Answering Probabilistic Graph Queries from a Single Model

by

Parmis Naddaf

B.Sc., Simon Fraser University, 2021

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:	**Parmis Naddaf**
Degree:	**Master of Science**
Thesis title:	**Answering Probabilistic Graph Queries from a Single Model**
Committee:	**Chair:** Yagiz Aksoy Assistant Professor, Computing Science
	Oliver Schulte Supervisor Professor, Computing Science
	Ke Li Committee Member Assistant Professor, Computing Science
	Ke Wang Examiner Professor, Computing Science

Abstract

A Probabilistic Graph Query (PGQ) specifies target components of a graph to be predicted, given observed components as evidence. For example, a single-link query specifies a target link, with other links and node attributes as evidence. Previous works have developed customized models for different PGQ types; this work describes a unified framework for answering various types of PGQs based on an inductively trained probabilistic Generative Graph Model (GGM). Given a trained GGM and any user PGQ, our approach constructs, without retraining, a conditional model that outputs the PGQ probability. In this thesis, we utilize Variational Graph Auto-Encoders (VGAEs) as GGM with Conditional Variational Auto-Encoders (CVAEs) as conditional models. For evaluation, we apply query answering to inductive link prediction queries on six benchmark datasets. We find that for most datasets and query types, the VGAE predictions are better than baseline methods customized for link prediction.

Keywords: Inductive Link Prediction; Variational Graph Auto-Encoder; Probabilistic Graph Query; Conditional Variational Auto-Encoder

Dedication

This thesis is dedicated to my family for their endless love, support, and encouragement. Their sacrifices and hard work have enabled me to pursue my education and I am forever grateful.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Oliver Schulte, for their unwavering support, guidance, and encouragement throughout my entire journey in the master's program.

A special thanks to Erfaneh Mahmoudzadeh, for their unrelenting support and willingness to go above and beyond to ensure the success of my research, and to Kiarash Zahirnia, who provided excellent support and guidance throughout my master's journey.

I extend my gratitude to to Dr. Ke Wang for fulfilling the role of my examiner, Dr. Ke Li for serving as a member of my committee, and Dr. Yagiz Aksoy for chairing my thesis defense. Their contributions and support are highly appreciated.

Table of Contents

List of Tables

List of Figures

Definitions

Notation	**Definition**
G	graph network
V	set of nodes in a graph
E	set of edges in a graph
$\boldsymbol{Z}$	latent representation matrix of nodes
$\boldsymbol{X}$	nodes feature matrix
$\boldsymbol{A}$	adjacency matrix
N	number of nodes in a graph
k	number of attributes per node
$G\|V$	subgraph of G induced by nodes V
$\boldsymbol{Y}$	set of target relational random variables where $Y_i = \boldsymbol{A}[u_i, v_i]$
$\boldsymbol{E}$	set of evidence relational random variables
$\mathcal{X}$	set of attributed node variables
$\mathcal{A}$	set of link variables
u, v	a random nodes in a graph
$\mathbb{P}$	probabilistic graph query
$\boldsymbol{y}$	target instance
$\boldsymbol{e}$	evidence instance
G_{in}, G_{tr}, G_{te}	input, train, and test graphs respectively
$\mathcal{A}_{in}, \mathcal{A}_{tr}, \mathcal{A}_{te}$	set of link variables defined by the input, train, and test graphs respectively
d	latent node embedding dimensions
Λ	block matrix of the SBM decoder
m	number of assignments in the target
p_θ	graph decoder
q_φ	graph encoder
$KL(p\|\|q)$	Kullback-Leibler divergence
μ	mean of a distribution
σ	standard deviation of a distribution

Chapter 1

Introduction and Overview

We review basic definitions concerning graphs, link prediction, and graph generative models. Then we give an overview of how we utilize a trained GGM to answer link prediction queries.

1.1 Definitions

1.1.1 Graphs

An attributed graph is a pair $G = (V, E)$ comprising a finite set of V nodes and E links. An undirected graph is a type of graph in which the links connecting pairs of vertices have no direction. In other words, if there is an link between nodes u and v, it is considered to be the same as the link between nodes v and u. Undirected graphs are often used to model symmetrical relationships between objects, such as friendships in a social network or roads in a transportation network. Figure 1.1 is one example of such graph.

