. In Bunne et al. [16], a pair of generative models are learned for incomparable spaces by defining an adversarial objective function based on the GW discrepancy. In addition, it imposes an orthogonal assumption

on the transformation between the sample and its latent space. However, it can not incorporate the graph structured data. Similar to our model in this work, Vayer et al. [97] and Xu et al. [109] have proposed to impose the fused GW regularization in their objective functions by combining GW and Wasserstein discrepancies.

Graph CCA (gCCA). In order to utilize *a priori* known information about geometry of the samples, gCCA methods [19, 18] have been proposed to construct a dependency graph between *samples* and directly impose it into a regularizer. Similar to classical CCA, gCCA learns an unstructured shared latent representation. Unlike our MoReL, though, they can neither take advantage of the dependency graph between *features*, nor explicitly model relational dependency between *features* across views. Therefore, they rely on *ad-hoc* post-processing procedures as a second step to infer inter-relations.

Graph representation learning. Graph neural network architectures have been shown to be effective for link prediction [43, 55, 46] as well as matrix completion for recommender systems [10, 70, 50, 63]. The first group of models is dealing with a single graph and is not able to deal with heterogeneous graphs, with multiple types of nodes and edges, and node attributes [116]. The second group utilizes the known item-item and user-user relationships and their attributes to complete the user-item rating matrix. However, they rely on two strict assumptions: 1) The inter-relation matrix is partially observed; and 2) Both views have structured information. The proposed MoReL achieves robust multi-view learning without these assumptions, making it more practical in multi-omics data integration.

5.3 Preliminaries

5.3.1 Wasserstein distance

Wasserstein distance (WD) quantifies the geometric discrepancy between two probability distributions by measuring the minimal amount of "work" needed to move all the mass contained in one distribution onto the other [89]. More specifically, given two probability measures $\mathbf{\Lambda} \in \mathcal{P}(\mathbb{X})$ and $\mathbf{\Delta} \in \mathcal{P}(\mathbb{Y})$, and a transportation cost $c : \mathbb{X} \times \mathbb{Y} \to \mathbb{R}_+$, WD is the solution to the following

optimization problem:

$$\inf_{\pi \in \Pi(\mathbb{X} \times \mathbb{Y})} \mathbb{E}_{(\mathbf{x},\, \mathbf{y}) \sim \pi}[c(\mathbf{x},\, \mathbf{y})] = \inf_{\pi \in \Pi(\mathbb{X} \times \mathbb{Y})} \int c(\mathbf{x},\, \mathbf{y})\, d\pi(\mathbf{x},\, \mathbf{y}),$$

where π is the transport map and $\Pi(\mathbb{X} \times \mathbb{Y}) \coloneqq \{\pi \in \mathcal{P}(\mathbb{X} \times \mathbb{Y}) \mid \int \pi(\mathbf{x}, \mathbf{y})\, d\mathbf{y} = \mathbf{\Lambda}(\mathbf{x}),\, \int \pi(\mathbf{x}, \mathbf{y})\, d\mathbf{x} = \mathbf{\Delta}(\mathbf{y})\}$ is the set of all admissible couplings. Assuming that the probability distributions are discrete, with probability mass functions $\sum_{i=1}^{n} a_i \delta_{\mathbf{x}_i}$ and $\sum_{j=1}^{m} b_j \delta_{\mathbf{y}_j}$, WD optimization could be simplified as follows:

$$\mathcal{D}_{\mathrm{W}}(\mathbf{\Lambda}, \mathbf{\Delta}) = \min_{\mathbf{T} \in \Pi(a,\, b)} \sum_{i=1}^{n} \sum_{j=1}^{m} T_{i,j}\, c(\mathbf{x}_i, \mathbf{y}_j),$$

where $T_{i,j}$ is an element of the transport matrix $\mathbf{T}$ whose row-wise and column-wise sums equal to $[a_i]_{i=1}^{n}$ and $[b_j]_{j=1}^{m}$, respectively.

5.3.2 Gromov-Wasserstein distance

Gromov-Wasserstein distance (GWD) has been proposed as a natural extension of WD when a meaningful transportation cost between the distributions cannot be defined. For example, when two distributions are defined in Euclidean spaces with different dimensions or more generally when $\mathbb{X}$ and $\mathbb{Y}$ are unaligned, i.e. when their features are not in correspondence [98]. Instead of measuring inter-domain distances, GWD measures the distance between pairs of samples in one domain and compares it to those in the other domain. More specifically, given two probability measures $\mathbf{\Lambda} \in \mathcal{P}(\mathbb{X})$ and $\mathbf{\Delta} \in \mathcal{P}(\mathbb{Y})$, as well as two domain-specific transportation costs $c^{(\mathbb{X})} : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$ and $c^{(\mathbb{Y})} : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}_+$, GWD is the solution to the following optimization problem:

$$\inf_{\pi \in \Pi(\mathbb{X} \times \mathbb{Y})} \mathbb{E}_{(\mathbf{x},\, \mathbf{y}) \sim \pi, (\mathbf{x}',\, \mathbf{y}') \sim \pi}[L(\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')] = \inf_{\pi \in \Pi(\mathbb{X} \times \mathbb{Y})} \int \int L(\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')\, d\pi(\mathbf{x},\, \mathbf{y})\, d\pi(\mathbf{x}',\, \mathbf{y}'),$$

where $L(\mathbf{x},\, \mathbf{x}',\, \mathbf{y},\, \mathbf{y}') = \parallel c^{(\mathbb{X})}(\mathbf{x},\, \mathbf{x}') - c^{(\mathbb{Y})}(\mathbf{y},\, \mathbf{y}') \parallel$, π is the transport map, and $\Pi(\mathbb{X} \times \mathbb{Y}) \coloneqq \{\pi \in \mathcal{P}(\mathbb{X} \times \mathbb{Y}) \mid \int \pi(\mathbf{x}, \mathbf{y})\, d\mathbf{y} = \mathbf{\Lambda}(\mathbf{x}),\, \int \pi(\mathbf{x}, \mathbf{y})\, d\mathbf{x} = \mathbf{\Delta}(\mathbf{y})\}$ is the set of all admissible couplings. Likewise, this can be derived for discrete distributions with probability mass functions $\sum_{i=1}^{n} a_i \delta_{\mathbf{x}_i}$

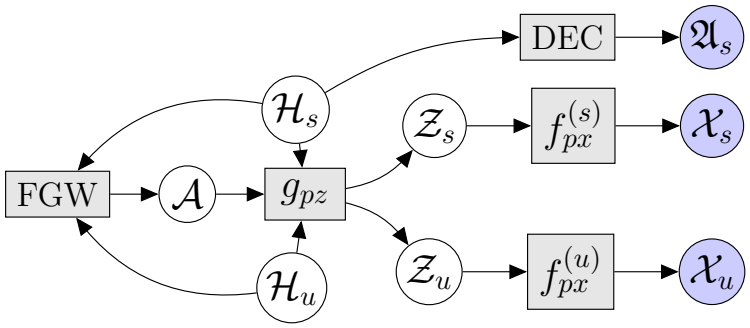

Figure 5.1: Graphical illustration of MoReL's generative flow with structured and unstructured views. DEC stand for decoder. The rest of variables and abbreviations are defined in the manuscript.

and $\sum_{j=1}^{m} b_j \delta_{\mathbf{y}_j}$, as follows:

$$\mathcal{D}_{\mathrm{GW}}(\mathbf{\Lambda}, \mathbf{\Delta}) = \min_{\mathbf{T} \in \Pi(a,\, b)} \sum_{i,i'=1}^{n} \sum_{j,j'=1}^{m} T_{i,j}\, T_{i',j'}\, L(\mathbf{x}_i, \mathbf{x}_{i'}, \mathbf{y}_j, \mathbf{y}_{j'}), \tag{5.1}$$

where $T_{i,j}$ is an element of transport matrix $\mathbf{T}$ whose row-wise and column-wise sums equal to $[a_i]_{i=1}^{n}$ and $[b_j]_{j=1}^{m}$, respectively.

5.4 Method

5.4.1 Problem formulation and notations

We propose a novel hierarchical generative model for multi-omics data integration that incorporates view-specific structure information when it is available. Given observations from structured and unstructured views, our model, Multi-modal Relational Learning (MoReL), aims to infer the *inter-relations* among entities, i.e. features, across all views. More specifically, assume that multiple views, $\mathcal{V}$, of data are given. Without loss of generality, we assume that the structure information, provided as a graph, is available for some of the views $\mathcal{V}_s \subset \mathcal{V}$, and the remaining views $\mathcal{V}_u = \mathcal{V} \setminus \mathcal{V}_s$ are unstructured. We note that every structure could be represented as a graph. For example, image and sequential data could be represented over grid and directed path graphs, respectively.

We represent the set of graphs for structured views by $\mathfrak{G}_s = \{\mathcal{G}^{(v)}\}_{v \in \mathcal{V}_s}$ and their adjacency matrices by $\mathfrak{A}_s = \{\mathbf{A}^{(v)}\}_{v \in \mathcal{V}_s}$. We also define $\mathcal{X}_s = \{\mathbf{X}^{(v)}\}_{v \in \mathcal{V}_s}$ as the set of node attributes

for structured views, and $\mathcal{X}_u = \{\mathbf{X}^{(v)}\}_{v\in\mathcal{V}_u}$ as the set of data for unstructured views. Moreover, N_v denotes the number of nodes in structured views and number of features for unstructured views. MoReL infers the interactions among the nodes in $\mathfrak{G}_s$ and features in $\mathcal{X}_u$. We represent these inter-relations by a multi-partite graph with $\sum_{v\in\mathcal{V}} N_v$ nodes and a multi-adjacency tensor $\mathcal{A} = \{\mathbf{A}^{(vv')}\}_{v,v'\in\mathcal{V},v\neq v'}$, where $\mathbf{A}^{(vv')}$ is the $N_v \times N_{v'}$ bi-adjacency matrix between views v and v'.

5.4.2 MoReL generative model

We define a hierarchical Bayesian model for MoReL with three sets of latent variables: 1) $\mathcal{H} = \mathcal{H}_s \cup \mathcal{H}_u = \{\mathbf{H}^{(v)}\}_{v\in\mathcal{V}_s\cup\mathcal{V}_u}$, which captures the (hidden) structural information; 2) $\mathcal{A}$, which encodes the interaction among features across views; and 3) $\mathcal{Z} = \mathcal{Z}_s \cup \mathcal{Z}_u = \{\mathbf{Z}^{(v)}\}_{v\in\mathcal{V}_s\cup\mathcal{V}_u}$, which summarizes the feature/attribute specific information. In our model, the joint probability of observations and latent variables factorizes as follows:

$$
\begin{aligned}
&p_\theta(\mathcal{X}_u,\, \mathcal{X}_s,\, \mathfrak{A}_s,\, \mathcal{H},\, \mathcal{A},\, \mathcal{Z}) = \\
&\qquad p_{\theta_x}(\mathcal{X}_u \mid \mathcal{Z}_u)\, p_{\theta_x}(\mathcal{X}_s \mid \mathcal{Z}_s)\, p_{\theta_g}(\mathfrak{A}_s \mid \mathcal{H}_s)\, p_{\theta_z}(\mathcal{Z} \mid \mathcal{H},\, \mathcal{A})\, p_{\theta_a}(\mathcal{A} \mid \mathcal{H})\, p(\mathcal{H}).
\end{aligned}
\tag{5.2}
$$

Figure 5.1 depicts the generative model of MoReL with structured and unstructured views. In the following subsections, we define different parts of the generative and inference model.

