Unsupervised Learning of Latent Edge Types from Multi-Relational Data

by

Ankita Sakhuja

B.Tech., Amity University, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Ankita Sakhuja**
Degree:	**Master of Science**
Thesis title:	**Unsupervised Learning of Latent Edge Types from Multi-Relational Data**
Committee:	**Chair:** Manolis Savva Assistant Professor, Computing Science
	Oliver Schulte Supervisor Professor, Computing Science
	Ke Li Committee Member Assistant Professor, Computing Science
	Jiannan Wang Examiner Associate Professor, Computing Science

Abstract

Many relational datasets, including relational databases, feature links of different types (e.g., actors *act in* movies, users *rate* movies), known as multi-relational, heterogeneous, or multi-layer networks. Edge types/network layers are often not explicitly labeled, even when they influence the underlying graph generation process. For example, IMDb lists Tom Cruise as a cast member of Mission Impossible, but not as its star. Inferring latent layers is useful for relational prediction tasks (e.g., predict Tom Cruise's salary or his presence in other movies). This thesis discusses **L**atent **L**ayer **G**enerative **F**ramework - **LLGF**, a generative framework for learning latent layers that generalizes Variational Graph Auto-Encoders (VGAEs) with arbitrary node representation encoders and link generation decoders. The decoder treats the observed edge type signal as a linear combination of latent layer decoders. The encoder infers parallel node representations, one for each latent layer.

We evaluate our proposed framework, LLGF, on eight benchmark graph learning datasets for this study. Four of the datasets are heterogeneous (originally labeled with edge types); we apply LLGF after removing the edge labels to assess how well it recovers ground-truth layers. LLGF increases link prediction accuracy, especially for heterogeneous datasets (up to 5% AUC), and recovers the ground-truth layers exceptionally well.

Keywords: Variational Graph Auto-Encoder, Latent Edge Type, Node Representation

Dedication

This thesis is dedicated to the loving memory of my maternal grandmother and to my friends and family for their endless love, support, and encouragement.

Acknowledgements

I want to thank Dr. Oliver Schulte for his constant encouragement, support, and guidance throughout my entire journey in the master's program.
I am also very thankful to all the professors who taught me and paved this learning path in Computing Science Department.

Table of Contents

List of Tables

List of Figures

Definitions

Notation	Definition
$\mathcal{G}$	graph network
$\mathcal{V}$	set of nodes in a graph
$\mathcal{E}$	set of edges in a graph
N	number of nodes in a graph
X	nodes feature matrix
L'	observed graph layers
L	latent graph layers
$R^{d'}$	dimension of node representation space
A'	adjacency matrix
Z	latent representation matrix of nodes
$\mathbf{z}$	node latent representation vector
m	m$^{\text{th}}$ node in a graph
n	n$^{\text{th}}$ node in a graph
w	shared weights between edges and nodes
Λ_l	block matrix in stochastic block model
W	weight matrix
l	latent layer or latent-edge index in a model
l'	observed layer or observed-edge index in a model
b	bias vector
d'	latent space dimensionality of a node
σ	sigmoid activation function
f	transformation function
$\odot$	element-wise multiplication
$\|$	concatenation
$\boldsymbol{\mu}$	mean vector
$\boldsymbol{\sigma}$	standard deviation vector

Chapter 1

Introduction and Overview

1.1 Definitions

A Graph $\mathcal{G}$ is formulated as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $m \in \mathcal{V}$ and edges $(m, n) \in \mathcal{E}$, where m and n are two nodes that are connected in the graph. These kinds of graphs are also known as undirected graphs, where an edge has no order as this represents a symmetric relation. An example of such a graph is a network of friends on any social media platform, for example - Facebook friends or Linkedin connections, as shown in Figure 1.1.

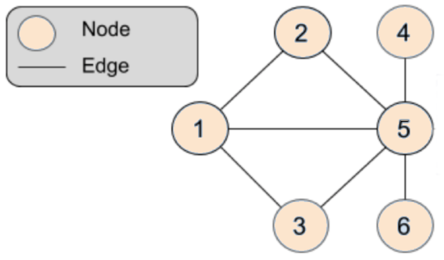


Figure 1.1: Example of an undirected graph with six nodes and seven edges

We can extend this simple graph to a more informative version by adding directions of edges (Directed graphs) or a weighted version of an undirected or directed graph where a positive value of weight indicates strength or weight of the relation. For example, in a protein-protein interaction graph, these weights can be the number of hydrogen bonds linked to the protein molecule; more bonds correspond to a stronger interaction between the two proteins. In the case of a weighted graph network, the formal definition of the edges changes slightly to the following form: $(m, w, n) \in \mathcal{E}$, where w is the weight of the edge.

Beyond the distinction between undirected, directed, and weighted edges, Hamilton [5] discusses graphs with different types of edges. These edge/relation types are represented as $\mathcal{T}$, for example, $(m, \mathcal{T}, n) \in E$, for which one adjacency matrix $A_{\mathcal{T}}$ is defined per edge type. These graphs are called multi-relational graphs in the book, and the entire graph can be summarized by an adjacency tensor $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{R}| \times |\mathcal{V}|}$, where $\mathcal{R}$ is the set of relations.

Two types of these multi-relational graphs are heterogeneous and multi-layer graphs discussed as follows.

Heterogeneous Graphs. Graph nodes and edges can also have a type. A graph with a single type of node carrying a single type of edge is called a homogeneous graph. A graph with two or more types of nodes and/or two or more types of an edge is called a heterogeneous graph. A citation network with edges of different types, say *Author-Conference* and *Paper-Author*, between nodes of *Paper*, *Conference* and *Author* type is an example of a heterogeneous network. See Figure 1.2 for an example.

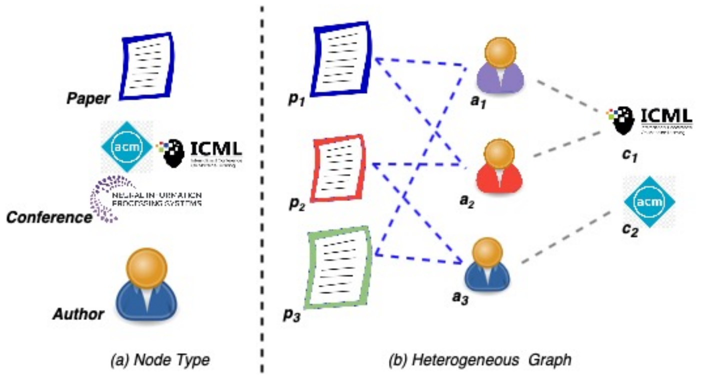


Figure 1.2: An illustrative example of a heterogeneous graph (Citation Network). (a) Three types of nodes (i.e., paper, conference, author). (b) A heterogeneous graph consists of three types of nodes and two types of connections.

Multi-Layer Graphs. In multi-layer graphs, we can decompose a graph in a set of l layers. Every node is assumed to belong to every layer, and each layer corresponds to a unique relation, representing the intra-layer edge type for that layer as discussed in [5]. They also assume that inter-layer edge types can exist, which connect the same node across layers. For instance, in a multi-layer social network, each node represents a user, and each layer might represent a different circle of friendship (e.g., School Friends or Co-workers) as shown in Figure 1.3. All edges between nodes (users) are assumed to interact on the friendship layer. However, the friendships in social network may result from relationships of different origins, such as colleagues in a workplace, members of a club or high school friends.

Multi-Layer Graphs with Latent Layers. The starting point of this thesis is the observation that in many domains, graph data are generated both by observed layers and by unobserved, or latent layers. A **latent-layer graph** is a multi-layer graph with some latent layers. We investigate methods for inferring latent layers from observed layers and observed node features. Our methods are based on the encoder-decoder framework from deep learning.

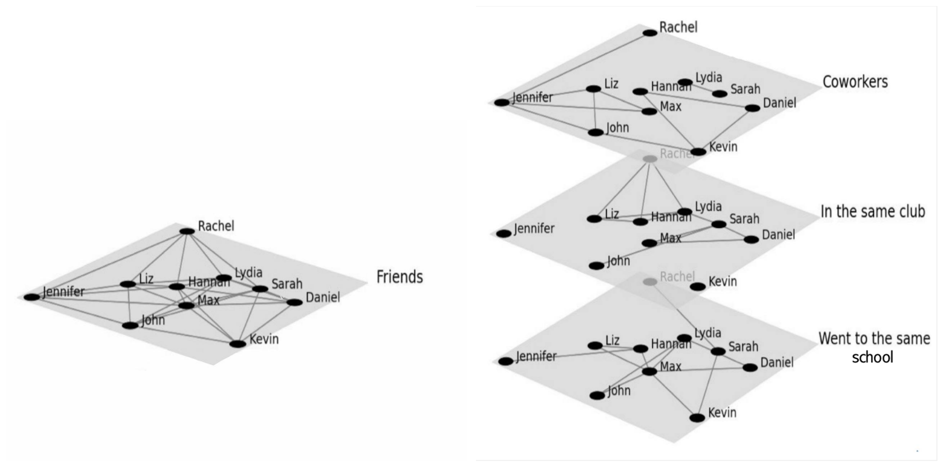


Figure 1.3: A toy example depicting latent layers in the Facebook social network. While there is only one observed layer - friendship, the interactions come from different sources; In this example, three latent layers.

1.2 An Encoder-Decoder Perspective

In the encoder-decoder framework, the graph representation learning problem inculcates two fundamental operations. First, an encoding operation maps each node within the graph to a low-dimensional vector or embedding space. Next, a decoding operation utilizes these low-dimensional node embeddings to reconstruct the neighborhood of each node in the original graph. This idea is summarized in Figure 1.4. Traditionally, this concept was introduced to model image data [7]. Recently, this concept has been incorporated for graph-structured models to map an input graph with adjacency matrix A and feature matrix X to a distribution of z using the function f_θ, which further employs variational approximation function, g_φ to produce the output $\hat{A}$ [8].