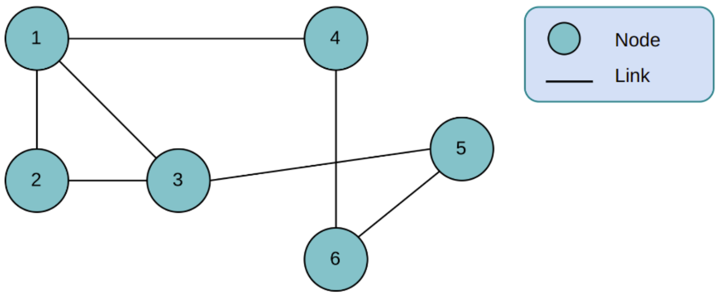
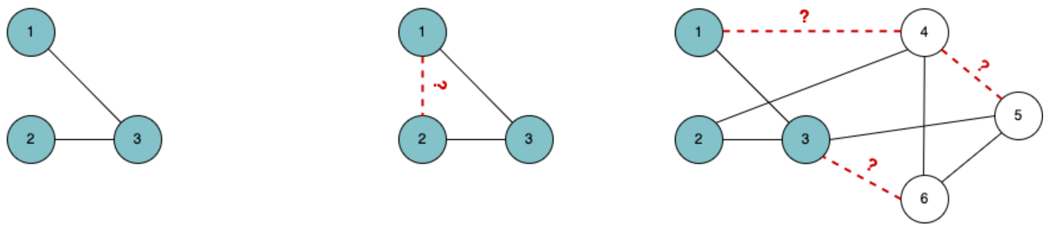


Figure 1.1: Example of an undirected graph with six nodes and seven links.

1.1.2 Link Prediction Task

Link prediction is a task in graph analysis that involves predicting missing links in a graph, based on the relationships between the existing vertices. The goal of link prediction is to identify which vertices in a graph are likely to be connected by a link, based on the patterns of connections and relationships between other nodes. Link prediction is an important tool in a variety of fields, including social network analysis and recommendation systems.

Inductive Link Prediction (ILP) is made based on the information learned from the existing graph, and then used to extend the graph to include missing links for the new nodes. This is in contrast to **transductive link prediction**, where the goal is to predict links for a specific set of nodes, given their features and relationships to other nodes in the graph. Figure 1.2 illustrates the difference between inductive and transductive link prediction methods.



(a) Training graph (b) Transductive prediction (c) Inductive prediction

Figure 1.2: The comparison of transductive and inductive link predictions.

1.1.3 Variational Auto-Encoders

A Variational Auto-encoder (VAE) is a type of generative model in deep learning that is used for unsupervised representation learning first introduced by [10]. It learns to map high-dimensional data, such as images or texts, into a lower-dimensional representation, called a latent space $\boldsymbol{Z}$, and then maps the latent representation back to the original high-dimensional space. This results in a compact and meaningful representation that can be used for various tasks such as image generation. The variational auto-encoder is a generative model that can generate new examples by sampling from the learned latent space and decoding the samples into high-dimensional data.

Variational Graph Auto-Encoders (VGAE)s are a special type of VAE that work with graph data [6, 11]. A VGAE encoder takes as input a feature matrix $\boldsymbol{X}$ and an adjacency matrix $\boldsymbol{A}$ and maps them to a lower dimensional distribution $\boldsymbol{Z}$. The decoder then by using a variational approximation function outputs the reconstructed adjacency matrix that can be used for link prediction. Figure 1.3 shows the architecture of the VGAE model.

1.1.4 Conditional Variational Auto-Encoders

A Conditional Variational Auto-Encoder (CVAE) extends the traditional Variational Auto-encoder (VAE) architecture to incorporate additional conditioning information [20]. The encoding and decoding process is conditioned on additional information, such as class labels, that provide context and guide the learning process. In the context of link prediction, the additional information is going to be the label of the links that we want to predict.