5.4.2.1 Optimal transport for multi-partite graph decoder

In this subsection, we define the generative distribution of the multi-adjacency tensor, $\mathcal{A}$. We note that inferring $\mathcal{A}$ is the main goal of our model. Given the structural latent variables $\mathcal{H}$, we introduce a fused Gromov-Wasserstein (FGW) distance based mapping to generate $\mathcal{A}$. FGW refers to distance metrics defined by combining WD and GWD, which has been proposed to compare structured distributions [95, 21]. Considering graphs with node attributes as structured distributions, WD compares node distributions in two graphs (i.e, node similarity), GWD measures the distance between pairs of nodes in one graph and compares it to those in the other (i.e., edge/path similarity).

FGW distance. Given two structured probability distributions, $\mathbf{\Lambda} \in \mathcal{P}(\mathbb{X})$ and $\mathbf{\Delta} \in \mathcal{P}(\mathbb{Y})$,

FGW is defined as follows:

$$\begin{aligned}\mathcal{D}_{\mathrm{FGW}}(\mathbf{\Lambda},\,\mathbf{\Delta}) &= \alpha\,\mathcal{D}_{\mathrm{W}}(\mathbf{\Lambda},\,\mathbf{\Delta}) + \beta\,\mathcal{D}_{\mathrm{GW}}(\mathbf{\Lambda},\,\mathbf{\Delta}) \\ &= \alpha \inf_{\pi_w \in \Pi(\mathbb{X}\times\mathbb{Y})} \mathbb{E}_{(\mathbf{x},\,\mathbf{y})\sim\pi_w}[c^{(\mathbb{X}\mathbb{Y})}(\mathbf{x},\,\mathbf{y})] \\ &\quad + \beta \inf_{\pi_{gw} \in \Pi(\mathbb{X}\times\mathbb{Y})} \mathbb{E}_{(\mathbf{x},\,\mathbf{y}),(\mathbf{x}',\,\mathbf{y}')\sim\pi_{gw}}[\|\, c^{(\mathbb{X})}(\mathbf{x},\,\mathbf{x}') - c^{(\mathbb{Y})}(\mathbf{y},\,\mathbf{y}')\,\|],\end{aligned} \tag{5.3}$$

where $\alpha, \beta \in [0,\,1]$ are scalar hyper-parameters, $\Pi(\mathbb{X}\times\mathbb{Y})$ is the set of all admissible couplings between $\mathbf{\Lambda}$ and $\mathbf{\Delta}$, and $c^{(\mathbb{X}\mathbb{Y})}$, $c^{(\mathbb{X})}$, and $c^{(\mathbb{Y})}$ are corresponding transportation cost functions. $\mathcal{D}_{\mathrm{FGW}}$ can be further simplified by choosing π_w to be equal to π_{gw} [21].

Relational learning via FGW. We are interested in aligning the nodes/features in every pair of views, i.e. (v, v'). Hence, we will have a FGW distance based decoder for every pair of views, in which each view independently belongs to either structured or unstructured views, i.e. $(v, v') \in \mathcal{V}$. To that end, we first define the transportation cost functions $c^{(vv')}$ and $c^{(v)}$, and then approximate $\mathcal{D}_{\mathrm{FGW}}$. We define the (inter-)cost function for the first term of FGW, i.e. $\mathcal{D}_{\mathrm{W}}$, as follows:

$$c^{(vv')}(\mathbf{H}^{(v)}_{i,:}\,,\,\mathbf{H}^{(v')}_{j,:}) = 1 - \sigma\left(\mathbf{H}^{(v)}_{i,:}\,(\mathbf{H}^{(v')}_{j,:})^T\right); \qquad\qquad v, v' \in \mathcal{V}, \tag{5.4}$$

where σ denotes the sigmoid function, and $\mathbf{H}^{(v)}_{i,:}$ represents the structural latent variable of node/feature i in view v. To calculate the $\mathcal{D}_{\mathrm{GW}}$, we define two different transportation costs based on the nature of the inputs. For the structured views, we define the cost function as a combination of the shortest path distance from graph and the distance between structural latent variables. More specifically, given the normalized shortest path distance matrix between every pair of nodes in the input graph $\mathbf{D}^{(v)}$:

$$c^{(v)}(\mathbf{H}^{(v)}_{i,:}\,,\,\mathbf{H}^{(v)}_{j,:}) = \mathbf{D}^{(v)} \odot \left(1 - \sigma\left(\mathbf{H}^{(v)}_{i,:}\,(\mathbf{H}^{(v)}_{j,:})^T\right)\right); \qquad \text{for } \; v \in \mathcal{V}_s,$$

where $\odot$ denotes the Hadamard product. This construction ensures both graph and attributes information are incorporated in the distance function. For unstructured views, we define the cost

function between two features as follows:

$$c^{(v)}(\mathbf{H}^{(v)}_{i,:}\,,\,\mathbf{H}^{(v)}_{j,:}) = 1 - \sigma\left(\mathbf{H}^{(v)}_{i,:}\,(\mathbf{H}^{(v)}_{j,:})^T\right); \qquad \text{for } \; v \in \mathcal{V}_u.$$

Noting the definitions of WD and GWD in Section 5.3, we rewrite $\mathcal{D}_{\mathrm{FGW}}$ between two views of data with shared transport matrix as follows:

$$\begin{aligned}\mathcal{D}_{\mathrm{FGW}}\Big(p(\mathbf{H}^{(v)}),\, p(\mathbf{H}^{(v')})\Big) = \\ \sum_{i=1}^{N_v}\sum_{j=1}^{N_{v'}} \min_{\mathbf{T}^{(vv')}_{gw}\in\Pi} \sum_{\mathbf{H}^{(v)}_{i,:},\mathbf{H}^{(v')}_{j,:},\mathbf{H}^{(v)\prime}_{i,:},\mathbf{H}^{(v')\prime}_{j,:}} \big[\alpha\, c^{(vv')}(\mathbf{H}^{(v)}_{i,:},\, \mathbf{H}^{(v')}_{j,:}) + \\ \beta \parallel c^{(v)}(\mathbf{H}^{(v)}_{i,:},\, \mathbf{H}^{(v)\prime}_{i,:}) - c^{(v')}(\mathbf{H}^{(v')}_{j,:}, \mathbf{H}^{(v')\prime}_{j,:}) \parallel \big].\end{aligned} \tag{5.5}$$

To approximate the FGW distance, we first deploy GW algorithm in equation (5.1) to obtain $\mathbf{T}^{(vv')}_{gw}$ and $\mathcal{D}_{\mathrm{GW}}$, and then utilize $\mathbf{T}^{(vv')}_{gw}$ along with the defined transportation cost $c^{(vv')}$ to calculate Wasserstein distance term in $\mathcal{D}_{\mathrm{FGW}}$ [21]. The pseudo-code in Algorithm 1 provides the details of the FGW distance calculation procedure. ρ in Algorithm 1 is a hyper-parameter. Please note that we use the same Sinkhorn solver as in [21] and [4].

We further can generate $\mathcal{A}$ for every pair of views based on $\mathbf{T}^{(vv')}_{gw}$ as follows:

$$p(\mathcal{A}\,|\,\mathcal{H}) = \prod_{\substack{v,v'\in\mathcal{V}\\ v\neq v'}} p(\mathbf{A}^{(vv')}\,|\,\mathbf{H}^{(v)},\mathbf{H}^{(v')}) = \prod_{\substack{v,v'\in\mathcal{V}\\ v\neq v'}} \mathrm{Ber}\left(\mathbf{A}^{(vv')}\,|\,\gamma\,\mathbf{T}^{(vv')}_{gw}/\max(\mathbf{T}^{(vv')}_{gw})\right), \tag{5.6}$$

where $\gamma \in [0,1]$ is a normalizing hyper-parameter, and Ber is short for Bernoulli. We note that the sum of the elements in each of the transport matrices $\mathbf{T}^{(vv')}_{gw}$ equals to one. Hence each of its elements has a small value. Therefore, we normalize the transport matrices (as $\gamma\,\mathbf{T}^{(vv')}_{gw}/\max(\mathbf{T}^{(vv')}_{gw})$) to avoid very sparse and trivial solutions. To use the reparametrization trick during training, we sample from concrete relaxation of Bernoulli [34]. We emphasize that our proposed FGW-based decoder is the key in aligning features/nodes across structured and unstructured views via accurate and efficient *distribution matching* scheme.

Algorithm 1: Computing fused Gromov-Wasserstein distance.

Input: $\boldsymbol{C}_{n\times n}^{(v)}$, $\boldsymbol{C}_{m\times m}^{(v')}$, $\boldsymbol{C}_{n\times m}^{(vv')}$, ρ

Definitions: $\odot$ = Hadamard product, $\langle\cdot,\cdot\rangle$ = Frobenius dot-product

// Cross-view similarity:

$\hat{\boldsymbol{C}}^{(vv')} = (\boldsymbol{C}^{(v)})^2\mathbf{1_n}\mathbf{1_m}^\top + \mathbf{1_n}\mathbf{1_m}^\top((\boldsymbol{C}^{(v')})^2)^\top$

// Initializing variables:

$\boldsymbol{T} = \mathbf{1_n}\mathbf{1_m}^\top, \quad \boldsymbol{\sigma} = \frac{1}{m}\mathbf{1_m}, \quad \boldsymbol{B}_{i,j} = \exp(\hat{\boldsymbol{C}}_{i,j}^{(vv')})/\rho$

for $t_1 = 1, 2, \ldots$ **do**

  $\mathcal{L} = \hat{\boldsymbol{C}}^{(vv')} - 2\boldsymbol{C}^{(v)}\boldsymbol{T}(\boldsymbol{C}^{(v')})^\top$

  for $t_2 = 1, 2, \ldots$ **do**

    $\boldsymbol{M} = \boldsymbol{B} \odot \boldsymbol{T}$

    for $t_3 = 1, 2, \ldots$ **do**

      $\boldsymbol{\delta} = \frac{1}{n\boldsymbol{M}\boldsymbol{\sigma}}, \quad \boldsymbol{\sigma} = \frac{1}{n\boldsymbol{M}^\top\boldsymbol{\delta}}$

    $\boldsymbol{T} = \text{diag}(\boldsymbol{\delta})\,\boldsymbol{M}\,\text{diag}(\boldsymbol{\sigma})$

$\mathcal{D}_W = \langle(\boldsymbol{C}^{(vv')})^\top, \boldsymbol{T}\rangle$

$\mathcal{D}_{GW} = \langle\mathcal{L}^\top, \boldsymbol{T}\rangle$

Return $\boldsymbol{T}$, $\mathcal{D}_W$, $\mathcal{D}_{GW}$

5.4.2.2 Prior construction and likelihoods

Prior. We impose independent zero-mean unit-variance Gaussian priors on elements of $\mathcal{H}$. The prior for $\mathcal{Z}$ is a multivariate Gaussian distribution whose mean and diagonal covariance matrix are constructed from the inferred multi-partite graph and the structural latent variable $\mathcal{H}$. We use two graph neural networks (GNNs) $g_{pz}^{(\mu)}$ and $g_{pz}^{(\sigma)}$ to map $\mathcal{H}$ and $\mathcal{A}$ to the parameters of $p_{\theta_z}(\mathcal{Z})$. Specifically,

$$p_{\theta_z}(\mathcal{Z}\,|\,\mathcal{H},\mathcal{A}) = \prod_{v\in\mathcal{V}_s}\prod_{i=1}^{N_v} p_{\theta_z}(\mathbf{Z}_{i,:}^{(v)}\,|\,\mathcal{H},\mathcal{A}); \qquad p_{\theta_z}(\mathbf{Z}_{i,:}^{(v)}\,|\,\mathcal{H},\mathcal{A}) = \mathcal{N}(\boldsymbol{\mu}_{pz}^{(v,i)}, \boldsymbol{\sigma}_{pz}^{(v,i)}),$$

$$\text{with} \qquad [\boldsymbol{\mu}_{pz}^{(v,i)}]_{v,i} = g_{pz}^{(\mu)}(\mathcal{H},\mathcal{A}), \qquad [\boldsymbol{\sigma}_{pz}^{(v,i)}]_{v,i} = g_{pz}^{(\sigma)}(\mathcal{H},\mathcal{A}).$$

We note that in this setting, $\mathcal{H}$ is considered as node attributes of the multi-partite interaction graph.