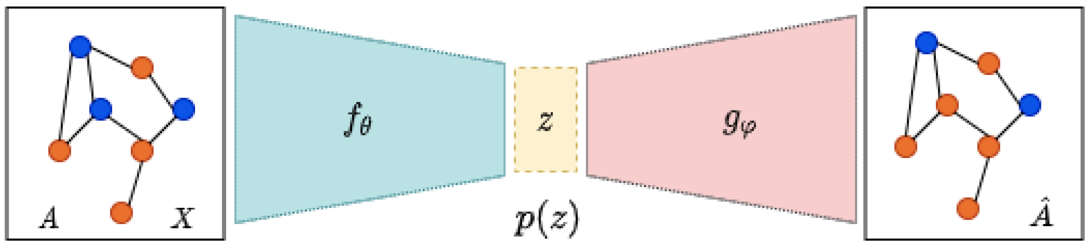


Figure 1.4: Overview of the encoder-decoder approach. The encoder map the nodes present in a graph to a low-dimensional embedding z using the function f_θ. The decoder further utilizes z to reconstruct local neighborhood information of nodes using function g_φ.

1.3 Background

The power of generative models is used in several fields such as image generation, speech synthesis, and many other areas. The datasets used in these kinds of tasks are either structured or unstructured datasets, for example, images and texts. As mentioned in [5], Graphs are a ubiquitous data structure and a universal language for describing complex systems. Recently, Variational Graph Auto-Encoders (VGAEs) have emerged as powerful generative models for graph-structured data, based on inferring latent node representations to predict the presence of a link between two nodes. The node representations support two key tasks [5, Sec.1.2]: 1) link or relation prediction: to predict whether two entities/nodes are related, and 2) node classification: to predict a type or class label for a given node/entity. For example in a social network, link prediction would allow the platform to recommend potential friends to a user. An example of node classification would be to infer a user's political affiliation from that of her friends. Examples of node classification beyond social networks include classifying the function of proteins in the interactomes [5] or classifying the topic of documents based on hyperlink or citation graphs [9]. According to Hamilton, link prediction has "countless real-world applications: recommending content to users in social platforms [29], predicting drug side-effects [31], or inferring new facts in a relational databases [1]-all of these tasks can be viewed as special cases of relation prediction."
Existing models [8, 15, 4] rely on Graph Neural Networks (GNN), in particular, Message Passing GNN's which update node representations by iterative message passing between neighboring nodes.
Most VGAEs are designed for homogeneous networks with a single type of edge. Recent work has shown that when the data provides edge labels, message-passing models can improve node representations by incorporating edge type information [20, 22, 16]. Network science has shown that many complex systems are composed of coupled graphs with different layers, each representing an edge type with a diverse community structure [27]. Considering multi-layer graph structure can improve the generative modeling performance. However, in many cases, the layers are latent and cannot be directly used with existing multi-layer methods [13]. For example, IMDb lists Tom Cruise as a cast member of Mission Impossible, but not as its star. To the best of our knowledge, no previous study has explored the possibility of utilizing latent graph layers to improve generative model capacity.

1.4 Overview

1.4.1 Approach

In this thesis, we propose LLGF, a new variational graph learning approach that unifies several existing methods and is capable of modeling both observed and latent graph layers. In particular, if we use L' and L to denote the number of observed and latent layers respectively, setting $L = 1$ is equivalent to a multi-relational VGAEs, and setting $L' = 1$ additionally is equivalent to VGAEs for homogeneous graphs. Whereas prior approaches do not model latent layers, we introduce a new formulation that accounts for them and demonstrates its advantages along two dimensions: improved accuracy on link prediction on observed layers and enhanced quality of edge embeddings.

The key idea is to view latent layers as constituent factors from which observed layers are generated. We leverage Message-Passing Graph Models, such as Relational Graph Convolutional Networks (RGCN), to infer L parallel node representations, which are aggregated into a final bottleneck node representation. Given two input node representations, the decoder outputs $L + L'$ scores, one for each latent and observed layer, where observed layer scores are modeled as a linear combination of latent layer scores. Each observed layer score is then mapped to a link probability for the input nodes.

1.4.2 Evaluation

We evaluate LLGF on eight benchmark graph learning datasets. Four of the datasets are heterogeneous (originally labeled with layers). By withholding the edge labels, we obtain ground-truth layer labels for a homogeneous dataset. For predictive performance, LLGF outperforms some of the leading Latent Variable Deep Graph Generative models, referred as VGAEs in this thesis. The improvement is most significant for the (originally) heterogeneous graphs (up to 5% AUC), which shows the usefulness of exploiting latent layers when a multi-layer structure underlies the data. On the heterogeneous datasets, we show that the proposed model can recover the ground-truth latent layers. This result is based on a novel L-dimensional latent representation for a dyad (pair) of nodes comprising the L decoders scores for the dyad. Our experiments provide evidence that this dyad representation captures the latent multi-layer structure better than a simple baseline concatenating the dyad's separate node representations. LLGF offers a new approach to inferring dyad representations, which can be used for downstream tasks like edge classification, without the need to store dyad representations during message-passing.

1.4.3 Contributions

The contributions of this thesis can be summarized as follows.

- A new variational deep graph learning framework that generalizes multi-relational graph learning to infer latent graph layers (edge types).
- A new latent layer decoder structure that decomposes observed layer signals into latent layer sources.
- A generalization of multi-relational graph convolutional neural networks to latent layer models.

Chapter 2

Related Work

To the best of our knowledge, the inference of latent edge types is a new topic in deep graph learning. While there are various approaches to generative modelling for neural networks (e.g. based on a GAN model [25]), most works use a variational graph auto-encoder (VGAEs) framework [9, 4, 23, 19, 17, 18]. In this thesis, we develop latent layer learning for VGAEs and leave other generative models for future work.
VGAEs apply the Variational Auto-Encoder (VAE) [7] with an encoder (inference) + decoder (generative) architecture. They employ high-capacity function approximators, typically Graph Neural Networks (GNNs) [28]. The encoder outputs a low-dimensional latent representation for each node in the graph. The decoder is a deterministic function with signature $DEC : R^{d'} \times R^{d'} \rightarrow R$ signature to reconstruct the graph edges and/or the node/edge attributes. Here $R^{d'}$ is the dimension of the node representation space.

Node Representation Learning with Multi-layer VGAEs Most previous VGAEs models were applied to graphs with a single layer (edge type). A few recent models have shown the power of incorporating observed layers. R-VGAE [12] is VGAEs architecture that uses a Relational GCN [20] as encoder. The decoder is a Stochastic Block Model (SBM) that performs link prediction but does not reconstruct the latent layers separately. O2MAC [3], proposes multi-view graph decoder function. While O2MAC uses GCN[9] as an encoder, it extends SBM to a multi-layer decoder by learning different block matrices for each relation. Although multi-layer SBM as the decoder can reconstruct observed layers of a graph, it is limited in terms of the model capacity and learning power. The differences to our model are as follows.

- We utilize a multi-layer RGCN encoder like R-VGAE, but also a multi-layer decoder.
- O2MAC does not incorporate observed layers in the encoder. Also, we add latent-layer specific transformation to the decoder, which substantially increases its expressive power. We believe that our LLGF, Latent Layer Generative Framework, is the first

Deep Graph Generative model that utilizes both observed and latent layers in both the encoder and decoder.

In addition to the multi-layer VGAEs models, other recent GNN models have shown the power of incorporating observed edge types [26, 6, 20, 14, 16, 30]. Hamilton discussed on an alternative route [5] called the parameter sharing Relational-GCN. Here observed layers are represented as linear combinations of a fixed set of basis adjacency matrices. The work in this thesis can be seen as learning an adjacency matrix basis represented by a set of latent layers.

Chapter 3

Method and Approach

We define a graph with different types of edges to be a **multi-layer graph** in which a layer is defined by an edge type as discussed in Chapter 1. While the node-set is shared, each layer of a multi-layer graph contains different interactions[1]. Mathematically, such graph is a pair of $|V| = N$ nodes and edge set E. The edge set is divided into layers $E = \bigcup\{E_1, ..., E_{L'}\}$, where $E_{l'}$ contains the edges in observed layer $l^{'}$. A binary adjacency matrix $\mathbf{A}^{'}_{l'} \in \{0, 1\}^{N\times N}$ represents the $l^{'}$-th observed layer, such that $\mathbf{A}^{'}_{l'mn} = 1$ if layer $l^{'}$ contains an edge between nodes m and n. The notations L and $L^{'}$ indicate the number of latent and observed layers respectively. Typically $L' < L$.