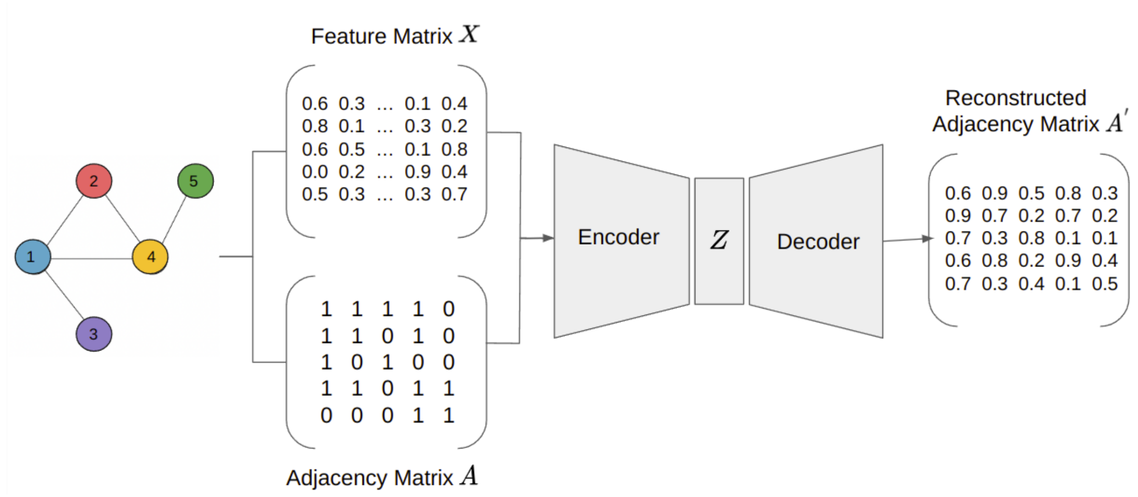


Figure 1.3: Example of a VGAE structure.

1.2 Background

Many prediction tasks for graph data can be formulated as a probabilistic *graph query*: predict a set of *target* links/attributes, given observed *evidence* links/attributes of a graph. For example, Inductive Link Prediction (ILP) involves predicting the existence of one or more links involving test nodes that are not observed during training. Previous researchers have developed a separate customized method for each type of graph query. For instance, different ILP variants have been studied in recent works, including the following evidence settings for predicting the existence of a single target link. (1) Condition on the attributes of the disconnected test nodes [8].(2) Condition on links among other test nodes only [21]. (3) Condition on links among training and test nodes [30].

1.3 Overview

1.3.1 Motivation

A query answering system defines a query language (e.g, SQL) provides fast answers for valid queries. If the observed workload indicates that a particular query type is especially important for users, the system can be optimized for it. A query answering system is important for deploying graph learning in a production environment where we do not know in advance which graph queries will be important, and users may not have the resources to build a customized machine learning solution for different query types. A query answering system based on a single model can be seen as a *multi-task system.*

1.3.2 Approach

Fig 1.4 shows the proposed system design. Our approach is a form of *domain-specific pre-training*: We learn a probabilistic generative model from data in a deployment domain. After learning a domain model from data, the model is used to answer queries with no further learning required. The deployment domain can be large, whereas queries are typically small (e.g., a single-link prediction query may involve two nodes and a k-hop neighborhood around them).

Specifically, we train a (VGAE) [6, 11] inductively, so that the VGAE can be applied to graphs of different sizes [7]. The joint probabilities over graphs (implicitly) define a conditional probability $\mathbb{P}(target|evidence)$ for every PGQ. The VGAE utilizes an encoder-decoder architecture with a Graph Neural Network (GNN) as an encoder. Our main technical contribution is to show how, given a PGQ specified by the user, the VGAE can be *dynamically* transformed to a zero-shot conditional VGAE (CVGAE) for the query. This approach can be seen as a classic *inference from a model* [17].

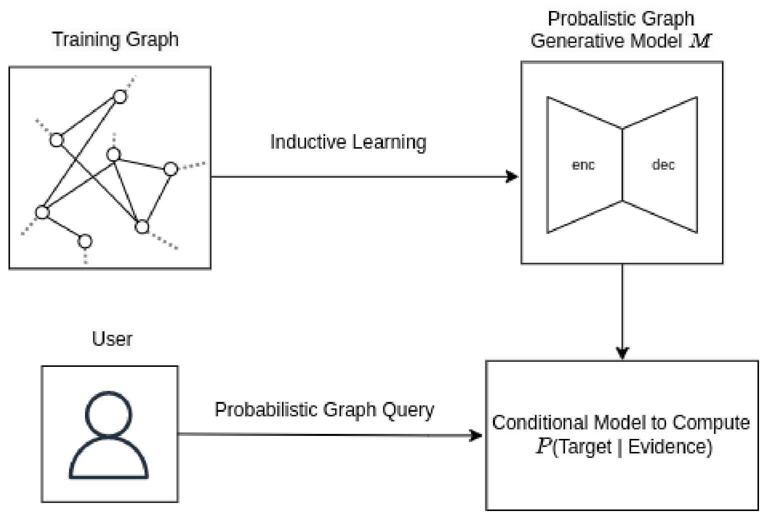

Training Graph
Probalistic Graph Generative Model M
Inductive Learning
enc
dec
User
Probabilistic Graph Query
Conditional Model to Compute P(Target | Evidence)

Figure 1.4: After training a single graph generative model, a user query can be answered by dynamically constructing a zero-shot conditional model.