Likelihood of observations. To reconstruct the input graphs in the structured views, we assume that the views and edges are conditionally independent. More specifically, we employ an inner-

product decoder as follows:

$$p_{\theta_g}(\mathfrak{A}_s \,|\, \mathcal{H}_s) = \prod_{v \in \mathcal{V}_s} \prod_{i,j=1}^{N_v} p_{\theta_g}\left(\mathbf{A}^{(v)}_{i,j} \,|\, \mathbf{H}^{(v)}_{i,:},\, \mathbf{H}^{(v)}_{j,:}\right);$$
$$p_{\theta_g}\left(\mathbf{A}^{(v)} i, j \,|\, \mathbf{H}^{(v)}_{i,:}, \mathbf{H}^{(v)}_{j,:}\right) = \mathrm{Ber}\left(\sigma(\mathbf{H}^{(v)}_{i,:}\,(\mathbf{H}^{(v)}_{j,:})^T)\right).$$

To generate the features in unstructured views and node attributes in structured views, we assume that views are conditionally independent. Hence, we can expand the feature reconstruction terms in the the equation (5.2) as follows:

$$p_{\theta_x}(\mathcal{X}_u \,|\, \mathcal{Z}_u) = \prod_{v \in \mathcal{V}_u} p_{\theta_x}(\mathbf{X}^{(v)} \,|\, \mathbf{Z}^{(v)}), \qquad p_{\theta_x}(\mathcal{X}_s \,|\, \mathcal{Z}_s) = \prod_{v \in \mathcal{V}_s} p_{\theta_x}(\mathbf{X}^{(v)} \,|\, \mathbf{Z}^{(v)}).$$

We note that p_{θ_x} could also be view specific depending on whether the node attributes/features in a view are discrete or continuous. In our experiments, we have deployed the Gaussian likelihood with unit variance. The mapping from $\mathcal{Z}$ to the parameters of $p_{\theta_x}(\mathcal{X})$, in our case, the mean of the Gaussian distribution, can be any highly expressive function such as neural networks. We denote these functions by $f_{px}^{(v,s)}$ and $f_{px}^{(v,u)}$.

5.4.3 Inference network and learning

Posterior. We model the posterior of the structural latent variables as a Gaussian distribution and infer its parameters independently for each view. More specifically,

$$q_{\phi_h}(\mathcal{H}_u \,|\, \mathcal{X}_u) = \prod_{v \in \mathcal{V}_u} q_{\phi_h}(\mathbf{H}^{(v)} \,|\, \mathbf{X}^{(v)}), \qquad q_{\phi_h}(\mathcal{H}_s \,|\, \mathcal{X}_s,\, \mathfrak{A}_s) = \prod_{v \in \mathcal{V}_s} q_{\phi_h}(\mathbf{H}^{(v)} \,|\, \mathbf{X}^{(v)},\, \mathbf{A}^{(v)}).$$

We use two GNNs for each structured view, $\{g_{qh}^{(\mu,v)}(\mathbf{X}^{(v)}, \mathbf{A}^{(v)}),\, g_{qh}^{(\sigma,v)}(\mathbf{X}^{(v)}, \mathbf{A}^{(v)})\}_{v \in \mathcal{V}_s}$, and two fully connected neural networks per unstructured view, $\{f_{qh}^{(\mu,v)}(\mathbf{X}^{(v)}),\, f_{qh}^{(\sigma,v)}(\mathbf{X}^{(v)})\}_{v \in \mathcal{V}_u}$, to map inputs to the mean and variance of the posteriors. We consider the variational distribution of $\mathcal{Z}$ to

be a multivariate Gaussian distribution, and it is factorized as follows:

$$q_{\phi_z}(\mathcal{Z}_u \,|\, \mathcal{X}_u) = \prod_{v \in \mathcal{V}_u} q_{\phi_z}(\mathbf{Z}^{(v)} \,|\, \mathbf{X}^{(v)}), \qquad q_{\phi_z}(\mathcal{Z}_s \,|\, \mathcal{X}_s,\, \mathfrak{A}_s) = \prod_{v \in \mathcal{V}_s} q_{\phi_z}(\mathbf{Z}^{(v)} \,|\, \mathbf{X}^{(v)},\, \mathbf{A}^{(v)}).$$

We use two GNNs per structured view, $\{g_{qz}^{(\mu,v)}(\mathbf{X}^{(v)},\, \mathbf{A}^{(v)}),\, g_{qz}^{(\sigma,v)}(\mathbf{X}^{(v)}, \mathbf{A}^{(v)})\}_{v \in \mathcal{V}_s}$, and two fully connected neural networks for each unstructured view, $\{f_{qz}^{(\mu,v)}(\mathbf{X}^{(v)}),\, f_{qz}^{(\sigma,v)}(\mathbf{X}^{(v)})\}_{v \in \mathcal{V}_u}$, in the same fashion as q_{ϕ_h} to infer parameters of q_{ϕ_z}.

Objective function. Having defined the prior and posterior distributions as well as the likelihood, we write the overall loss function as the sum of the negative variational ELBO and FGW regularization terms. Specifically,

$$\begin{aligned}
\mathcal{L} &= -\,\mathrm{ELBO} + \mathcal{L}_{\mathrm{FGW}} \\
&= \mathbb{E}_{q_{\phi_z}(\mathcal{Z}_u \,|\, \mathcal{X}_u)} \log p_{\theta_x}(\mathcal{X}_u \,|\, \mathcal{Z}_u) + \mathbb{E}_{q_{\phi_z}(\mathcal{Z}_s \,|\, \mathcal{X}_s, \mathfrak{A}_s)} \log p_{\theta_x}(\mathcal{X}_s \,|\, \mathcal{Z}_s) + \mathbb{E}_{q_{\phi_h}(\mathcal{H}_s \,|\, \mathcal{X}_s, \mathfrak{A}_s)} \log p_{\theta_g}(\mathfrak{A}_s \,|\, \mathcal{H}_s) \\
&\quad + \mathbb{E}_{q_{\phi_z}(\mathcal{Z}_u, \mathcal{H}_u \,|\, \mathcal{X}_u)} \log p_\theta(\mathcal{Z}_u \,|\, \mathcal{A}, \mathcal{H}) + \mathbb{E}_{q_{\phi_z}(\mathcal{Z}_s, \mathcal{H}_s \,|\, \mathcal{X}_s, \mathfrak{A}_s)} \log p_\theta(\mathcal{Z}_s \,|\, \mathcal{A}, \mathcal{H}) \\
&\quad - \mathbb{E}_{q_{\phi_z}(\mathcal{Z}_u \,|\, \mathcal{X}_u)} \log q_{\phi_z}(\mathcal{Z}_u \,|\, \mathcal{X}_u) - \mathbb{E}_{q_{\phi_z}(\mathcal{Z}_s \,|\, \mathcal{X}_s, \mathfrak{A}_s)} \log q_{\phi_z}(\mathcal{Z}_s \,|\, \mathcal{X}_s,\, \mathfrak{A}_s) \\
&\quad - \mathbb{E}_{q_{\phi_h}(\mathcal{H}_u \,|\, \mathcal{X}_u)} \log q_{\phi_h}(\mathcal{H}_u \,|\, \mathcal{X}_u) - \mathbb{E}_{q_{\phi_h}(\mathcal{H}_s \,|\, \mathcal{X}_s, \mathfrak{A}_s)} \log q_{\phi_h}(\mathcal{H}_s \,|\, \mathcal{X}_u,\, \mathfrak{A}_s)) \\
&\quad + \mathbb{E}_{q_{\phi_h}(\mathcal{H}_u \,|\, \mathcal{X}_u)} \log p(\mathcal{H}_u) + \mathbb{E}_{q_{\phi_h}(\mathcal{H}_s \,|\, \mathcal{X}_s, \mathfrak{A}_s)} \log p(\mathcal{H}_s) \\
&\quad + \sum_{v \in \mathcal{V}} \sum_{\substack{v' \in \mathcal{V} \\ v' \neq v}} \mathcal{D}_{\mathrm{FGW}} \left(p(\mathbf{H}^{(v)}),\, p(\mathbf{H}^{(v')}) \right).
\end{aligned} \tag{5.7}$$

While, as mentioned previously, we use the Sinkhorn algorithm to calculate the $\mathcal{D}_{\mathrm{FGW}}$, the overall loss is optimized using stochastic gradient descent based optimization algorithms such as Adam [51].

5.5 Experiments

5.5.1 Datasets and evaluation metrics

Datasets. We use the same datasets as BayReL in [41], i.e. microbiome-metabolite interactions in cystic fibrosis (CF) and gene-drug interactions in precision medicine. Dataset description and graph construction procedure are detailed in the Appendix. We want to emphasize that although these datasets have structured views, have no missing samples and their samples are completely paired, in many real-world cases these assumptions are not satisfied. These datasets were chosen merely to get a better understanding of the advantages of MoReL specially compared to BayReL. We evaluate MoReL in different settings. More specifically, we demonstrate the performance of MoReL when: 1) one or both views are unstructured, 2) there are missing samples, and 3) samples are not paired. Furthermore, we have a comprehensive comparison with BayReL when both views are structured.

Evaluation metrics. To quantify the performance of the methods, we use the same evaluation metrics as the ones introduced in BayReL [41]. Since in these datasets, the true negatives, i.e. non-interactions, are not known; and there are only a small subset of true positives, i.e. true interactions, well-known classification metrics cannot be used for evaluation. Therefore, positive accuracy and negative accuracy have been defined to evaluate microbiome-metabolite experiments. Positive accuracy refers to the accuracy of identifying validated interactions with *P. aeruginosa*. Negative accuracy exploits the fact that there should not be any common metabolite targets between known anaerobic microbes (*Veillonella*, *Fusobacterium*, *Prevotella*, and *Streptococcus*) and notable pathogen *P. aeruginosa*. Let $\mathcal{B}$ denote the set of all microbes and $\mathcal{A}_1$ and $\mathcal{A}_2$ represent two disjoint sets of metabolites. Negative accuracy is defined as $1-\frac{\sum_{i\in\mathcal{A}_1}\sum_{j\in\mathcal{A}_2}\sum_{l\in\mathcal{B}}\mathbb{1}(i \textbf{ and } j \text{ are connected to } l)}{|\mathcal{A}_1|\times|\mathcal{A}_2|\times|\mathcal{B}|}$, where $\mathbb{1}(\cdot)$ is the indicator function. Having both higher positive and negative accuracy is desired.

For precision medicine, we compare the prediction sensitivity of identifying known interactions in the test sets while tracking the average density of the overall constructed graphs. We note that inferring very dense graphs would lead to high prediction sensitivity as it will includes most of the

possible interactions. Therefore, tracking the sparsity of the inferred graphs is the key to properly evaluate the models' capability in predicting meaningful interactions.

5.5.2 Baselines and experimental setups

Baselines. We compare MoReL with three baselines including Spearman's Rank Correlation Analysis (**SRCA**), **BCCA** [57], and **BayReL** [41]. While SRCA applies to raw data, BCCA first finds low-dimensional latent representations of views via matrix factorization and then the interactions are discovered based on the correlation between representations. While BCCA and SRCA could not incorporate the structure of data and need a two-step procedure to infer the interaction between features across the views, BayReL is able to use the structure of data and infer the relations without any *ad-hoc* post-processing procedure. However, BayReL suffers from three strict assumptions: 1) All views of data are structured; 2) There are no missing samples in any views; and 3) Samples are paired, i.e. the ID of samples are known. We emphasize that MoReL is the very first model that not only can infer interactions across structured and unstructured views but also is able to handle missing and unpaired samples in different domains, making it more applicable in real-world multi-omics data integration. A widely used method for multi-omics data integration is MOFA [5]. The mathematical modeling of MOFA is the same as BCCA except for the data likelihood part. While BCCA only supports continuous data, MOFA can have likelihoods for discrete random variables (e.g. Poisson). Since our datasets do not have discrete features, we are only reporting BCCA results.