3.1 Model Definition

3.1.1 Learning

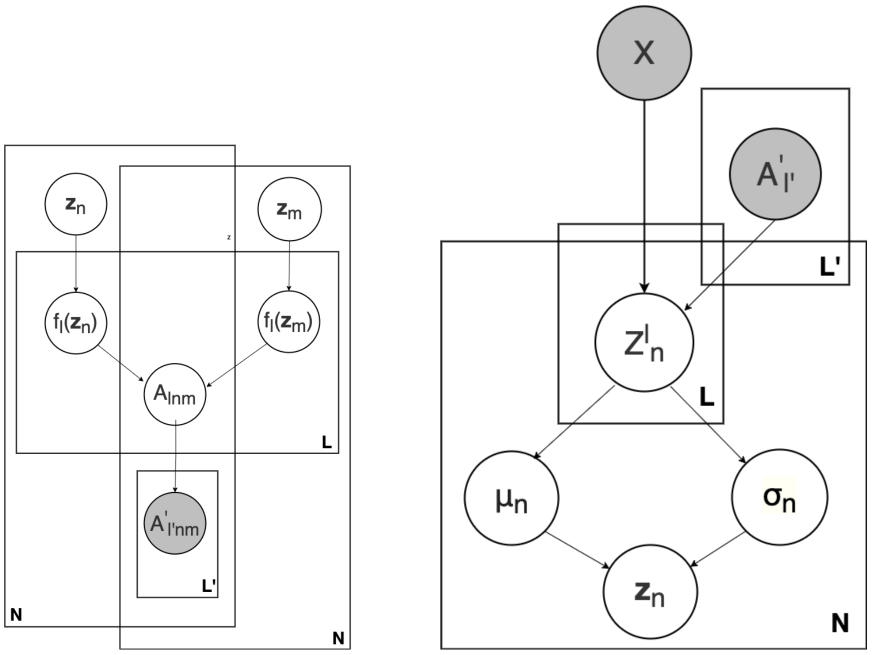
The training objective of VGAEs is usually the variational lower bound:

$$\mathcal{L}(X, \mathbf{A}^{'}) = E_{q(Z|X,\mathbf{A}')}[\log p(\mathbf{A}^{'}|Z)] - KL[q(Z|X, \mathbf{A}^{'}), p(Z)] \tag{3.1}$$

where $KL[.,.]$ is Kullback-Leibler divergence, Z is the nodes latent representation matrix, X is the nodes feature matrix, $p(Z)$ is prior distribution of Z, and $q(Z|X, \mathbf{A}^{'})$ and $p(\mathbf{A}^{'}|Z)$ are variational posterior and generative distribution respectively. The prior distribution $p(Z)$ is usually taken as a standard Gaussian distribution. The approximate posterior $q(Z|X, \mathbf{A}^{'})$ is computed by a high-capacity model (the encoder or inference model). The reconstruction likelihood $p(\mathbf{A}^{'}|Z)$ is computed by another high-capacity model (the decoder or generative model).

The LLGF plate diagram along with its generative and inference architecture is illustrated in Figure 3.1, 3.2 and 3.3 respectively. The next two sections describe LLGF's Generative and Inference model.

[1]This definition is often associated with **multiplex** in Network Theory literature.



(a) (b)

Figure 3.1: The plate diagram of *LLGF*. (Left) The generative model/decoder reconstructs L' observed layer by first generating L latent layers and then integrating them. (Right) The inference model/encoder, given the adjacency matrix with observed layers $\mathbf{A}'$ and node features X, learns variational posterior by applying L message-passing NN to infer and utilize the latent interactions. Here we assume isotropic Gaussian as a prior.

3.1.2 Generative/Decoder Model

The input to the decoder model is a matrix $\mathbf{Z}_{N\times d'}$ where d' is the dimension of the node latent representations $\mathbf{z}_m$ where $m = 1, \ldots, N$. The generative model consists of three blocks.

The Transformation Block reconstructs latent layers. It comprises L deterministic transformation of node latent representations. The notation $f_l(\mathbf{z}_m) \in R^d$ denotes the result of applying the l-th transformation function to the latent representation of node m. Each transformation function $f_l()$ is implemented by a neural network.

The Reconstruction Block comprises L decoder functions denoted by DEC_l, one for each latent layer, with signature $DEC_l : R^d \times R^d \to R$. Each decoder $DEC_l(f_l(\mathbf{z}_m), f_l(\mathbf{z}_n))$ returns a real number for two nodes and a latent layer. Any standard multi-relational decoder can be utilized; for an overview see [5, Sec.4.3].

The Aggregation Block reconstructs observed layers by aggregating reconstructed latent layers. It implements a function with signature $AGG : R^L \to [0, 1]^{L'}$. This function

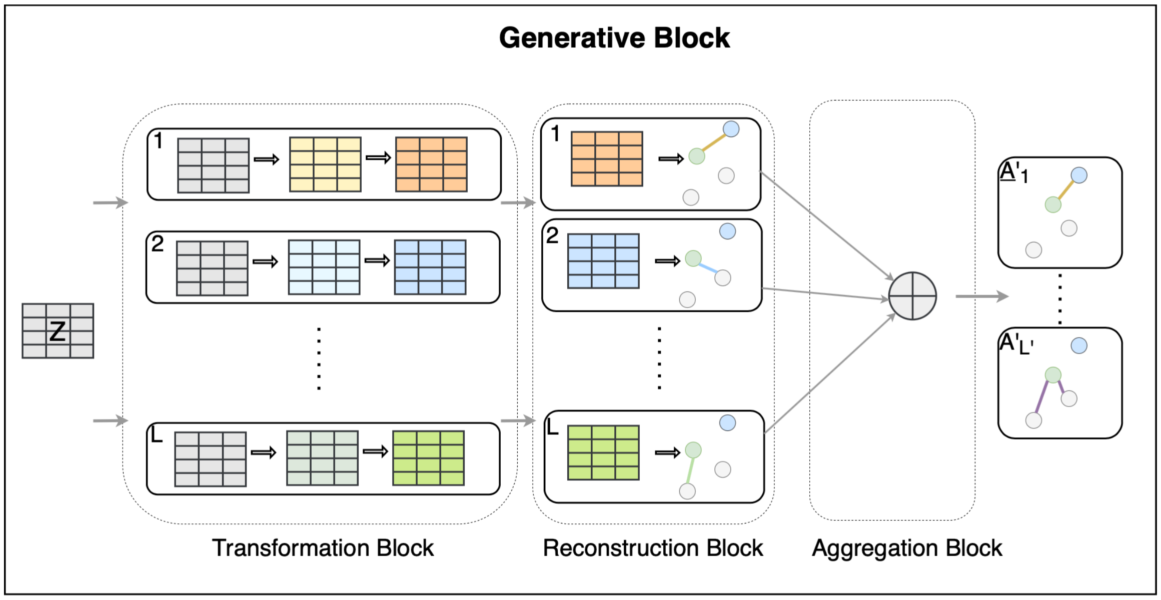


Figure 3.2: An illustration of *LLGF* generative architecture. The generative model/decoder retrieves observed layers of the graph by aggregating the generated latent layers. This architecture consists of three blocks. 1) Transformation Block: Given a node m representation, $\mathbf{z}_m$, this block returns this representation *w.r.t.* each of the latent layers as $f_l(\mathbf{z}_m)$ where $0 \leq l \leq L$. The l-th node representation defines node interactions in the l-th latent layer. 2) Reconstruction Block: Reconstructs the l-th latent layer given the l-th transformation of node representation, formulated as $f_l(Z)$. 3) Aggregation Block: This block builds the observed layers, $A'_{l'}$ where $0 \leq l' \leq L'$, given the generated latent layers.

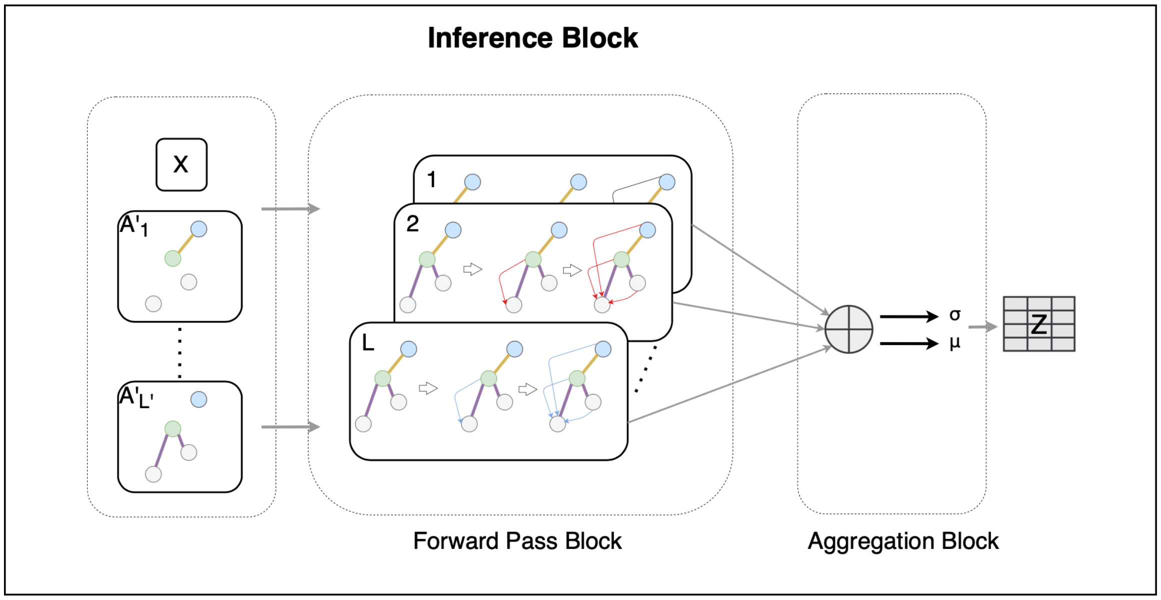


Figure 3.3: An illustration of *LLGF* inference architecture. The inference model learns the variational posterior distribution by utilizing (message-passing) neural nets. The inference model/encoder is designed to consider both observed and latent layers.

returns the probability of having an edge in observed layer l', given the interactions of the nodes in the latent layers. In our design, the edge reconstruction decoder is a mixture of the latent-layer decoders:

$$AGG_{l'}(\mathbf{z}_m, \mathbf{z}_n) = \sum_{l=1}^{L} w_{l'l} DEC_l(f_l(\mathbf{z}_m), f_l(\mathbf{z}_n)) \tag{3.2}$$

$$p(\mathbf{A}'_{l'mn} = 1 | \mathbf{z}_m, \mathbf{z}_n) = \sigma(AGG_{l'}(\mathbf{z}_m, \mathbf{z}_n)) \tag{3.3}$$

where $w_{l'l}$ are trainable weights and $\sigma(.)$ denotes the sigmoid activation function.