CVAEs were introduced as neural models for generating structured output given initial information [20]; to our knowledge, this work is the first application of CVAEs to PGQs. A CVAE comprises two components: a (conditional) prior network for encoding the evidence and a recognition network for jointly encoding the evidence and the target. The conditional probability $P(target|evidence)$ is estimated by sampling using the prior network.

1.3.3 Evaluation

Our evaluation focuses on inductive link prediction because it is a challenging task that arises frequently in practice. We investigate four versions of the ILP problem (cf. fig. 3.1), depending on whether 1) the target is a single-link between test nodes or comprises multiple links, and 2) the evidence comprises only links among test nodes (fully-inductive), or also links among training and test nodes (semi-inductive). For a single-link target, well-established single-link methods are used as baselines. We find that inference from the VGAE model is competitive with, and for many datasets more accurate than, custom methods for single-link prediction. For multi-link targets, our baseline is to estimate link probabilities independently. We find that joint prediction scores are more accurate than independent prediction, leading to superior performance by the VGAE model.

1.3.4 Contributions

The contributions of this thesis can be summarized as follows.

- Introducing the task of answering probabilistic graph queries.
- A single model approach for answering PGQs: estimate a conditional probability for a large class of graph queries by inference from a single inductively trained GGM.
- A method for dynamically constructing a conditional prediction model for each given graph query.
- An application of our query answering method to new practically useful link prediction queries, including joint predictions of multiple links.

Chapter 2

Related Work

Graph Queries. Powerful (non-probabilistic) graph query languages have been developed, such as Cypher and SPARQL [4,15]. Graph queries are typically used to retrieve data that is stored in a graph data model. Such graph queries typically return a set of nodes or links that satisfy a complex condition [5]. In contrast, probabilistic graph queries return a probability for a subgraph, conditional on another subgraph. Below we often refer to probabilistic graph queries as "graph queries" for short, although the concept is different from that used in graph query languages.

Inductive Graph Training. Inductive graph learning has been a major topic in recent research [7,16,28]. GraphSage [7] learns embeddings for the new nodes in a graph by aggregating information from their local neighborhood and tries to capture the structural properties of the graph by looking at the nodes and links surrounding each node. DeepGL [16] uses network motifs as base features for network representation learning. GraphSAINT [28] uses a sampling strategy to select a small subset of nodes and their neighborhoods for each training iteration to achieve faster computation and memory efficiency.

The proposed query answering approach can be used with any inductive encoder-decoder graph neural network architecture, which we train with the variational Evidence Lower BOund (ELBO) objective described in equation 3.1 below. To our knowledge, this is the first inductive training with the variational ELBO objective.

Graph Generative Models. Graph generative models are machine learning models that are designed to generate graphs that exhibit similar properties to a given input graph or a set of input graphs. GraphGAN consists of two neural networks, a generator and a discriminator, that are trained together in a game-like manner to generate new graph that is similar to the training graph [24]. GraphGANs do not provide explicit graph probabilities for the entire graph, although some designs use local link probabilities to generate the graph. GraphVAEs generate graphs from a continuous embedding in the context of variational autoencoders but cannot be applied to graphs of different sizes from the training graph [19].

Auto-regressive models like GRAN [12] provide explicit graph probabilities for graphs of different sizes, and likely admit graph-conditional variants—the target graph links can be incrementally built up from the evidence subgraph. They typically assume a node ordering for sequentially generating subgraphs. An interesting research question is how to find an effective node ordering dynamically for the target nodes in a graph query, conditional on the evidence subgraph. For graph diffusion, DiGress method is a variant that conditions the diffusion process on a subgraph [23, Sec.E]. However, the subgraph is restricted to known nodes. It is not clear whether graph diffusion models support computing explicit graph probabilities. In general, computing exact data likelihoods in diffusion models is a difficult problem [25, Sec.4.3].