Hyper-parameters. In all of our experiments, to have a fair comparison, architectural hyper-parameters (i.e. number of layers and number of neurons) were set to be the same as in BayReL. Other hyper-parameters that are unique to MoReL were tuned using the validation set. More specifically, the number of hidden layers as well as their dimensions are the same for the corresponding functions in both structured and unstructured views. We use graph convolutional layers [56] for structured views and fully connected layers for unstructured views except for reconstructing $\mathcal{X}$ from $\mathcal{Z}$, for which we use fully connected layers in all of the views. The mapping from inputs to the mean and variance parameters of $\mathcal{H}$ are two 2-layer neural networks (16 and 8 dimensional layers)

Table 5.1: Comparison of positive accuracy (in %) on CF dataset at negative accuracy of $> 97\%$.

	SRCA	BCCA	MoReL $_{\text{uu}}$	MoReL $_{\text{us}}$
Positive accuracy	26.41	28.30±3.21	56.16±1.85	63.77±1.11

with a shared first layer for each view. We use two 2-layer neural networks (16 and 8 dimensional layers) with a shared first layer for each view for the mapping from $\mathcal{H}$ to the mean and variance of $\mathcal{Z}$. We use a 3-layer fully connected neural network (8 and 16 dimensional hidden layers) for each view as the reconstruction function mapping $\mathcal{Z}$ to $\mathcal{X}$. The temperature for relaxed Bernoulli distribution is set to 0.3. The normalizing parameter γ in equation 5.6 is 0.9 while α and β in $\mathcal{D}_{\text{FGW}}$ are set to 1 and 0.5, respectively. We used the exponential decaying learning rate with the decay rate of 0.01 and initial learning rate of 0.01. All of our results are averaged over multiple runs with different random seeds. We have implemented MoReL and all the competing methods in Tensorflow [1]. All the experiments are performed on a workstation with a single NVIDIA P100 GPU.

5.5.3 Discussion, datasets with unstructured views

Table 5.1 shows the performance of three variants of MoReL and competing methods for microbiome-metabolite data integration with the CF data. In these experiments, we assume that the samples are paired and all are available in both views. In MoReL $_{\text{uu}}$, we report the results when both views are unstructured. In MoReL $_{\text{us}}$, we have the graph of interactions between microbiomes while the metabolite view is assumed to be unstructured. Comparing MoReL $_{\text{uu}}$ and baselines that do not incorporate any graph-structured data as input, we observe an almost 30% improvement in positive accuracy while maintaining higher than 97% negative accuracy. This demonstrates that our proposed MoReL, even without any structural information, is effective in inferring meaningful interactions. Further incorporating the network between microbiomes (i.e. MoReL $_{\text{us}}$) leads to a 37% and 7% improvement compared to the baselines and MoReL $_{\text{uu}}$, respectively. This shows not only the importance of incorporating view-specific side information, but also the effectiveness of FGW-based decoder in aligning structured and unstructured views. Further results on interpretability

Table 5.2: Comparison of prediction sensitivity (in %) in the precision medicine experiment.

Avg. degree	0.10	0.15	0.20	0.25	0.30	0.40	0.50
SRCA	8.03	12.00	17.15	20.70	26.85	34.93	45.79
BCCA	9.65±0.75	14.34±0.06	18.96±0.42	23.29±0.52	28.22±0.66	38.02±2.15	46.88±1.88
MoReL $_{uu}$	11.29±0.16	15.74±0.62	21.21±0.81	26.20±1.10	30.47±1.07	39.05±0.75	50.19±0.19
MoReL $_{us}$	12.79±0.39	17.51±2.21	22.82±1.01	29.58±1.08	35.05±1.27	45.74±1.75	53.16±0.96

and robustness of MoReL on CF dataset is provided in Appendix C.

The results for prediction sensitivity of two variants of MoReL and competing methods in the precision medicine experiments are shown in Table 5.2. We observe that both MoReL $_{uu}$, where both views are unstructured, and MoReL $_{us}$, where the graph structure between genes is given, consistently outperform the baselines by a significant margin using graphs of different densities. This, once again, proves that MoReL is able to learn meaningful relations both in sparse and dense graphs. Comparing the results for MoReL $_{us}$ and BCCA, the difference between their performance increases as the the density of the bipartite graph increases. This confirms that MoReL $_{us}$ can identify potential gene-drug interactions more robustly.

5.5.4 Comparison with BayReL

While the primary goal of experiments so far was showing the effectiveness of MoReL in integrating unstructured and structured views, here we investigate the advantages of MoReL over BayReL.

All structured. As mentioned earlier in the manuscript, BayReL assumes that all of the views are structured. To show the expressive power of MoReL, we train it in the same setting as BayReL where all of the views are structured. Particularly, we assume that for CF dataset both metabolites

Table 5.3: Comparison of positive accuracy (in %) on CF dataset with structured views.

	BayReL	MoReL $_{ss}$
Positive accuracy	82.70±4.70	89.50±3.29

Table 5.4: Comparison of prediction sensitivity (in %) in the precision medicine experiment with structured views.

Avg. degree	BayReL	MoReL $_{ss}$
0.4	47.90±0.43	49.24±1.64
0.5	56.76±0.50	58.92±0.40

network and microbiome network are observed in the microbiome-metabolite experiment. Also, in precision medicine experiment both drug network and gene regulatory network are known *a priori*. For fair comparison, we set the number of layers as well as hidden dimensions to be the same in both models. We train MoReL with the exponential decaying learning rate with the initial rate of 0.01 and decay rate of 0.001 for 120 training epochs. For BayReL, we use the setting reported in Hajiramezanali et al. [41]. The results for CF and precision medicine are summarized in Tables 5.3 and 5.4. We see that MoReL outperforms BayReL on CF dataset by a margin of 7% which indicates that knowing the metabolic pathways can greatly improve interaction learning. In precision medicine experiment, we observe a consistent 2% improvement by MoReL compared to BayReL.

We emphasize that the declined performance of MoReL $_{uu}$ and MoReL $_{us}$ (shown in Tables 5.1 and 5.2) compared to BayReL is expected, as they uses less information than BayReL. Incorporating this extra information in MoReL enhances its performance substantially. Note that BayReL is bound to use the same set of functions for all views to account for arbitrarily rotations and transformations, which limits its expressive power. However, the FGW based decoder in MoReL allows to have different processing functions for each view. We argue that this increases the expressive power and plays the key role in enhancing the performance.

Paired vs. unpaired. To show that MoReL can handle unpaired input samples, we perform an ablation study on CF dataset. We reverse the order of samples in metabolite view while keeping the order of samples in microbiome. We report the performance of BayReL and MoReL $_{us}$ where we don't use the structure of metabolite view. The results are shown in Table 5.5. While MoReL

Table 5.5: Comparison of positive accuracy (in %) and negative accuracy (in %) on CF dataset with unpaired samples.

	BayReL	MoReL $_{\text{us}}$
Positive accuracy	31.56	63.24±2.13
Negative accuracy	72	97

performed virtually the same as a completely paired scenario (shown in Table 5.1), BayReL's performance drastically declined. We note that the reported negative accuracy is the best one achieved by BayReL.

Missing samples. We should again point out that in a setting where all views are structured but the number of node attributes are not the same in different views, BayReL cannot be deployed (as it uses the same processing functions for all views). To see how MoReL $_{\text{us}}$ performs in such a scenario, we randomly remove 10% of samples in metabolite view of CF dataset. MoReL achieves positive accuracy (in %) of 61.36±3.74 with negative accuracy of 97%. This again shows the robustness of FGW-based decoder in aligning nodes with different number of samples.

Computational complexity. We have also benchmarked computational complexity of MoReL and BayReL by tracking their runtime on CF dataset on the same hardware. While BayReL takes 0.6 seconds per training epoch, MoReL takes 2.7 seconds per training epoch. This is due to the computational overhead caused by deploying FGW regularization. Considering the model flexibility and significant prediction performance improvement, such computational overhead is acceptable.

6. CONCLUSIONS

In this dissertation, we have developed multiple novel Bayesian models to address challenges in graph analytics problems. Our proposed methods model different components of a graph including nodes, node attributes, and the graph structure, as distributions, hence they are equipped with natural uncertainty estimates.

First, we have developed a Bayesian graph representation learning method that infers implicit posterior as node embeddings. Combining the advantages of semi-implicit hierarchical variational distribution and VGAE with a Bernoulli-Poisson link decoder, SIG-VAE has been shown to capture both the graph structural and node attribute information in the latent space. By providing a surrogate evidence lower bound that is asymptotically exact, the optimization problem for SIG-VAE model inference is amenable via stochastic gradient descent, without compromising the flexbility of its variational distribution. Our experiments with different graph datasets have shown the promising capability of SIG-VAE in a range of graph analysis applications with interpretable latent representations, thanks to the hierarchical construction that diffuses the distributions of neighborhood nodes in given graphs.

Next, we have developed a unified framework for connection sampling in GNNs that generalizes existing stochastic regularization techniques for training GNNs. Our proposed method, Graph DropConnect (GDC), not only alleviates over-smoothing and over-fitting tendencies of deep GNNs, but also enables learning with uncertainty in graph analytic tasks with GNNs. To that end, we have shown that training a GNN with our learnable GDC is equivalent to an approximation of training Bayesian GNNs. Instead of using fixed sampling rates, our GDC technique parameters can be trained jointly with GNN model parameters. Our experimental results shows that GDC boosts the performance of GNNs in semi-supervised classification task by alleviating over-smoothing and over-fitting. We further show that the quality of uncertainty derived by GDC is better than DropOut in GNNs.

Finally, we have proposed MoReL, a novel Bayesian deep generative model that efficiently

infers hidden molecular relations across heterogeneous views of data. By using a fused Gromov-Wasserstein (FGW) based decoder, MoReL addresses several main shortcomings of the state-of-the-art omics data integration model. Specifically, MoReL can 1) integrate both structured and unstructured omics datasets while accounting for arbitrarily permutation and/or transformation caused by processing features with different deep functions across the views; 2) handle unpaired samples across the views of data; 3) combine multiple views from different data sources with any number of missing samples. Our experiments on two real-world datasets have demonstrated substantial improvement in inferring meaningful relations as well as improving prediction sensitivity compared to the competing methods. MoReL has shown the promising potential for multi-view learning, in particular multi-omics data integration for biological knowledge discovery, when facing heterogeneous data from different views.

REFERENCES

[1] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48. ACM, 2013.

[3] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008.

[4] David Alvarez-Melis and Tommi Jaakkola. Gromov-Wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1881–1890. Association for Computational Linguistics, October-November 2018.

[5] Ricard Argelaguet, Britta Velten, Damien Arnol, Sascha Dietrich, Thorsten Zenz, John C Marioni, Florian Buettner, Wolfgang Huber, and Oliver Stegle. Multi-omics factor analysis—a framework for unsupervised integration of multi-omics data sets. *Molecular systems biology*, 14(6):e8124, 2018.

[6] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, Joseph Lehár, Gregory V Kryukov, Dmitriy Sonkin, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer

drug sensitivity. *Nature*, 483(7391):603–607, 2012.

[7] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[8] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[9] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

[10] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[11] Christopher M Bishop and Michael E Tipping. Variational relevance vector machines. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 46–53. Morgan Kaufmann Publishers Inc., 2000.

[12] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[13] Aleksandar Bojchevski and Stephan Gunnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

[14] Shahin Boluki, Randy Ardywibowo, Siamak Zamani Dadaneh, Mingyuan Zhou, and Xiaoning Qian. Learnable Bernoulli dropout for Bayesian deep learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 3905–3916, 2020.

[15] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany,

August 2016. Association for Computational Linguistics.