Our design enforces the interpretation of the latent layers as components of observed layers. The $w_{l'l}$ weights, which are shared across nodes and links, represent how much of each latent component is present in the observed layer at the graph level. To encourage sparsity, it is possible to add L1-penalty on the $w_{l'l}$ weights to the ELBO objective in equation (3.1). Sparse weights can be interpreted as associating different latent layers to different observed layers. We leave adding a sparsity term for future work; as we report in Section 4.6.4, we observe sparse weights even without an explicit penalty term.

The $f_l()$ transformations generate node representations specific to a latent layer. For example, $f_l(\mathbf{z}_m)$ may represent the community memberships of node m in latent layer l, whereas $f_{l^*}(\mathbf{z}_m)$ may represent community memberships of the same node in latent layer l^*. The $f_l()$ neural networks increase the model capacity, since standard multi-relational decoders DEC_l use a fixed parametric form.

Example Using non-linear node transformation function $f(.)$, SBM as a decoder function, i.e $DEC_l(f_l(\mathbf{z}_m), f_l(\mathbf{z}_n)) = f_l(\mathbf{z}_m)^\intercal \Lambda_l f_l(\mathbf{z}_n)$, and linear aggregation function, the generative model will be:

$$p(\mathbf{A}'_{l'mn} = 1 | \mathbf{z}_m, \mathbf{z}_n) = \sigma(\sum_{l=1}^{L} w_{l'l}(f_l(\mathbf{z}_m)^\intercal \Lambda_l f_l(\mathbf{z}_n))) \text{ for } l^{'} \in \{1..L^{'}\} \tag{3.4}$$

where $\Lambda_l \in R^{d \times d}$, $|f_l(\mathbf{z})| = d$, is the block matrix in the Stochastic Block Models [11]. Each entry Λ_{lty} controls the probability of an edge in latent layer l when node n and node m are assigned to cluster t and cluster y respectively, according to their l-representations $f_l(\mathbf{z}_m)$ and $f_l(\mathbf{z}_n)$.

3.1.3 Inference/Encoder Model

Several Graph Neural Networks (GNNs) have been proposed for graph data with single or multiple observed layers, such as the Relational Graph Convolutional Network (RGCN) [20]. To leverage existing GNN encoders for inferring latent layers, we use L parallel inference models to produce L node representation matrices Z^l. The final encoding step aggregates the latent-layer node representations into a single one. The proposed encoder comprises two main blocks.

The **Forward Pass Block** includes L parallel propagating models for calculating forward-pass update of each node regarding a latent layer in a graph. Each model is chosen to be a message-passing neural network. The inputs of each model are the observed layers of the graph, $\mathbf{A}^{'} = \{\mathbf{A}^{'}_1, \mathbf{A}^{'}_2, ..., \mathbf{A}^{'}_{L'}\}$, and the node feature matrix X. Model l returns a node representation matrix for the l-th latent layer, using message passing among the observed layers.

Example Using the two-layer version of Relational Graph Convolutional Networks (RGCN) [20], and $Tanh$ activation function, the l-th propagating model will be:

$$LRGCN(\mathbf{A}^{'}, X) = Z^l = Tanh(\sum_{l''=1}^{L'} \mathbf{A}^{'}_{l''}(Tanh(\sum_{l'=1}^{L'} \mathbf{A}^{'}_{l'} X W^0_{ll'}))W^1_{l''}) \text{ for } l \in \{1..L\}. \quad (3.5)$$

This expression can be read as follows. 1) The matrices $W^0_{ll'}$ linearly transform the input features in observed layer l' to a node representation for inferring the l-th latent layer. 2) For each node, the transformed features of its l'-neighbors are aggregated. 3) The latent l-representations are aggregated across observed layers $l^{'}$, with a final non-linearity. These aggregation operations are repeated based on another set of weight matrices W^1. Thus the $W_{ll'}$ generalize the multi-relational GCN approach of introducing a separate weight matrix per observed layer.

For another example, generalizing the single layer Graph Attention Networks [2] with $ReLU$ activation function leads to l-th node representation:

$$LGA(\mathbf{A}^{'}, X) = Z^l = ReLU(\sum_{l'=1}^{L'} (\mathbf{A}^{'}_{l'} \odot G_{ll'}) X W_{ll'}) \quad (3.6)$$

$$G_{ll'mn} = F_{ll'}(\mathbf{x}_m \| \mathbf{x}_n) \quad \text{for} \quad l \in \{1..L\}. \quad (3.7)$$

Here $F_{ll'}()$ represents learnable attention weights computed by a neural network. The symbols $\odot$ and $\|$ denote element-wise multiplication and concatenation respectively.

The Aggregation Block aggregates the L node representations to compute the parameters of the variational posterior distribution parameters.

Example In our experiments we assume a posterior Gaussian distribution $q(\mathbf{z}_m|X, \mathbf{A}^{'}) = \mathcal{N}(\mathbf{z}_m|\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m^2)$, and weighted linear aggregation function:

$$\boldsymbol{\mu}_m = \sum_{l=1}^{L} \mathbf{z}_m^l W_{\mu l} \tag{3.8}$$

$$log\ \boldsymbol{\sigma}_m = \sum_{l=1}^{L} \mathbf{z}_m^l W_{\sigma l}, \tag{3.9}$$

where

- $\boldsymbol{\mu}_m$ and $\boldsymbol{\sigma}_m$ are mean and standard deviation vectors for the node m posterior,
- $\mathbf{z}_m^l$ is the node m representation for latent layer l,
- and $W_{\mu l}$ and $W_{\sigma l}$ are $|\mathbf{z}_m^l| \times d^{'}$ learnable matrices.

3.2 LLGF as a General Framework and its Special Cases

We can deploy existing benchmarks for the graph-structured data as special cases of our model. We believe the proposed model can be considered as a generalized Variational Deep Graph Generative Framework, and previous VGAEs can be used as special cases of it. Below, we show how some of the popular VGAEs can be derived from the proposed model as an example.

VGAE. For this setting, we ignore that there are latent layers existing in the graph by setting $L = 1$ and stick to the following setting.

1. *Propagation Model* as a single layer GCN,
2. Encoder's *Aggregation Block* formulated as equations 4.1 and 4.2,
3. Inner product normalized by sigmoid function for *Reconstruction Block*,
4. Identity function as decoder's *Aggregation Block*, then we recover the VGAE model [8].

DGLFRM. We apply the same encoder setting we used for VGAE combined with

1. Adding stick-breaking process prior,
2. Two layer Neural Net as the node transformation model,
3. Normalized inner product as *Reconstruction Block*,
4. Identity function for decoder's *Aggregation Block* to recover DGLFRM model [15].

R-VGAE. We set $L = 1$ ignoring the latent layer existing in the graph with the following setting.

1. A 2-layer RGCN as a *propagating model*,
2. Equations 4.1 and 4.2 as the encoder's *Aggregation Block*,
3. Identity function for decoder's *Aggregation Block* with node transformation model,
4. SBM as *Reconstruction Block*,
5. Identity function for decoder's *Aggregation Block* to recover R-VGAE model [12].

Chapter 4

Experiments and Results

4.1 Datasets

Based on the composition of graphs, we categorize our datasets into two categories, homogeneous and heterogeneous networks. The statistics of the datasets used for our experiments are shown in Table 4.1. Homogeneous graphs includes **Les Misérables**, **Citeseer**, **Cora**, and **Pubmed** [10][21]. This study treats links present in these datasets as undirected edges. Heterogeneous graphs are **IMDb**, **DBLP**, **ACM** and **IMDb - F** [30] which all consist of three different types of nodes and two types of undirected edges. In all the experiments, the edge types are hidden, and a symmetric adjacency matrix A', is given to the models, including LLGF.

Table 4.1: Statistics of the Datasets.

Dataset	Nodes	Edges	#Node Type	#Edge Type
Les Misérables	77	254	1	1
Citeseer	3,327	4,732	1	1
Cora	2,708	5,429	1	1
Pubmed	19,717	44,338	1	1
IMDb	12,772	19,120	3	2
DBLP	18,405	47,283	3	2
ACM	8,993	18,929	3	2
IMDb-F	52,234	70,623	3	2

1. Homogeneous graphs

   Citation networks. We consider three citation network datasets: Citeseer, Cora and Pubmed[21]. The dataset contains paper as nodes and citations as edges. This thesis treats the citation links as (undirected) edges and constructs a binary, symmetric adjacency matrix similar to the methodology followed in [8]. The text attributes in the papers correspond to node-level features in the graph.

Les Misérables. Here we use an undirected network of co-occurrences of characters in the Victor Hugos classic novel *Les Misérables* [10] where the network nodes represent the characters, and the edges represent whether two characters appeared in the same chapter of the novel or not.

2. Heterogeneous graphs
For the experiments present in this study, heterogeneous data is converted to a sparse matrix for efficient arithmetic operations.

IMDb. Here we extract a subset of IMDb, which contains movies, actors, and directors. The edges contain interactions between movies and actors, and, movies and directors.

DBLP. We use a subset of DBLP, which contains three types of nodes: papers, authors, and conferences.

ACM. We utilize papers published in KDD, SIGMOD, SIGCOMM, MobiCOMM, and VLDB. We construct a heterogeneous graph that comprises papers, authors, and subjects.

IMDb-F. This version of IMDb is an abundant version where we extract a network with a larger number of movies, actors, and directors.

4.2 Data-Preprocessing

For each originally heterogeneous dataset with N number of nodes, we define a $N \times N$ adjacency matrix A' such that its element $A'_{mn} = 1$ if there is an edge from node m to node n and 0 otherwise.
We also define a node feature matrix X, which is of size N by the number of features for each node.

4.3 Baselines

We compare the proposed model against four baselines:

VGAE. Comprises two-layer GCN as encoding procedure. The decoder has no learnable parameters and the reconstruction is done through an inner product operation [8].

VGAE-SBM. This is a variation of the model discussed above. The decoder in this baseline is replaced with SBM to add learnable parameters for reconstruction.

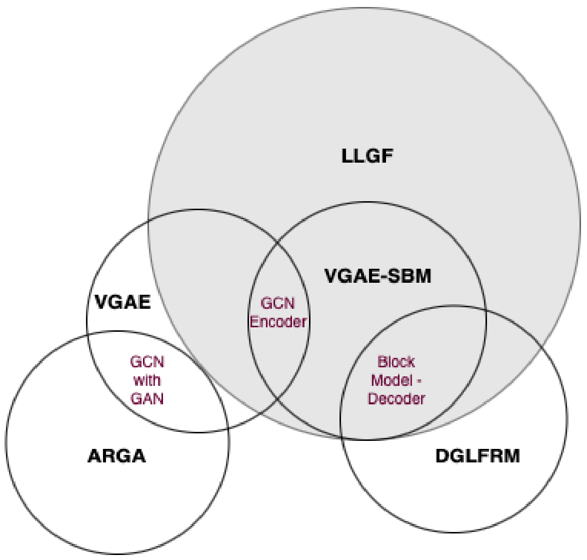


Figure 4.1: Venn diagram showing the overlapping characteristics of each baseline. Each circle represents the baselines (ARGA, VGAE, VGAE-SBM, and DGLFRM) alongside the proposed model - LLGF. Each intersection contains the features common amongst baselines. Refer to section 3.2 to see how we can extend our framework to existing models

ARGA. Proposes an adversarial training scheme to regularize a GCN-based graph auto-encoder and enforces it to match a prior distribution [17].

DGLFRM. Combines stochastic block model with a deep generative model to generate latent community structures and links within a graph. The inference/encoder model is a graph convolutional network similar to the VGAE model. However, this model applies a non-parametric before learning the number of latent communities (node embedding dimensionality) for each dataset. This model also employs a nonlinear learnable transformation in the decoder [15].

Figure 4.1 provides a visual understanding of baselines used in this thesis, illustrating the shared features amongst them using a Venn diagram.

4.4 Experimental Setup

4.4.1 LLGF Setup

In all experiments, we use the following architecture to implement our model.

Inference Model We use two layer GCN as the propagating model in the Forward Pass Block. We also assume isotropic Gaussian prior with parameters

$$\mu = GCN_{\mu}(\mathbf{A}', \|_{l=1}^{L} Z^{l}) \text{ and } \tag{4.1}$$

$$log\ \sigma = GCN_{\sigma}(\mathbf{A}', \|_{l=1}^{L} Z^l). \tag{4.2}$$

Generative Model We use two layer fully connected neural network with $Tanh$ activation function as the nodes transformation function, i.e

$$f_l(\mathbf{z}_m) = Tanh(Tanh(\mathbf{z}_m W_{l0} + \mathbf{b}_{l0}))W_{l1} + \mathbf{b}_{l1}), \tag{4.3}$$

We also use simple Inner Dot Product for Reconstruction Block and linear aggregation function, see equations 4.4 and 4.5.

$$p(\mathbf{A}_{lmn}|\mathbf{z}_m, \mathbf{z}_n) = f_l(\mathbf{z}_m)^{\intercal} f_l(\mathbf{z}_n) \tag{4.4}$$

$$p(\mathbf{A}'|\mathbf{A}_1, \ldots, \mathbf{A}_L) = \sigma(\sum_{l=1}^{L} \mathbf{A}_l) \tag{4.5}$$

Note Ws and **b**s are learning parameters. We call this setting of our model as **LLGF-Dot** in this thesis. The first two GCNs in encoders have dimensions 32 and 64, respectively, and each node is mapped to a 64-dimensional latent space. The non-linear node transformations in the decoder have two layers with dimensions 32. We use $L \in \{1, 2, 6, 4, 8, 10\}$ in all the experiments . Instead of a greedy search, we start with $L = 1$ and increase it while AUC increases on the validation set. The LLGF-Dot is trained using Adam optimizer with a learning rate of 0.001, and 0.3 dropout.

4.4.2 Baselines Setup

For all the baselines, we use 200 as the number of epochs. For all other hyper-parameters, we apply the same settings as reported in their original paper. We also use the same split for a better evaluation for training, validation, and test in all experiments.

4.4.3 Results on Link Prediction

We compare models based on their ability to learn meaningful latent node representations and predict missing edges. In link prediction task, we are given a graph with a fraction of edges removed, and we would like to predict these missing edges. Similar to [8, 15, 17], we hold out 10% and 5% of the edges and the same number of non-existent edges as test and validation set, respectively. We use the validation set to fine-tune the hyper-parameters and report the results on the test set. We evaluate model performance based on their Area Under Curve (AUC) and Average Precision (AP) metrics. The results are shown in Table 4.2 and 4.3.

Table 4.2: Link prediction task in different homogeneous networks. See Table 4.1 for dataset details.

Method	Les Misérables		Citeseer		Cora		Pubmed	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
VGAE [8]	91.2 ± 0.01	91.5 ± 0.01	90.5 ± 0.01	90.06 ± 0.04	90.96 ± 0.00	92.79 ± 0.02	90.47 ± 0.01	91.01 ± 0.00
VGAE-SBM	91.7 ± 0.03	91.81 ± 0.04	90.67 ± 0.03	90.81 ± 0.04	91.25 ± 0.06	92.44 ± 0.05	91.01 ± 0.03	91.21 ± 0.01
DGLFRM [15]	91.4 ± 0.02	91.61 ± 0.03	92.05 ± 0.01	92.62 ± 0.04	92.28 ± 0.02	93.14 ± 0.02	94.21 ± 0.01	94.16 ± 0.01
ARGA [17]	92.50 ± 0.02	92.73 ± 0.01	91.40 ± 0.02	91.79 ± 0.01	92.52 ± 0.01	92.99 ± 0.00	90.36 ± 0.04	91.02 ± 0.01
LLGF-Dot(Ours)	$\mathbf{95.12} \pm 0.01$	$\mathbf{95.8} \pm 0.02$	$\mathbf{94.59} \pm 0.01$	$\mathbf{94.08} \pm 0.01$	$\mathbf{94.56} \pm 0.01$	$\mathbf{94.51} \pm 0.00$	$\mathbf{96.89} \pm 0.00$	$\mathbf{96.97} \pm 0.01$

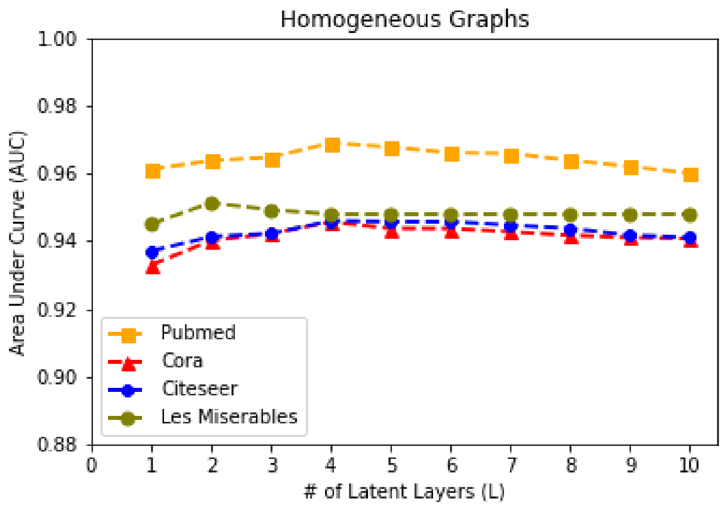
Table 4.3: Link prediction task in different heterogeneous networks. See Table 4.1 for dataset details.