[16] Charlotte Bunne, David Alvarez-Melis, Andreas Krause, and Stefanie Jegelka. Learning generative models across incomparable spaces. In *International Conference on Machine Learning*, pages 851–861. PMLR, 2019.

[17] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[18] Jia Chen, Gang Wang, Yanning Shen, and Georgios B Giannakis. Canonical correlation analysis of datasets with a common source graph. *IEEE Transactions on Signal Processing*, 66(16):4398–4408, 2018.

[19] Jia Chen, Gang Wang, and Georgios B Giannakis. Multiview canonical correlation analysis over graphs. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2947–2951. IEEE, 2019.

[20] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.

[21] Liqun Chen, Zhe Gan, Yu Cheng, Linjie Li, Lawrence Carin, and Jingjing Liu. Graph optimal transport for cross-domain alignment. In *International Conference on Machine Learning*, pages 1542–1553. PMLR, 2020.

[22] Siheng Chen, Aliaksei Sandryhaila, José MF Moura, and Jelena Kovačević. Signal recovery on graphs: Variation minimization. *IEEE Transactions on Signal Processing*, 63(17):4609–4624, 2015.

[23] Samir Chowdhury and Facundo Mémoli. The gromov–wasserstein distance between networks and stable network invariants. *Information and Inference: A Journal of the IMA*, 8(4):757–787, 2019.

[24] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

[25] Siamak Zamani Dadaneh, Shahin Boluki, Mingzhang Yin, Mingyuan Zhou, and Xiaoning Qian. Pairwise supervised hashing with Bernoulli variational auto-encoder and self-control gradient estimator. In *The Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020.

[26] Siamak Zamani Dadaneh, Shahin Boluki, Mingyuan Zhou, and Xiaoning Qian. Arsm gradient estimator for supervised learning to rank. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3157–3161, 2020.

[27] Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

[28] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[29] Michaël Defferrard, Lionel Martin, Rodrigo Pena, and Nathanaël Perraudin. Pygsp: Graph signal processing in python, 2017. URL `https://doi.org/10.5281/zenodo.1003158`.

[30] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 240–250. Association for Computational Linguistics, June 2019.

[31] Michael C Fu. Gradient estimation. *Handbooks in operations research and management science*, 13:575–616, 2006.

[32] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

[33] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[34] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in neural information*

processing systems, pages 3581–3590, 2017.

[35] Zoubin Ghahramani and Thomas L Griffiths. Infinite latent feature models and the Indian buffet process. In *Advances in neural information processing systems*, pages 475–482, 2006.

[36] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[37] Anirudh Goyal Alias Parth Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in neural information processing systems*, pages 6713–6723, 2017.

[38] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[39] Ehsan Hajiramezanali, Siamak Zamani Dadaneh, Alireza Karbalayghareh, Mingyuan Zhou, and Xiaoning Qian. Bayesian multi-domain learning for cancer subtype discovery from next-generation sequencing count data. In *Advances in Neural Information Processing Systems*, pages 9115–9124, 2018.

[40] Ehsan Hajiramezanali, Arman Hasanzadeh, Nick Duffield, Krishna R Narayanan, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2019.

[41] Ehsan Hajiramezanali, Arman Hasanzadeh, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayrel: Bayesian relational learning for multi-omics data integration. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19251–19263. Curran Associates, Inc., 2020.

[42] Ehsan Hajiramezanali, Arman Hasanzadeh, Nick Duffield, Krishna Narayanan, Mingyuan Zhou, and Xiaoning Qian. Semi-implicit stochastic recurrent neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3342–3346. IEEE, 2020.

[43] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[44] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[45] A. Hasanzadeh, X. Liu, N. Duffield, and K. R. Narayanan. Piecewise stationary modeling of random processes over graphs with an application to traffic prediction. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3779–3788, 2019.

[46] Arman Hasanzadeh, Ehsan Hajiramezanali, Nick Duffield, Krishna R Narayanan, Mingyuan Zhou, and Xiaoning Qian. Semi-implicit graph variational auto-encoders. In *Advances in Neural Information Processing Systems*, 2019.

[47] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

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

[49] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[50] Vassilis Kalofolias, Xavier Bresson, Michael Bronstein, and Pierre Vandergheynst. Matrix completion on graphs. *arXiv preprint arXiv:1408.1717*, 2014.

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

[52] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[53] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.

[54] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

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

[56] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[57] Arto Klami, Seppo Virtanen, and Samuel Kaski. Bayesian canonical correlation analysis. *Journal of Machine Learning Research*, 14(Apr):965–1003, 2013.

[58] Ponnambalam Kumaraswamy. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46(1-2):79–88, 1980.

[59] Su-In Lee, Safiye Celik, Benjamin A Logsdon, Scott M Lundberg, Timothy J Martins, Vivian G Oehler, Elihu H Estey, Chris P Miller, Sylvia Chien, Jin Dai, et al. A machine learning approach to integrate big data for precision medicine in acute myeloid leukemia. *Nature communications*, 9(1):1–13, 2018.

[60] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[61] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[62] Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 496–503, 2003.

[63] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296, 2011.

[64] Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient MCMC.

In *Advances in Neural Information Processing Systems*, pages 2917–2925, 2015.

[65] Lars Maaloe, Casper Kaae Sonderby, Soren Kaae Sonderby, and Ole Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, pages 1445–1453, 2016.

[66] David JC MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

[67] Facundo Mémoli. Spectral gromov-wasserstein distances for shape matching. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 256–263. IEEE, 2009.

[68] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011.

[69] Dmitry Molchanov, Valery Kharitonov, Artem Sobolev, and Dmitry Vetrov. Doubly semi-implicit variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2593–2602. PMLR, 2019.

[70] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707, 2017.

[71] James T Morton, Alexander A Aksenov, Louis Felix Nothias, James R Foulds, Robert A Quinn, Michelle H Badri, Tami L Swenson, Marc W Van Goethem, Trent R Northen, Yoshiki Vazquez-Baeza, et al. Learning representations of microbe–metabolite interactions. *Nature methods*, 16(12):1306–1314, 2019.

[72] Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018.

[73] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[74] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[75] Maximillian Nickel and Douwe Kiela. Poincare embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc., 2017.

[76] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José M. F. Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[77] John Paisley, David Blei, and Michael Jordan. Variational Bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.

[78] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[79] Ferran Parés, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Fluid communities: a competitive, scalable and diverse community detection algorithm. In *International Conference on Complex Networks and their Applications*, pages 229–240. Springer, 2017.

[80] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

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

[82] Gabriel Peyré, Marco Cuturi, and Justin Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672. PMLR, 2016.

[83] Robert A Quinn, Katrine Whiteson, Yan-Wei Lim, Peter Salamon, Barbara Bailey, Simone Mienardi, Savannah E Sanchez, Don Blake, Doug Conrad, and Forest Rohwer. A

winogradsky-based culture system shows an association between microbial fermentation and cystic fibrosis exacerbation. *The ISME journal*, 9(4):1024–1038, 2015.

[84] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333, 2016.

[85] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[86] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[87] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

[88] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[89] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4): 1–11, 2015.

[90] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[91] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[92] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[93] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

[94] Romain Thibaux and Michael I Jordan. Hierarchical beta processes and the Indian buffet process. In *Artificial Intelligence and Statistics*, pages 564–571, 2007.

[95] Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pages 6275–6284. PMLR, 2019.

[96] Alexandre Irrthum Vân Anh Huynh-Thu, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PloS one*, 5(9), 2010.

[97] Titouan Vayer, Laetita Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. Fused gromov-wasserstein distance for structured objects: theoretical foundations and mathematical properties. *arXiv preprint arXiv:1811.02834*, 2018.

[98] Titouan Vayer, Rémi Flamary, Romain Tavenard, Laetitia Chapel, and Nicolas Courty. Sliced gromov-wasserstein. In *NeurIPS 2019-Thirty-third Conference on Neural Information Processing Systems*, volume 32, 2019.

[99] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[100] Christian Von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399, 2002.

[101] Alex H Wagner, Adam C Coffman, Benjamin J Ainscough, Nicholas C Spies, Zachary L Skidmore, Katie M Campbell, Kilannin Krysiak, Deng Pan, Joshua F McMichael, James M Eldred, et al. Dgidb 2.0: mining clinically relevant drug–gene interactions. *Nucleic acids research*, 44(D1):D1036–D1044, 2016.

[102] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

[103] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440, 1998.

[104] Jason Weston, Frederic Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

[105] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[106] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

[107] Hongteng Xu, Dixin Luo, and Lawrence Carin. Scalable gromov-wasserstein learning for graph partitioning and matching. *Advances in neural information processing systems*, 32: 3052–3062, 2019.

[108] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. Gromov-wasserstein learning for graph matching and node embedding. In *International conference on machine learning*, pages 6932–6941. PMLR, 2019.

[109] Hongteng Xu, Dixin Luo, Ricardo Henao, Svati Shah, and Lawrence Carin. Learning autoencoders with relational regularization. In *International Conference on Machine Learning*, pages 10576–10586. PMLR, 2020.

[110] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462, 2018.

[111] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[112] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

[113] Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *International*

Conference on Machine Learning, pages 5660–5669, 2018.

[114] Mingzhang Yin and Mingyuan Zhou. ARM: Augment-REINFORCE-merge gradient for stochastic binary networks. In *International Conference on Learning Representations*, 2019.

[115] Mingzhang Yin, Nhat Ho, Bowei Yan, Xiaoning Qian, and Mingyuan Zhou. Probabilistic best subset selection by gradient-based optimization. *arXiv preprint arXiv:2006.06448*, 2020.

[116] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 793–803, 2019.

[117] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems*, 31:5165–5175, 2018.

[118] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5829–5836, 2019.

[119] Mingyuan Zhou. Infinite edge partition models for overlapping community detection and link prediction. In *Artificial Intelligence and Statistics*, pages 1135–1143, 2015.

[120] Mingyuan Zhou, Haojun Chen, Lu Ren, Guillermo Sapiro, Lawrence Carin, and John W Paisley. Non-parametric Bayesian dictionary learning for sparse image representations. In *Advances in neural information processing systems*, pages 2295–2303, 2009.

[121] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

APPENDIX A

ADDITIONAL DISCUSSION AND RESULTS FOR SIG-VAE

In this chapter, we first provide the detailed review of the related literature as well as the connection to our proposed work. Dataset statistics, network setups, and implementation details of performance evaluation experiments for different graph analytic tasks are then presented with richer experimental results in addition to the ones discussed in the main text.

A.1 Node embedding

Node embedding is to represent each node in a graph by a low-dimensional vector in a latent space. The geometric relations of vectors in the latent space reflect the probability of two corresponding nodes interacting with each other in the graph [44]. A good node embedding preserves node connectivity in graph as well as local neighborhood structures. More formally, node embedding can be formulated as follows.

Node embedding. Given a graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ the set of edges, with the adjacency matrix $\mathbf{A}$, $\mathbf{X} \in \mathbb{R}^{N \times M}$ denoting M-dimensional node attributes for $N = |\mathcal{V}|$ nodes, and a function $s_G : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ measuring node similarity, find an encoder function, $\mathbf{ENC} : \mathbb{R}_+^{N \times N} \times \mathbb{R}^{N \times M} \to \mathbb{R}^l$, a decoder function, $\mathbf{DEC} : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}_+$, and a latent representation of nodes $\mathbf{Z} \in \mathbb{R}^{N \times l}$ such that

$$\mathbf{Z} = \mathbf{ENC}(\mathbf{A}, \mathbf{X}),$$
$$\hat{s}_{i,j} \triangleq \mathbf{DEC}(\mathbf{z}_i, \mathbf{z}_j),$$

where $\mathbf{z}_i$ corresponds to the embedding representation of node $v_i \in \mathcal{V}$. Optimal parameters of **ENC**

and **DEC** functions can be derived by finding the solutions to the following optimization problem

$$\min_{\mathbf{ENC},\mathbf{DEC}} \quad \sum_{i=1}^{N} \mathbf{loss}(\hat{s}_{i,j}, s_G(v_i, v_j)),$$

where **loss** is a user-specified loss function based on the ultimate objective of network analysis.