Method	IMDb		DBLP		ACM		IMDb-F	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
VGAE[8]	84.47 ± 0.02	85.87 ± 0.01	91.90 ± 0.00	92.00 ± 0.01	83.00 ± 0.02	84.00 ± 0.03	82.5 ± 0.02	82.7 ± 0.02
VGAE-SBM	76.55 ± 0.00	76.02 ± 0.00	90.02 ± 0.00	91.90 ± 0.01	92.48 ± 0.01	94.62 ± 0.00	77.25 ± 0.00	77.5 ± 0.00
DGLFRM[15]	86.16 ± 0.03	86.68 ± 0.02	91.96 ± 0.01	93.33 ± 0.01	90.35 ± 0.01	91.02 ± 0.01	84.45 ± 0.03	84.6 ± 0.02
ARGA[17]	82.49 ± 0.01	83.14 ± 0.02	91.55 ± 0.00	92.96 ± 0.00	92.19 ± 0.02	90.18 ± 0.01	84.05 ± 0.03	84.2 ± 0.01
LLGF-Dot(Ours)	$\mathbf{90.25} \pm 0.01$	$\mathbf{91.97} \pm 0.01$	$\mathbf{93.17} \pm 0.02$	$\mathbf{94.04} \pm 0.03$	$\mathbf{96.94} \pm 0.01$	$\mathbf{97.61} \pm 0.02$	$\mathbf{91.1} \pm 0.02$	$\mathbf{91.34} \pm 0.03$

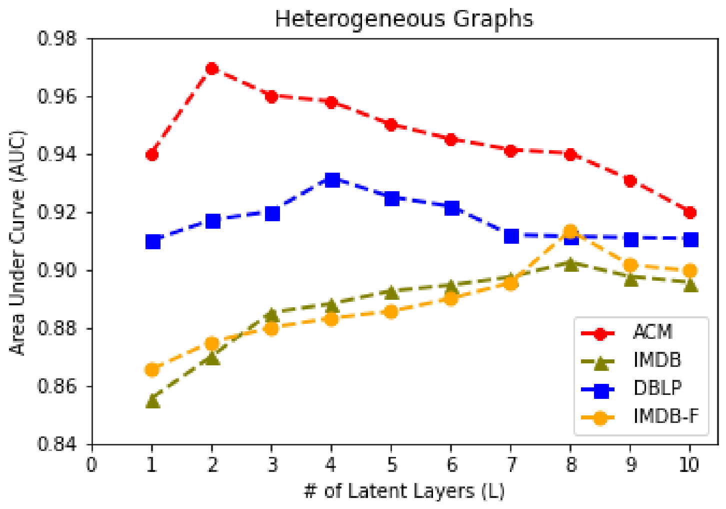
As shown in the tables, LLGF-Dot outperforms the baselines on all datasets. Although we have a fine improvement on the homogeneous graphs, on the heterogeneous graphs with hidden latent layers, we gain up to 5.5% improvement, which justifies the importance of considering the graph multi-layer structure in designing the deep generative models. We will discuss the effects of the latent layer in the proposed model performance in more detail in section 4.5. Note that we set L as discussed in section 4.4.1 and report the result of the final settings. We use $L = 4$ for all homogeneous datasets. For ACM, we have $L = 2$ whereas, in IMDb, DBLP, and IMDb-F, L is set to 8, 6, and 8 respectively.

4.5 Sensitivity to Hyper-Parameters

The proposed model involves an important hyper-parameter L, which defines the number of components to infer the interactions of the latent layers. Recall that $L = 1$ is equivalent to the model ignoring any latent layers present in the dataset. In Figure 4.2a and 4.2b, we examine how this parameter affects the performance of the proposed model on the homogeneous and heterogeneous datasets, respectively. As illustrated, the sensitivity of the proposed model to parameter L is not as noticeable for the homogeneous datasets compared to the heterogeneous datasets. The model performance gradually increases from $L = 1$ to 4 and then gradually decreases for homogeneous datasets such as Citeseer, Les Misérables, Cora, and Pubmed. For the datasets, IMDb, DBLP, ACM, and IMDb-F that we know to comprise latent layer structure, the model Area Under Curve (AUC) performance is strongly sensitive to the parameter L.



(a)



(b)

Figure 4.2: LLGF-Dot sensitivity to L. Figure 4.2a and 4.2b shows the effect of L on the model for homogeneous and heterogeneous datasets. The proposed model's sensitivity to L for the homogeneous datasets is not as noticeable as it is for the heterogeneous datasets. As expected, this parameter results in an increase in performance by introducing latent layers.

Overall, we observe that incorporating latent layers improves link prediction in all datasets, but the extent of the improvement depends on the dataset's characteristics.

We also analyze the ability of a baseline model to utilize hidden graph structure by adding learning parameters. We increase the number of model parameters by adding dimensions (latent communities) to the node latent space d'. Figure 4.3 shows the sensitivity to d' of DGLFRM (the model with the second-best performance in link prediction). Expanding d' from 32 to 128 barely makes more than 1% difference in AUC. This is evidence that our AUC improvements are due to modeling latent layers specifically, not simply by adding parameters to graph VAE models.

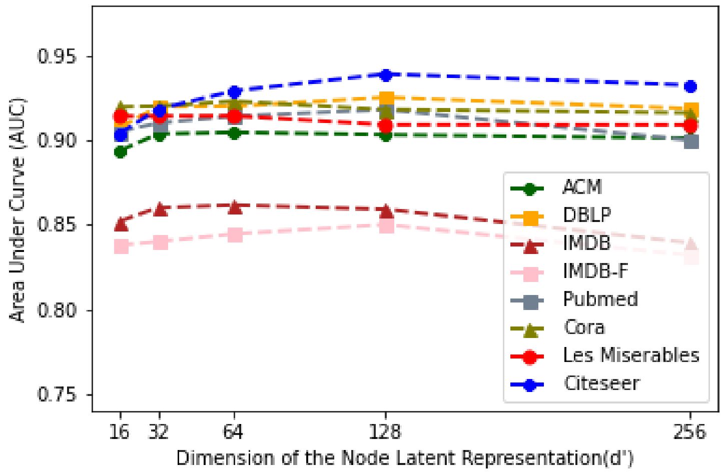


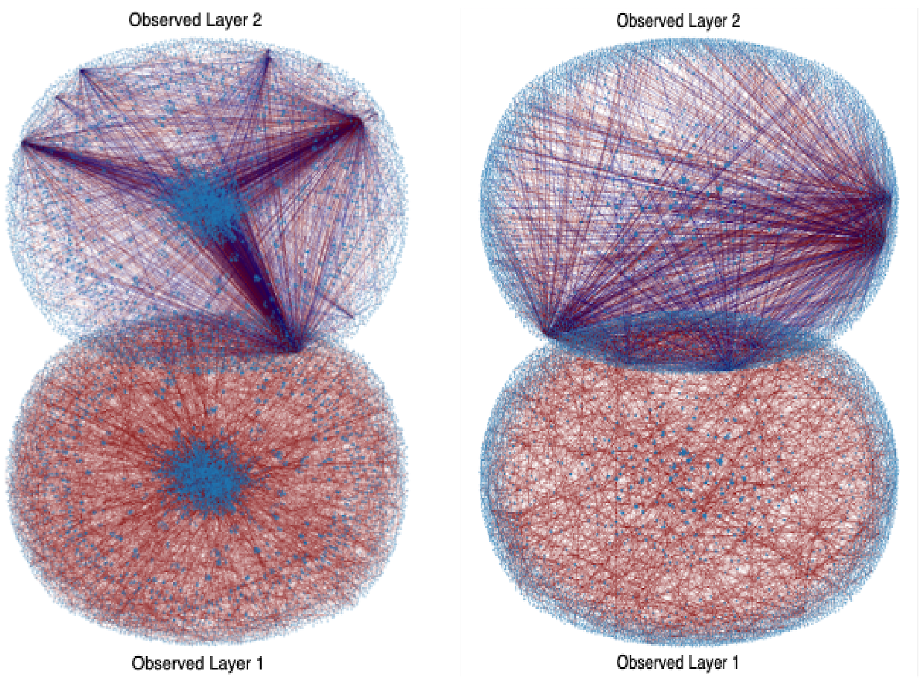
Figure 4.3: Sensitivity of baseline to the number of parameters. This figure depicts DGLFRM's (model with the 2^{nd} best performance) ability to utilize the graph latent structure by expanding the dimension of the latent space. As d' increases, the baseline capacity increases. This experiment attests that expanding d' from 32 to 128 barely increases AUC.

4.6 Qualitative Analysis

In this section, we demonstrate and discuss our model's ability to recover latent layers present in the graph and analyze the interpretable nature of these layers. As discussed in this thesis, four of the evaluated datasets are heterogeneous (originally labeled with layers). To keep this discussion brief, we study results for two out of these four datasets on different experiments.

4.6.1 Graph Latent Layer Retrieval

To show the model's ability to retrieve the latent layers, we withhold the edge labels and obtain the ground-truth layer labels for homogeneous datasets. Figure 4.4a and 4.4b shows the retrieved latent layers of the graph, for ACM and DBLP, respectively.



(a) ACM **(b)** DBLP

Figure 4.4: Retrieved latent layers of the graph by LLGF. The observed layers present in the dataset are depicted by the different levels (2 in ACM and DBLP). While the nodes are common between these 2 levels, the interaction between nodes is different and is limited to only one type. The edge colors indicate the model's prediction for latent layers; note that the node and edge types are hidden during the training.

In each figure, observed layers in a graph are represented by different levels, which is 2 for ACM and DBLP. While the nodes are common between these levels, the interactions between these nodes are different and limited to only one type. For instance, in the ACM dataset, the edges in the bottom level show Paper-Author interaction, and the upper level presents Paper-Subject interactions. The colors depict the retrieved latent layer by the model. For this analysis, we used LLGF-Dot with $L = 2$ and used the decoder's *reconstruction block* to color the edges using the following condition:

$$\textit{if} \quad p(\mathbf{A}_{1mn}|\mathbf{z}_m, \mathbf{z}_n) > p(\mathbf{A}_{2mn}|\mathbf{z}_m, \mathbf{z}_n) \quad \textit{return} \quad \text{blue} \quad \textit{else} \quad \text{red.}$$

In the best-case scenario, the model colors all edges in the same level with the same color. As it is illustrated, our model can recover the latent layers. To the best of our knowledge, the LLGF is the first graph generative model with the ability to recover the latent layers. Note that for DBLP, for visualization purposes, we randomly select 10% of the nodes and include their interactions due to the large size of the dataset.