Different node embedding methods vary in the choice of the **loss** function, s_G, **ENC**, **DEC** and the optimization algorithm. For example, in graph factorization (GF) method [2], s_G is defined based on the adjacency matrix, i.e., $s_G(v_i, v_j) = A_{i,j}$; **loss** is the mean squared error; and the inner-product decoder is adopted, i.e., $\mathbf{DEC}(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T \mathbf{z}_j$.

A.2 Variational inference with normalizing flows

To increase the expressive power of a probabilistic model, a simple but powerful idea is to transform the corresponding random variables with complex deterministic and/or stochastic mappings. To construct flexible, arbitrarily complex and scalable approximate posterior distributions, normalizing flow (NF) transforms a simple random variable through a sequence of invertible differentiable functions with tractable Jacobians. More specifically, NF uses an invertible, smooth mapping $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ to transform a random variable z with distribution $q(z)$ to the resulting random variable $z' = f(z)$ with the distribution:

$$q(z') = q(z) \left| \det \frac{\partial f^{-1}}{\partial z'} \right| = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1}. \tag{A.1}$$

One may apply a chain of K transformations f_k to obtain the density $q_K(z)$ from a random variable z_0 with distribution q_0 as:

$$\ln q_K(z_K) = \ln q_0(z_0) - \sum_k \ln \left| \det \frac{\partial f_k}{\partial z_k} \right|. \tag{A.2}$$

While normalizing flow helps to improve the model flexibility of the corresponding variational posterior, it requires the mapping to be deterministic and invertible, and the mixing distribution in the hierarchy to have an explicit density function. Removing these restrictions, there have been

several recent attempts to define highly flexible variational posterior with implicit models. While an implicit variational distribution can be made highly flexible, it becomes necessary in each iteration to address the problem of density ratio estimation, which is often transformed into a problem related to learning generative adversarial networks [36]. SIVI addresses this issue by using an analytic conditional variational distribution which is not required to be reparameterizable.

A.3 Graph dataset details

Table A.1 provides the detailed statistics of the graph datasets used in our experiments.

A.4 Experimental setups and hyperparameter tuning

Interpretable latent representations experiments. In these experiments, the code provided by Kipf and Welling [55] is used to derive the embedding for VGAE. The size of the first hidden layer of VGAE is 256 and the size of the output layer is 3. For SIG-VAE, two stochastic layers with sizes equal to [32, 32] and an additional GCN layer of size 16 are used to model the $\boldsymbol{\mu}$. The dimension of injected standard Gaussian noises $[\epsilon_1, \epsilon_2]$ are [32, 32]. Covariance matrix $\boldsymbol{\Sigma}$ is deterministic and is inferred through two layers of GCNs with sizes equal to [32, 16]. To remove the effect of decoder, we consider the inner-product decoder for this set of experiments.

Link prediction with node attributes For SIG-VAE, we use a stochastic layer with size equal to 32 and an additional GCN layer of size 16 is used to model $\boldsymbol{\mu}$. The dimension of injected

Table A.1: Graph dataset statistics for SIG-VAE experiments.

Dataset	Type	Nodes	Edges
Cora	Citation	2,708	5,429
Citeseer	Citation	3,327	4,732
Pubmed	Citation	19,717	44,338
USAir	Transportation	332	2,126
NS	Collaboration	1,589	2,742
Router	Internet	5,022	6,258
Power	Energy	4,941	6,594
Yeast	Protein	2,375	11,693

Bernoulli noise ϵ for the stochastic layer is 64. For SIVI-VGAE, we use two GCN layers with sizes equal to [32, 16] followed by a fully connected layers with size 16 to infer $\boldsymbol{\mu}$. We inject 64-dimensional Bernoulli noise to the fully connected layer. We implement NF-VGAE by extending VGAE (two GCN layers with sizes equal to [32, 16]) with invertible linear-time transformations of length 4 to keep its number of parameters close to the competing methods. We learn the model parameters for 3500 epochs with the learning rate 0.0005 and the validation set used for early stopping.

Link prediction without node attributes. For SIG-VAE, we use a stochastic layer with size equal to 32 and an additional GCN layer of size 16 is used to model $\boldsymbol{\mu}$. The dimensions of injected Bernoulli noise ϵ is 32. For SIVI-VGAE, we use two GCN layers with sizes equal to [32, 16] followed by a fully connected layer with sizes 16 to infer $\boldsymbol{\mu}$. We inject 32-dimensional Bernoulli noise to the fully connected layers. We learn the all model parameters for 2500 epochs with the learning rate 0.0005 and use the validation set for the early stopping. We use a two-stage learning process for SIG-VAE, SIVI-VGAE, and NF-VGAE. First, the embedding of each node is learned in the 128-dimensional latent space while injecting 5-dimensional Bernoulli noise to the system in the case of SIG-VAE and Naive SIG-VGAE. Then we use the learned embedding as node features for the second stage to learn 16 dimensional embedding while injecting more noise to SIG-VAE. We follow the same procedure for SIVI-VGAE too.

Graph generation. We have not specifically tuned the model but directly adopt the implementation setups for link prediction with and without node attributes.

Node classification and graph clustering. We use two GCN layers with sizes equal to [32, 16] followed by a fully connected layer with sizes 16 to infer $\boldsymbol{\mu}$. We inject 64-dimensional Bernoulli noise to the GCN layers. Learning rate is set to be 0.0005.

Analysis of the complexity. For the analysis of the real-world graph dataset Cora on a single GeForce GTX 1080 GPU node, it took 24.5, 11.7 , and 9.5 seconds for SIG-VAE, NF-VGAE, and VGAE methods with 100 epochs, respectively. For the analysis of the small real-world graph dataset NS on a same GPU node, it took 7.23, 7.84, and 7.09 seconds for SIG-VAE, NF-VGAE, and VGAE

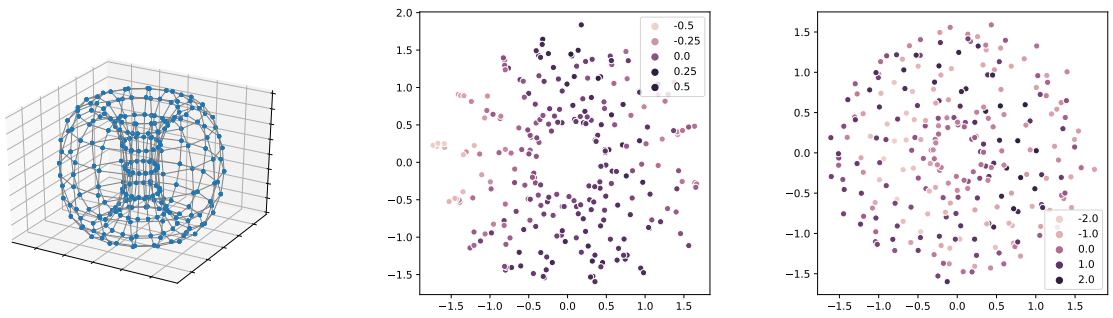

Figure A.1: Torus graph (**Left**) and its latent representation using SIG-VAE (**Middle**) and VGAE (**Right**). The latent representations (middle and right) are heat maps in $\mathbb{R}^3$. We expect that the embedding of the torus graph with the inner-product decoder to be multiple lines coming out of the center in $\mathbb{R}^3$, which is clearly better captured by SIG-VAE.

methods with 100 epochs, respectively.

A.5 Additional experimental results

A.5.1 Interpretable latent representations

In addition to the results of the **Swiss roll** graph in the manuscript, we also compare the latent representations of SIG-VAE and VGAE for a **torus** graph with 256 nodes connected by 512 edges as illustrated in Figure A.1. We consider the coordinates of each node in $\mathbb{R}^3$ as node attributes for both methods in this experiment. We expect that the embedding of nodes to be symmetric since the graph itself is symmetric. We know that the inner-product decoder tries to embed a ring graph to a circle in space. Also, connected nodes should be in the same angle. Thus, the embedding of connected circles as in torus in $\mathbb{R}^3$ should be some lines coming out of center while their altitude is changing periodically. As we can see in Figure A.1, SIG-VAE demonstrates a better latent representation than VGAE. To gain more insights about the posterior distributions, we show the distributions inferred by SIG-VAE and VGAE for three nodes in Figure A.2. The inferred distributions are indeed skewed and multi-modal, very different from Gaussian. Being able to capture complex non-Gaussian distributions helps the model to represent the graph structure in a more meaningful way.

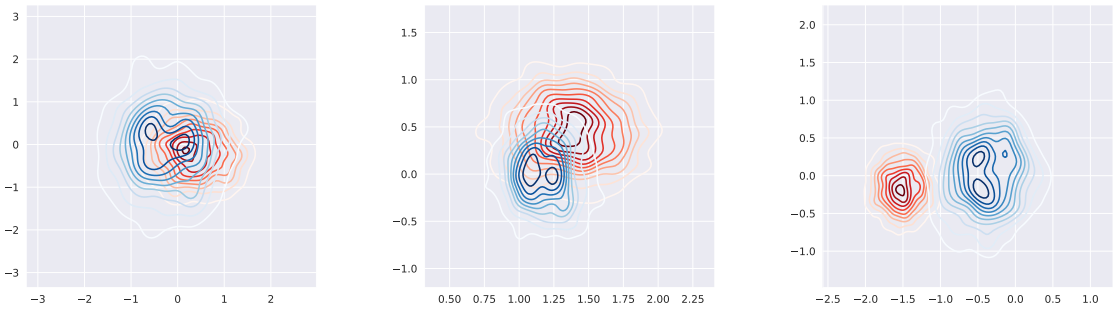

Figure A.2: Latent representation distributions of three nodes in the torus graph using SIG-VAE (**Blue**) and VGAE (**Red**). SIG-VAE clearly infers more complex distributions that are multi-modal or skewed. This helps SIG-VAE to better represent the nodes in the latent space.

A.5.2 Link prediction

More complete link prediction results with the standard deviation values from different runs are presented here. As we can see in Tables A.2 and A.3, SIG-VAE shows the consistent superior performance compared to the competing methods, especially over the baseline VGAE, in terms of both AUC and AP. It is interesting to note that, while the proposed sparse decoder works well for the sparser graphs, especially NS and Router sparse datasets, SIG-VAE with the inner-product decoder shows superior performance for the USAir graph which is much denser. Compared to the baseline VGAE, both SIVI-VGAE and NF-VGAE improve the results with a large margin in terms of both

Table A.2: AUC of link prediction in networks without node attributes. * indicates that the numbers are reported from Zhang and Chen [117].

Data	MF*	SBM*	N2V*	LINE*	SC*	VGAE*	SEAL*	G2G	**NF-VGAE**	**SIVI-VGAE**	**SIG-VAE**(IP)	**SIG-VAE**
USAir	94.08	94.85	91.44	81.47	74.22	89.28	97.09	92.17	95.74	94.22	**97.56**	94.52
	±0.80	±1.14	±1.78	±10.71	±3.11	±1.99	±0.70	±1.65	± 1.74	±0.43	±0.23	±0.28
NS	74.55	92.30	91.52	80.63	89.94	94.04	97.71	98.18	98.38	98.00	**98.75**	**99.17**
	±4.34	±2.26	±1.28	±1.90	±2.39	±1.64	±0.93	±0.51	±0.46	±0.34	±0.12	±0.45
Yeast	90.28	91.41	93.67	87.45	93.25	93.88	97.20	97.34	97.86	93.36	**98.11**	**98.32**
	±0.69	±0.60	±0.46	±3.33	±0.40	±0.21	±0.64	±0.32	±0.44	±0.63	±0.18	±0.26
Power	50.63	66.57	76.22	55.63	91.78	71.20	84.18	91.35	94.61	93.67	**95.045**	**96.23**
	±1.10	±2.05	±0.92	±1.47	±0.61	±1.65	±1.82	±0.41	±0.65	±0.78	±0.15	±0.12
Router	78.03	85.65	65.46	67.15	68.79	61.51	95.68	85.98	93.56	92.66	**95.94**	**96.13**
	±1.63	±1.93	±0.86	±2.10	±2.42	±1.22	±1.22	±1.25	±0.79	±0.25	±0.23	±0.26

Table A.3: AP of link prediction in networks without node attributes. * indicates that the numbers are reported from Zhang and Chen [117].

Data	MF*	SBM*	N2V*	LINE*	SC*	VGAE*	SEAL*	G2G	**NF-VGAE**	**SIVI-VGAE**	**SIG-VAE**(IP)	**SIG-VAE**
USAir	94.36	95.08	89.71	79.70	78.07	89.27	95.70	90.22	96.27	94.48	**97.50**	94.95
	±0.79	±1.10	±2.97	±11.76	±2.92	±1.29	±0.21	±2.61	± 1.51	±0.80	±0.14	±0.28
NS	78.41	92.13	94.28	85.17	90.83	95.83	98.12	97.43	98.52	97.83	**98.53**	**99.24**
	±3.85	±2.36	±0.91	±1.65	±2.16	±1.04	±0.77	±2.34	±0.29	±0.40	±0.09	±0.40
Yeast	92.01	92.73	94.90	90.55	94.63	95.19	97.95	97.83	98.18	94.24	**97.97**	**98.41**
	±0.47	±0.44	±0.38	±2.39	±0.56	±0.36	±0.35	±0.28	±0.22	±0.46	±0.14	±0.13
Power	53.50	65.48	81.49	56.66	91.00	75.91	86.69	92.29	95.76	93.80	**96.50**	**97.28**
	±1.22	±1.85	±0.86	±1.43	±0.58	±1.56	±1.50	±0.37	±0.55	±0.83	±0.17	±0.30
Router	82.59	84.67	68.66	71.92	73.53	70.36	95.66	86.28	95.88	92.80	**94.94**	**96.86**
	±1.38	±1.89	±1.49	±1.53	±1.47	±0.85	±1.23	±1.32	±0.34	±0.18	±0.13	±0.27

AUC and AP, showing the benefits of more flexible variational posterior. Comparing SIG-VAE with two other flexible inference methods shows that not only SIG-VAE is not restricted to the Gaussian assumption, which is not a good fit for link prediction with the inner-product decoder [27], but also it is able to model flexible posterior considering graph topology. The results for the link prediction of the Power graph clearly magnifies this fact as SIG-VAE improves the accuracy by 34% compared to VGAE.

A.5.3 Drug-drug interaction network

Here, we also include the results on a drug-drug interaction network [106] capturing drug effect change due to the action of another drug. When several drugs are administered together, there might be adverse drug reactions due to drug-drug interactions. It is thus crucial to identify them during drug development. With a similar setup as in the manuscript, SIG-VAE achieves AUC and AP at 92.51 and 92.81, respectively. For comparison, VGAE gets 90.22 (AUC) and 90.29 (AP), respectively, and GAE gets 90.73 (AUC) and 91.15 (AP). Hyperparameters are inherited from the original paper of each method.

APPENDIX B

ADDITIONAL DISCUSSION AND RESULTS FOR GDC

In this chapter, we first provide an ablation study on local GDC. Dataset statistics, and further implementation details are also presented. Finally, schematics of different stochastic regularization techniques for GCNs are provided.

B.1 Ablation study: global versus local

We further investigate our learnable GDC, in which for each edge at each layer a different connection sampling distribution is learned. We refer to this scenario as the *local* learnable GDC. This, indeed, is a more general case than learning a single distribution for all edges in a layer. Expanding the variational beta-Bernoulli GDC to local learnable GDC is straightforward. Note that the KL term in the loss function can be derived in the same manner as in the global learnable GDC – as described in Section 4 of the paper – except that it will include the sum of $\mathrm{num\_layers} \times \mathrm{num\_edges}$ terms as opposed to the $\mathrm{num\_layers}$ terms in the global GDC.

By training the aforementioned model on the citation datasets, we find that the accuracy degrades and the KL divergence reduces to zero for every choice of prior. This phenomenon, which is known as *posterior collapse* or *KL vanishing*, is a common problem in variational auto-encoders for language modeling [15, 37, 30]. It is often due to over-parametrization in the model, which is indeed the case in the local learnable GDC. A solution to this issue could be making the parameters of the distribution dependent on the graph topology and/or node attributes. We leave this for future studies.

B.2 Datasets and implementation details

All of the models are implemented in PyTorch [80]. All of the simulations are conducted on a single NVIDIA GeForce RTX 2080 GPU node. We evaluate our proposed methods, GCN-BBDE and GCN-BBGDC, and baselines on three standard citation network benchmark datasets. We

Table B.1: Graph dataset statistics for GDC experiments.

Dataset	# Classes	# Nodes	# Edges	# Features
Cora	7	2,708	5,429	1,433
Citeseer	3	3,327	4,732	3,703
Cora-ML	7	2,995	8,416	2,879

preprocess and split the dataset as done in [56] and [13]. For Cora and Cora-ML, we use 140 nodes for training, 500 nodes for validation and 1000 nodes for testing. For Citeseer, we use 120 nodes for training and the same number of nodes as Cora for validation and testing. Table B.1 provides the detailed statistics of the graph datasets used in our experiments. The warm-up factor used in GCN-BBGDC with more than 2 layers for Cora and Cora-ML is $\min(\{1, \text{epoch}/20\})$, and for Citeseer is $\min(\{1, \text{epoch}/40\})$. We have deployed Adam optimizer [51] in all of our experiments.

B.3 GDC versus other stochastic regularization techniques

To further clarify the differences of our proposed GDC from existing stochastic regularization techniques, we draw the schematics of a GCN layer to which DropOut, DropEdge, Node Sampling, and our GDC are applied; shown in figures below. The input graph topology for the GCN layer is depicted in B.1. The number of input and output features are both two in this toy example.

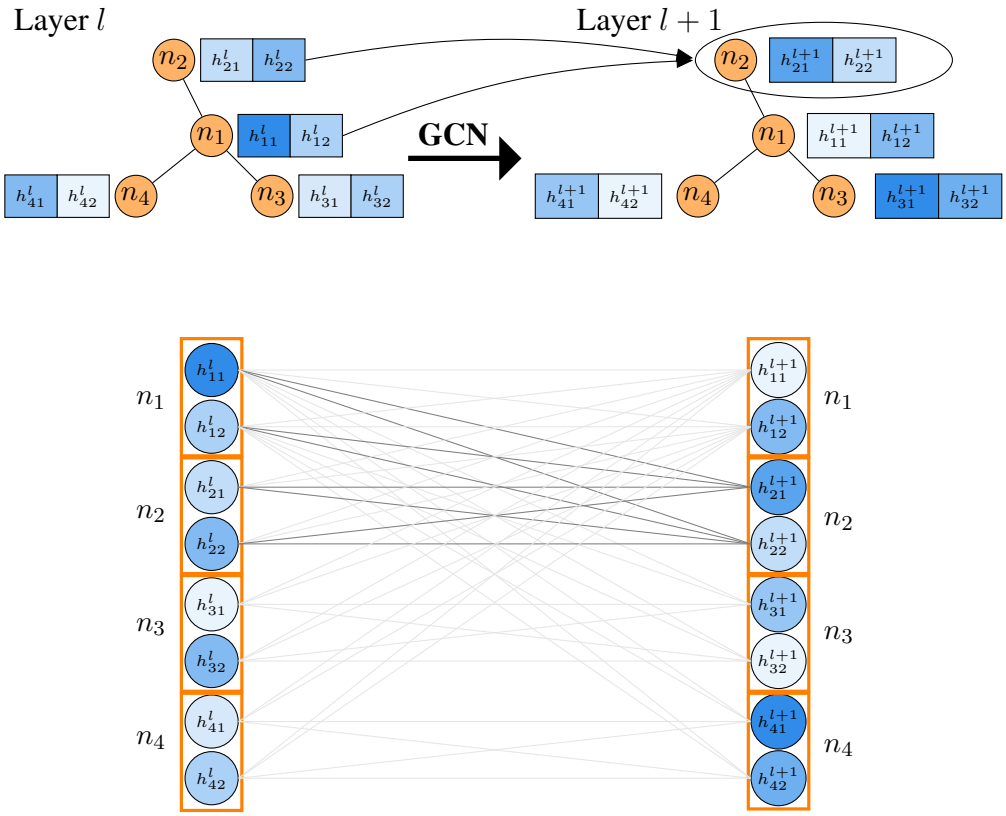

Figure B.1: **Top**: Schematic of a GCN layer on a graph with 4 nodes. Number of both input and output features are two. The connections are localized as explicitly depicted for node 2. **Bottom**: The same GCN layer shown in a more conventional way, i.e. each layer is a vector of neurons or features. Each circle is a feature and each square represents a node. The connections are sparse and the sparsity is based on the input graph topology. The connections for node 2 in layer $l+1$ are highlighted.

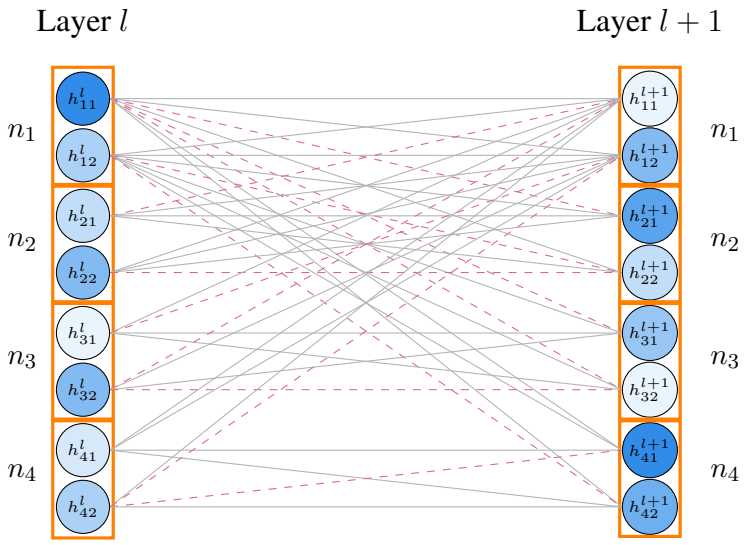

Figure B.2: Schematic of our proposed GDC. Each circle is a feature and each square represents a node. GDC drops connections independently across layers. The dashed lines show dropped connections and the gray ones show the kept connections.

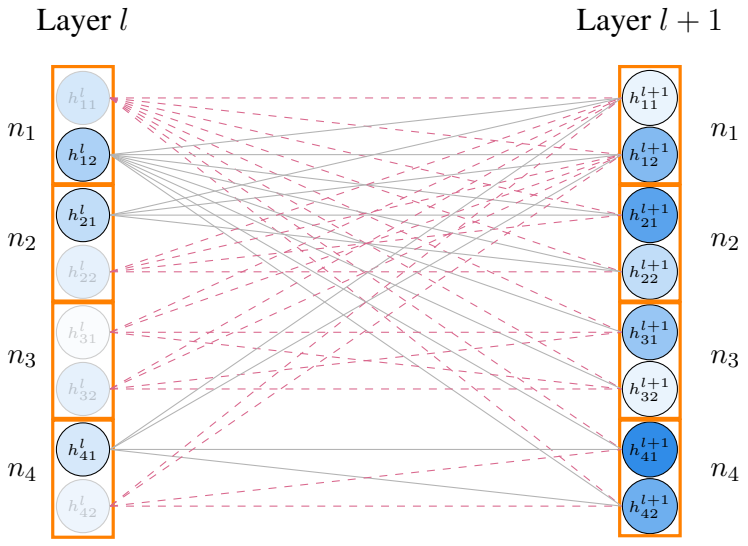

Figure B.3: Schematic of DropOut [91]. Each circle is a feature and each square represents a node. DropOut drops features at each layer. The faded circles represent dropped features while the other ones are kept. The dashed lines show dropped connections and the gray ones show the kept ones.