4.6.2 Edge Representation

As discussed earlier, LLGF, unlike the previous VGAEs, can simultaneously learn the (non-existent) edge representations. Using the proposed model, an edge between nodes m and n can be represented by L dimensional vector $\mathbf{E}_{:nm}$, which is the output of the decoder's reconstruction block for these two nodes. Elements of this vector can be interpreted as the interaction features that are necessary for the decoder's aggregation block to predict an edge and its type if *observed.*

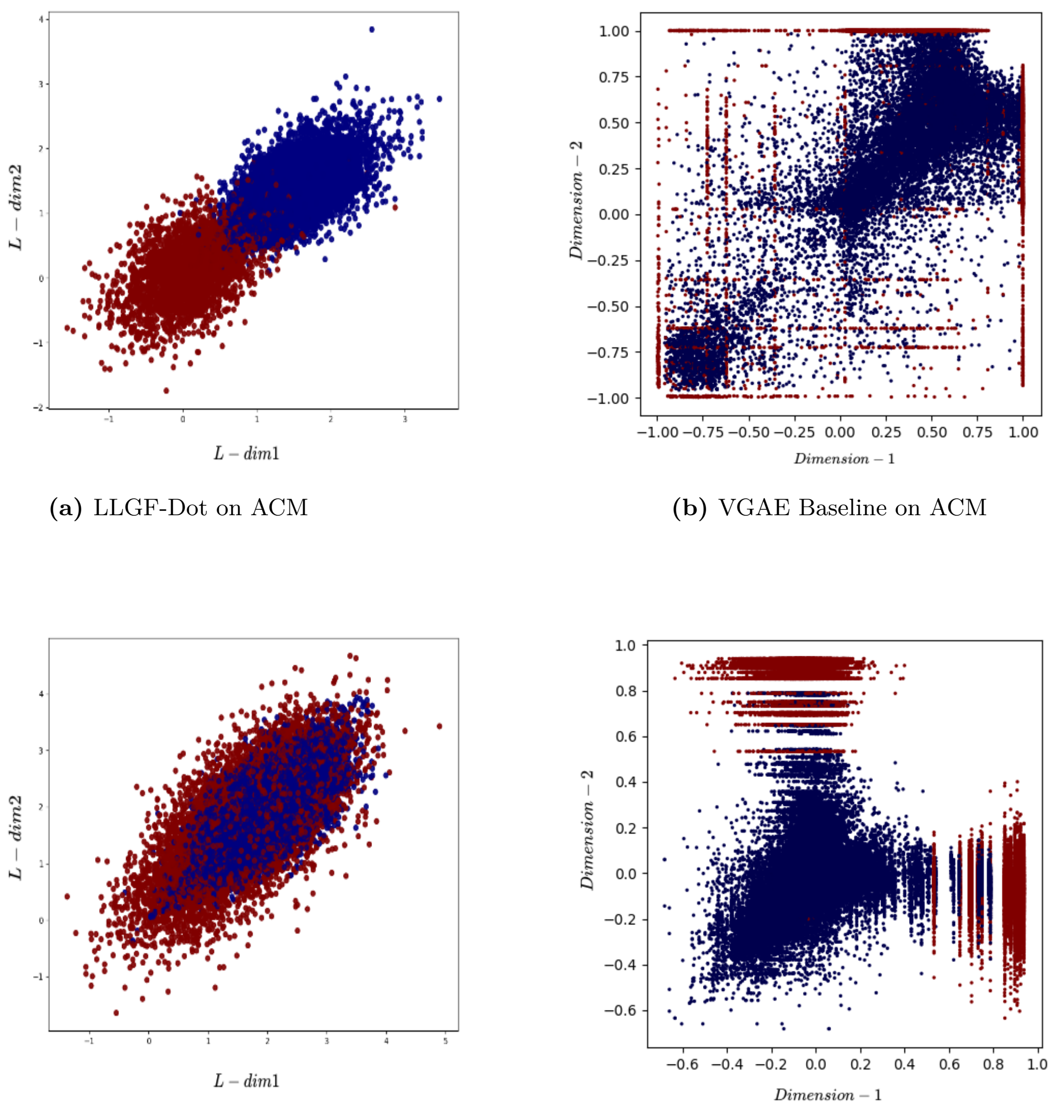
Figure 4.5a and 4.5c illustrates $\mathbf{E}$ for all the edges of ACM and DBLP learned by LLGF-Dot with $L = 2$; we set $L = 2$ to have a 2-dimensional visualization. As our experiment shows, the edge representation of our model captures the observed multi-layer structure. To have better intuition, we depict edge representation obtained from concatenation of the baselines node latent representation, in particular, VGAE [8]. To have a two-dimensional edge representation for baseline we set $d' = 1$ and learned the model as before. In this scenario, the representation of an edge between nodes m and n is $\mathbf{z}_n||\mathbf{z}_m$. Figures 4.5b and 4.5d show the two-dimensional edge representations obtained by concatenating the dyad's two-dimensional node representations for ACM and DBLP. The color here also shows the edge type.
We apply a $Tanh$ function to have the baseline representation in range $[-1, 1]$. Our experiment shows that our dyad representation in Figure 4.5 (Left) captures the latent multi-layer structure better than concatenating the dyad's two-dimensional reduced node representations (Right).

4.6.3 Clustering Edge Representations

We perform an empirical analysis on the edge representations retrieved from our model LLGF-Dot and compare them with baseline methods - VGAE[8] and VGAE-SBM. Again, we train LLGF-Dot to generate these edge representations for ACM and DBLP and run a K-Means clustering algorithm using $k = 2$, which signifies the number of observed edges. Later, we use these cluster predictions to compute Normalized Mutual Information *(NMI)* and Adjusted Rand Index *(ARI)* using the ground-truth edge labels.
We perform a similar clustering experiment and evaluation on the edge representations computed for baselines using the concatenation of node representations for these edges. Figure 4.6 denotes how well our edge representations performed versus baseline methods on these clustering metrics, where 0 denotes low correlation of clusters and 1 denotes perfect labeling.

4.6.4 Latent Layer Weights Assignment

In this section we analyze the weights of latent layers computed towards reconstructing observed layers by our model LLGF-Dot for datasets - ACM, IMDb, DBLP, and IMDb-F.

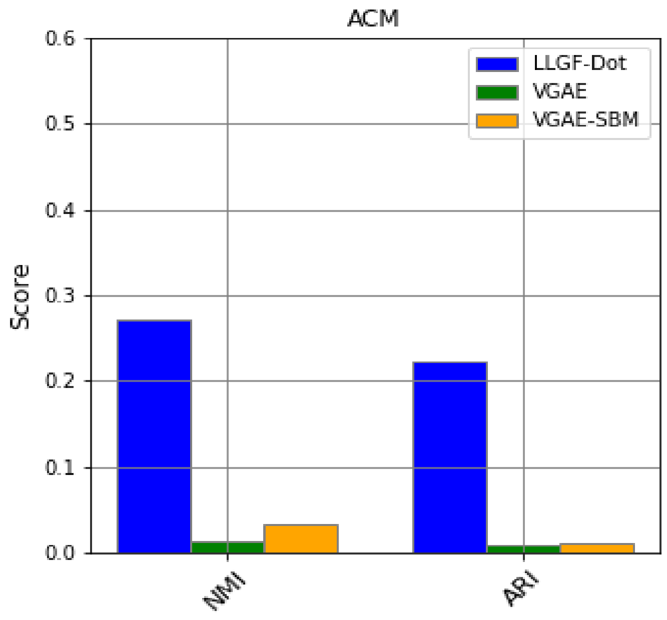


(a) LLGF-Dot on ACM **(b)** VGAE Baseline on ACM

(c) LLGF-Dot on DBLP **(d)** VGAE Baseline on DBLP

Figure 4.5: Edge representation visualization. Color depicts observed edge types, which are Paper-Subject (blue) and Paper-Author (red) in ACM, and, Paper-Conference (blue) and Paper-Author (red) in DBLP. Fig (Left) shows that by using the two dimensions of the latent layers, we can reconstruct the edges reasonably well versus retrieving the edges by simply concatenating the representation of nodes for the edges learned by the baseline method - VGAE[8] (Right). See *Edge Representation* in section 4.6.3 for more details.

We train LLGF-Dot to generate latent layers for each type of observed layer existing in these datasets. Table 4.4 captures some of these L dimensional weights computed for observed layers for the above-mentioned datasets. Here, we set L as discussed in Section 4.4.3. The table shows that the weights are quite sparse (many 0s), and in some cases disjoint, which



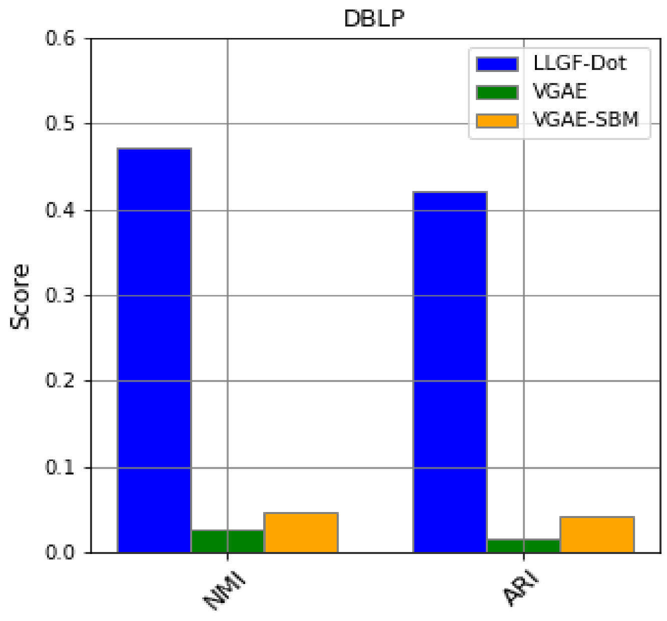


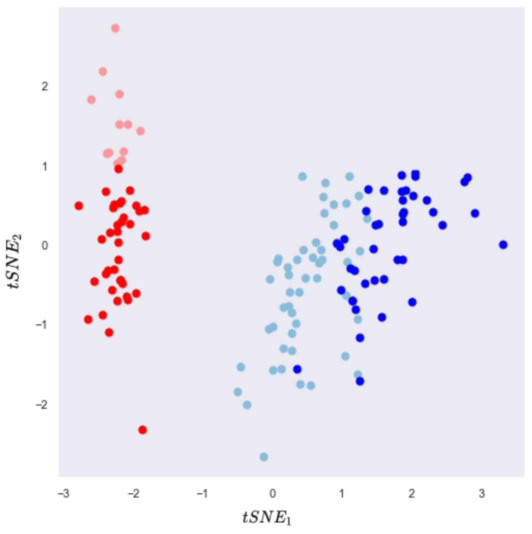
Figure 4.6: Cluster quality evaluation of edge representation by $LLGF - Dot$ and for two baselines using concatenation method of node representations, learned with the original setup. K-Means clustering is performed on the representations to predict the clusters in an unsupervised setting. Further, clustering is evaluated using Normalized Mutual Information (NMI), Adjusted Rand Index (ARI) using ground-truth edge labels for ACM and DBLP.

means that different observed layer utilize different latent layers. It is interesting to note that we observe relative sparsity without an explicit penalty term to encourage sparse weights.

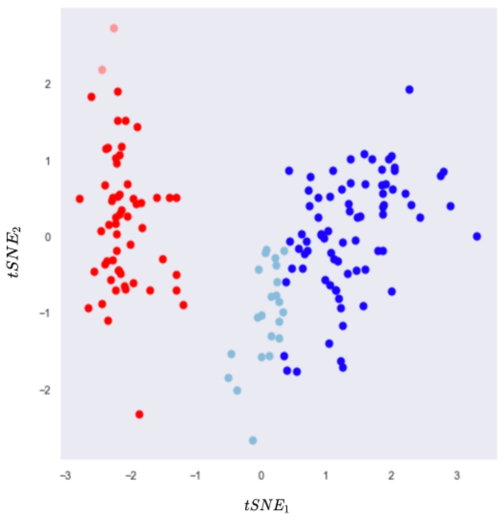
Table 4.4: Latent layer weights assignment analysis. Observed layers denoted by L' represent the ground-truth labels present in each dataset. Latent layers denoted by L represent weights formulated in the reconstruction block.

Dataset	Observed Layers (L')	Latent Layers (L)							
ACM	Paper-Author				0.7	0.3			
	Paper-Subject				0.4	0.6			
IMDb	Movie-Director	0.2	0.1	0.4	0	0.1	0.2	0	0
	Movie-Actor	0	0.1	0	0	0.4	0.3	0.2	0
DBLP	Paper-Conference		0	0	0.5	0.3	0.2	0	
	Paper-Author		0.1	0.4	0.2	0.1	0.1	0.1	
IMDb-F	Movie-Director	0.1	0.3	0.2	0.1	0.1	0.1	0.1	0
	Movie-Actor	0.2	0.1	0	0.1	0.3	0.1	0.2	0

We also perform an experiment to visualize the quality of representations captured in two of the four datasets - IMDb and IMDb-F by our model LLGF-Dot. First we start with the dyad representation $\mathbf{z}_n||\mathbf{z}_m$ that concatenates the node representations for each observed edge. Then we apply t-SNE [24] to project the dyad representations to a 2-dimensional space. Each edge is then colored according to its ground-truth observed edge type. Here the shading is applied based on the probability of the ground-truth assignment. Figure 4.7 illustrates these dyad representations for datasets - IMDb and IMDb-F. The lighter shades of blue and red represent the weaker strength of these edge types.



(a) IMDb



(b) IMDb-F

Figure 4.7: Latent Layer visualization. Dyad edge representations are projected to a 2-dimensional space using t-SNE [24]. The figure shows 70 randomly selected observed edges. Color depicts observed edge types: Movie-Actor (blue) and Movie-Director (red) in IMDb and IMDb-F respectively. The lighter shades of blue and red represent the weaker strength of these edge types.

Chapter 5

Conclusion and Future Work

Layers or edge types influence the graph generation process; recent neural graph models have successfully incorporated available layer labels to improve graph generation. However, often layer structure is not explicitly annotated in the data. Our work shows that interpretable latent layers can be usefully inferred to enhance the expressive capacity of variational generative models. We described LLGF, a general framework for learning latent layers that generalize Variational Graph Auto-Encoder (VGAEs) models to incorporate arbitrary multi-relational representation encoders and link generation decoders.

Our decoder framework perceives latent graph layers as components of the observed layers and uses a learnable aggregation function to generate observed layers by aggregating the latent layers. Our encoder framework is based on learning parallel node representations, one for each latent layer. Our experiments applied the encoder framework to add latent layer learning to the Relational GCN [20]. We evaluated LLGF on eight benchmark graph learning datasets on the link prediction task. Four of the datasets were heterogeneous (originally labeled with layers). By with-holding the edge labels, we obtain ground-truth layer labels for a homogeneous dataset. LLGF increased link prediction accuracy, especially for heterogeneous datasets (up to 5% AUC), and recovered the ground-truth layers very well.

Future Research. Latent layers or edge types represent an important aspect of the graph generation process and open up several promising directions for future research.

Learning the number of latent layers. While in this thesis, we treated the number of latent layers as a hyper-parameter, it is possible to apply Bayesian techniques to estimate them from data. For example, [15] applies a stick-breaking process to estimate the dimension of node representation in the VGAE model, which can likely be adapted to edge types.

Learning with Edge Attributes. Many real-world graph data specify observed attributes for links (e.g the salary of a person working in a company), which can be predicted from learned edge representations. We expect improvement in the prediction of latent edge types and edge attributes by learning them jointly.

Graph Embeddings. The learned $w_{ll'}$ parameters in our decoder model capture aspects of global graph structure because they measure the presence of a latent layer in an observed layer. The representation of global graph structure can be strengthened with graph embeddings, which are known to increase the generative power of many graph network models [5]. Graph embeddings can be easily added to our latent layer framework through message passing.

Inductive Graph Learning. Another future direction is to evaluate the usefulness of latent layers for inductive learning [5, Sec.6.1.1.], where a system is trained on observed nodes and then applied to unseen nodes.

In sum, latent layer learning represents an important component of graph generation that is interpretable and increases the predictive power of a multi-relational graph network model.

Bibliography

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[2] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks. *arXiv preprint arXiv:1904.05811*, 2019.

[3] Shaohua Fan, Xiao Wang, Chuan Shi, Emiao Lu, Ken Lin, and Bai Wang. One2multi graph autoencoder for multi-view graph clustering. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *Proceedings of The Web Conference*, 2020.

[4] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2434–2444, 2019.

[5] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[6] Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. An attention-based graph neural network for heterogeneous structural learning. *CoRR*, abs/1912.10832, 2019.

[7] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR*, 2014.

[8] Thomas Kipf and M. Welling. Variational graph auto-encoders. *ArXiv*, abs/1611.07308, 2016.

[9] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[10] Donald E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing.* Association for Computing Machinery, New York, NY, USA, 1993.

[11] Pierre Latouche, Etienne Birmelé, and Christophe Ambroise. Overlapping stochastic block models with application to the french political blogosphere. *Ann. Appl. Stat.*, 5(1):309–336, 2011.

[12] Irene Z Li, Alexander R. Fabbri, S. Hingmire, and D. Radev. R-vgae: Relational-variational graph autoencoder for unsupervised prerequisite chain learning. In *COLING*, 2020.

[13] Xiaoke Ma, Di Dong, and Quan Wang. Community detection in multi-layer networks using joint nonnegative matrix factorization. *IEEE Trans. Knowl. Data Eng.*, 31(2):273–286, 2019.

[14] Yao Ma, Suhang Wang, Charu C. Aggarwal, Dawei Yin, and Jiliang Tang. Multi-dimensional graph convolutional networks. In *SIAM International Conference on Data Mining, SDM*, pages 657–665, 2019.

[15] Nikhil Mehta, Lawrence Carin, and Piyush Rai. Stochastic blockmodels meet graph neural networks. *CoRR*, abs/1905.05738, 2019.

[16] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. *CoRR*, abs/1906.01195, 2019.

[17] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*, 2018.

[18] Davide Rigoni, Nicolò Navarin, and Alessandro Sperduti. Conditional constrained graph variational autoencoders for molecule design. In *2020 IEEE Symposium Series on Computational Intelligence, SSCI*, pages 729–736, 2020.

[19] Arindam Sarkar, Nikhil Mehta, and Piyush Rai. Graph representation learning via ladder gamma variational autoencoders. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5604–5611, 2020.

[20] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607, 2018.

[21] P. Sen, Galileo Namata, M. Bilgic, L. Getoor, B. Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29:93–106, 2008.

[22] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. *CoRR*, abs/1811.04441, 2018.

[23] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning - ICANN*, volume 11139 of *Lecture Notes in Computer Science*, pages 412–422, 2018.

[24] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[25] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[26] Ziqiang Weng, Weiyu Zhang, and Wei Dou. Adversarial attention-based variational graph autoencoder. *IEEE Access*, 8:152637–152645, 2020.

[27] James D. Wilson, John Palowitch, Shankar Bhamidi, and Andrew B. Nobel. Community extraction in multilayer networks with heterogeneous community structure. *J. Mach. Learn. Res.*, 18:149:1–149:49, 2017.

[28] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[29] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery ; Data Mining*, KDD '18, page 974–983, New York, NY, USA, 2018. Association for Computing Machinery.

[30] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 11960–11970, 2019.

[31] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 06 2018.