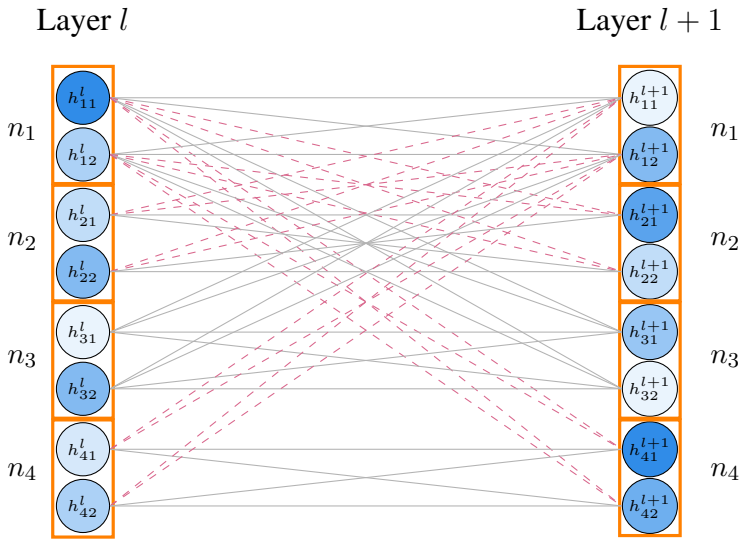

Figure B.4: Schematic of DropEdge [87]. Each circle is a feature and each square represents a node. DropEdge drops edges between nodes hence all of the connections between their corresponding channels are dropped. Note that the mask in DropEdge is symmetric. In this example, the edge between nodes 1 and 2 as well as the edge between nodes 1 and 4 are dropped. The dashed lines show dropped connections and the gray ones show the kept ones.

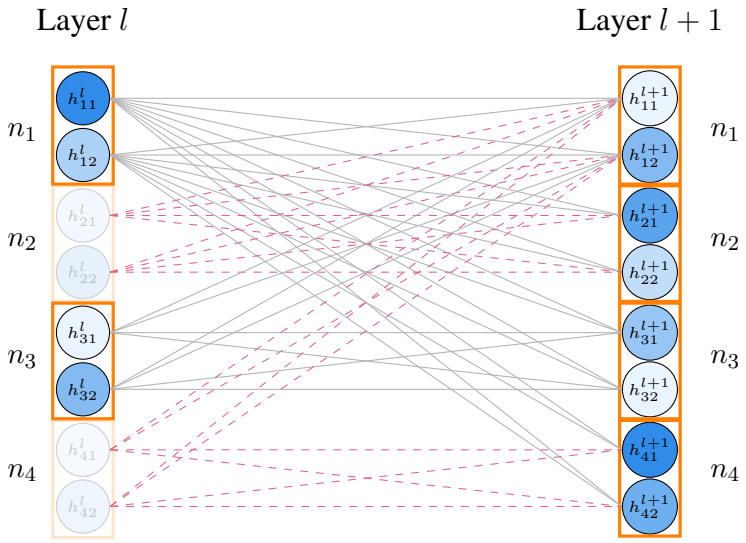

Figure B.5: Schematic of the node sampling strategy in FastGCN [20]. Each circle is a feature and each square represents a node. FastGCN drops nodes hence all of the connections to that node are dropped. The faded nodes represents the dropped nodes. The dashed lines show dropped connections and the gray ones show the kept ones.

APPENDIX C

ADDITIONAL DISCUSSION AND RESULTS FOR MOREL

In this chapter, we first provide detailed description of datasets used in the experiments. Moreover, we provide additional results for MoReL on CF dataset.

C.1 Data description

Microbiome-metabolome interactions. The goal studying this dataset is to detect the microbe-metabolite interactions in patients with Cystic Fibrosis (CF). This dataset includes the 16S ribosomal RNA (rRNA) sequencing and metabolomics for 172 patients diagnosed with CF. We follow the same preprocessing steps as in Morton et al. [71], Hajiramezanali et al. [41], and filter out microbes that appear in less than ten samples, which results in 138 unique microbial taxa and 462 metabolite features. To construct the microbiome network, we perform a taxonomic enrichment analysis using Fisher's test and calculating p-values for each pairs of microbes as in Hajiramezanali et al. [41]. More specifically, the Benjamini-Hochberg procedure [9] is adopted for multiple test correction and an edge is added between two microbes if the adjusted p-value is lower than 0.01, The microbiome graph has 984 edges with the graph density of 0.102. For the metabolomics network, there are 1185 edges in total, with each edge representing a connection between metabolites via a same chemical construction [71]. The graph density of the metabolite network is 0.011. We use 80% of the reported target molecules of *P. aeruginosain* studies in Quinn et al. [83] and Morton et al. [71] as a test set to evaluate the predicted microbiome-metabolome interactions. The remaining 20% of the reported molecules are considered as a validation set and are only used for the early stopping purpose.

Precision medicine. Here we aim to identify genetic markers of cancer drug responses. This is a very challenging task due to the very limited number of observations with respect to the system complexity and huge number of biological and experimental confounders, which often leads to significant false positive associations [6]. We consider a dataset from 30 acute myeloid leukemia (AML) patients that contains gene expression and drug sensitivity data of 160 chemotherapy drugs

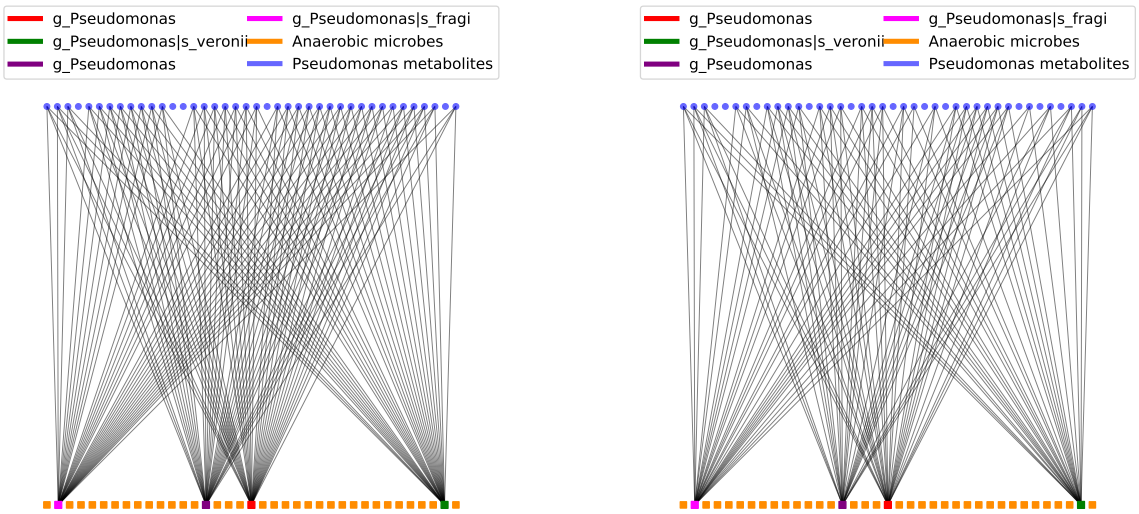

Figure C.1: A sub-network of the relational graph consisting of *P. aeruginosa* microbes, their validated targets, and anaerobic microbes, inferred using MoReL$_{ss}$ (**Left**) and MoReL$_{us}$ (**Right**) with sub-network negative accuracy of 100%.

and targeted inhibitors [59]. For gene expression, we preprocessed the RNA-Seq data resulting in 9073 genes [59]. Following Hajiramezanali et al. [41], we construct the gene regulatory network based on the publicly available expression data of the 14 AML cell lines from the Cancer Cell Line Encyclopedia1 (CCLE) using R package GENIE3 [96]. Moreover, We construct drug-drug interaction networks based on their action mechanisms. Specifically, the selected 53 drugs are categorized into 20 broad pharmacodynamics classes [59]; 14 classes contain more than one drugs. Only 16 out of the 53 drugs are shared across two classes. We consider that two drugs interact if they belong to the same class. We use the area under the drug response curve reported in the CCLE dataset to indicate drug sensitivity across a range of drug concentrations [6, 59]. Following [59], we only consider the drugs that have less than 50% cell viability in at least half of the samples, resulting in 53 drugs. We use 797 reported drug-gene interactions in The Drug–Gene Interaction Database (DGIdb) [101] in order to evaluate different models. We note that our test and validation sets only include the interactions for 43 of the 53 drugs in the dataset. We use 20% of the evaluation set as the validation set. Please note that the validation set has been only used for early stopping.

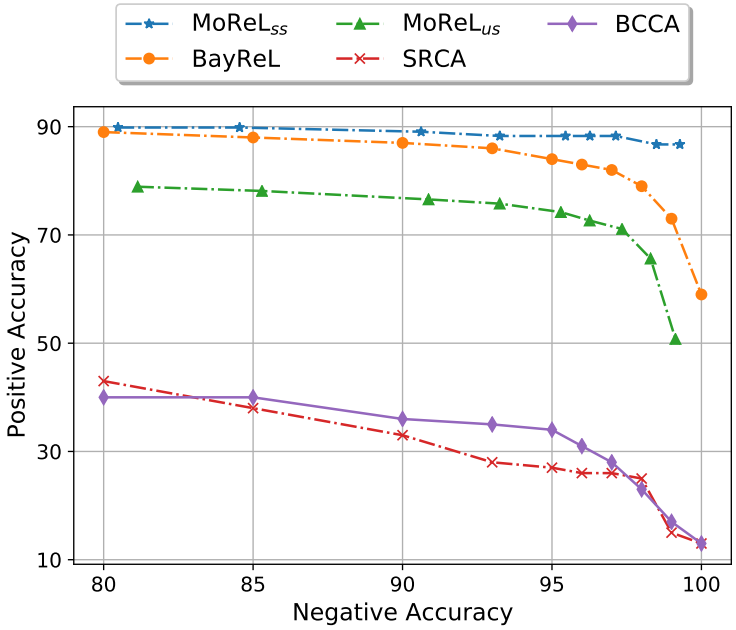

Figure C.2: Positive accuracy vs negative accuracy of various models in CF data.

C.2 Additional results for CF dataset

In this section, we provide additional results for CF dataset demonstrating interpretability and robustness of MoReL.

Figure C.1 shows two sub-networks of the inferred bipartite relational graphs by MoReL$_{ss}$ and MoReL$_{us}$, consisting *P. aeruginosa*, anaerobic microbes, and validated target nodes of *P. aeruginosa* and all of the inferred interactions between them. Based on the biology knowledge, the expected interactions in these sub-networks should be that the four highlighted nodes in the bottom row are connected to all of the nodes in the top row, and any other nodes in the bottom row are not connected to any of the top nodes. At the sub-network negative accuracy of 100% (i.e. any nodes in the bottom row other than the four highlighted ones are not connected to any of the top nodes), while MoReL$_{us}$ identifies 70% of the validated edges of *P. aeruginosa*, MoReL$_{ss}$ identifies 86.8% of the edges. We note that BayReL identifies 78% of the validated interactions [41]. This clearly shows the effectiveness of our proposed FGW-based decoder and interpretability of MoReL to identify inter-relations.

In the main manuscript, we only reported the results for one specific threshold value of negative accuracy (97%). Here we provide additional results with other threshold values, which show similar improvements over competing methods and similar trends by MoReL$_{ss}$ and MoReL$_{us}$, as clearly observed in Figure C.2. We note that there is a trade-off between positive and negative accuracy, and the optimal point can be chosen depending on the application.