Link Prediction Methods. SEAL is a transductive link prediction method that extracts a local subgraph around each target link, and aims to learn a function to map the subgraph patterns to link existence [29]. GraIL is an inductive extension of SEAL to directed multi-relational knowledge graphs [21]. One limitation of these method is that they are designed to do single-link prediction and for a set of target links, a different enclosing subgraph must be extracted for each target link. DEAL uses two auto-encoder networks to learn embeddings of the source and target nodes separately [8]. The source and target embeddings are then combined and passed through a fully connected layer to predict the existence of a link between them. DEAL's performance is highly dependent on the node features of the graph as it does not use any information from the structure of the graph.

CVAE for PGQs. CVAEs are models for generating structured output [20], which makes them suitable for graph prediction. While they have been extensively applied to visual data, such as images and video, to our knowledge their application to graph queries is new. These methods use the conditional ELBO to define a CVAE training objective, for a fixed type of target and evidence. In our experiments, the CVAE architecture is used for inference but the conditional ELBO is not used for training.

Inference from a Model. Previous works on graph neural networks have focused on specific predictive tasks specified by the researcher. To our knowledge, inference from a generative model to estimate conditional probabilities is a novel application of inductive neural graph learning. However, general query answering capabilities are a major use case for non-neural statistical-relational models, such as Markov Logic networks [2] and Probabilistic Soft Logic [9] which are used for reasoning and learning in domains that involve uncertainty and incomplete information. The fact that general query answering is a key capability of previous graph models motivates our goal of adding this capability to neural models. Statistical-relational models are based on very different assumptions and model classes (e.g. exponential random graph models) from VGAEs.

Chapter 3

Method and Approach

This chapter will define probabilistic graph queries, differentiate different kinds of graph queries, and propose a method to answer them using CVAE methods with VGAE models.

3.1 Problem Statement

Probabilistic Graph Queries. Graph queries specify a set of links and attributes for the nodes involving the links. Probabilistic Graph Queries allow users to query over nodes and structures of graphs and return probabilistic answers to those queries.The specification can involve a language as complex as first-order logic [2]. In this thesis we introduce a relatively simple PGQ syntax, consisting primarily of conjunctions of links and/or attributes, which is powerful enough to express the prediction tasks that have previously been studied in graph learning.

Data Format. An attributed graph is a pair $G = (V, E)$ comprising a finite set of nodes V and links E where each node is assigned a k-dimensional attribute $\boldsymbol{x}_i$ with $k > 0$. An attributed graph can be represented by an $N \times N$ adjacency matrix $\boldsymbol{A}$ with $\{0, 1\}$ Boolean entries, together with an $N \times k$ node feature matrix $\boldsymbol{X}$.

Definition of Probabilistic Graph Queries. Answering a graph query requires assigning a joint probability to joint values for a set of *target* relational random variables $\boldsymbol{Y}$, given values of *evidence* random variables $\boldsymbol{E}$. A relational random variable corresponds to either a node attribute or a link. Accordingly we define $\mathcal{X} = \{\boldsymbol{x}[u, i] : u \in V, 1 \leq i \leq k\}$ as the set of **attributed node variables** and $\mathcal{A} = \{\boldsymbol{A}[u, v] : u \in V, v \in V\}$ as the set of **link variables**. A **probabilistic graph query** is of the form $\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{E} = \boldsymbol{e})$ where the **target** $\boldsymbol{Y} = (\boldsymbol{X}^{\boldsymbol{Y}}, \boldsymbol{A}^{\boldsymbol{Y}})$ and the **evidence** $\boldsymbol{E} = (\boldsymbol{X}^{\boldsymbol{E}}, \boldsymbol{A}^{\boldsymbol{E}})$ with $\{\boldsymbol{X}^{\boldsymbol{E}}, \boldsymbol{X}^{\boldsymbol{Y}}\} \subseteq \mathcal{X}$ and $\{\boldsymbol{A}^{\boldsymbol{E}}, \boldsymbol{A}^{\boldsymbol{Y}}\} \subseteq \mathcal{A}$. The following types of queries are differentiated:

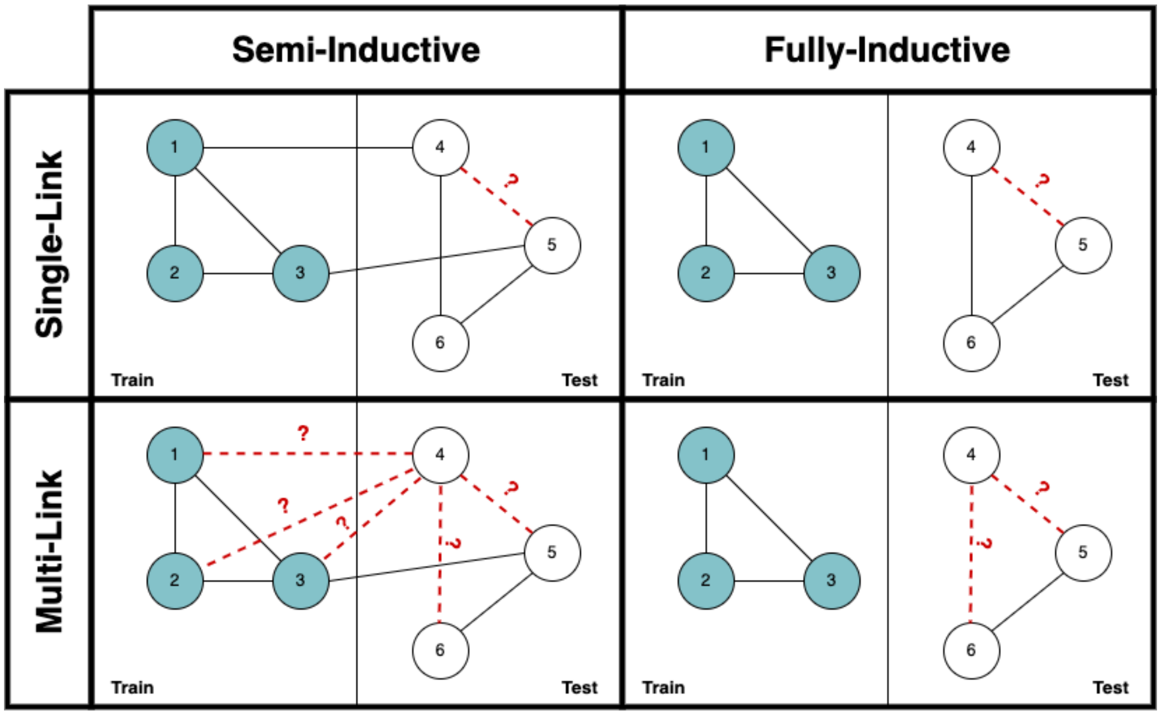


Figure 3.1: Illustration of Semi/fully-inductive and single/multi-link prediction queries. The dashed red lines represent target links.

- A query is **attribute-complete** if the following conditions hold. For each node u that appears in the evidence $\boldsymbol{E}$ or target $\boldsymbol{Y}$, the evidence contains all attributes of u. That is, for each u, and attributes $1 \leq i \leq k$, all $\boldsymbol{x}[u, i]$ must appear in $\boldsymbol{E}$.
- A **link prediction** query targets only links (i.e., $\boldsymbol{Y} \subseteq \mathcal{A}$).
- Given a partition of nodes into observed training nodes and unobserved test nodes, a query is **inductive** if a test node appears in the target.

Link prediction queries are typically attribute-complete, and node classification queries are not. In the next section we describe a general procedure for constructing a CVAE for any attribute-complete link prediction query given a trained VGAE, *without* further training. CVAEs can be used to answer both inductive queries and purely transductive link prediction queries.

3.1.1 Examples of Inductive Link Prediction Queries

Single-link prediction from the remaining graph is the most common link prediction setting, One example of it is in recommendation systems [30]. In a recommendation setting, neighborhood prediction corresponds to predicting whether a user will select a set of items. In a social network setting, it corresponds to predicting a group of friends for a user. This is while Single-link prediction from attributes only is a type of cold-start setting [8].

Symbol	Semi-Inductive	Fully-Inductive
$\boldsymbol{A}^{\boldsymbol{Y}}$ (Single)	$\{\boldsymbol{A}[u,v]\}$ where $u \in V_{te}, v \in V$	$\{\boldsymbol{A}[u,v]\}$ where $u \in V_{te}, v \in V_{te}$
$\boldsymbol{A}^{\boldsymbol{Y}}$ (Multi)	$\{\boldsymbol{A}[u,v_1],...,\boldsymbol{A}[u,v_m]\}$ where $u \in V_{te}, v_i \in V, m = N$	$\{\boldsymbol{A}[u,v_1],...,\boldsymbol{A}[u,v_m]\}$ where $u \in V_{te}, v_i \in V_{te}, m = \lvert V_{te}\rvert - 1)$
$\boldsymbol{A}^{\boldsymbol{E}}$	$\mathcal{A}_{in}$ - $\boldsymbol{A}^{\boldsymbol{Y}}$	$\mathcal{A}_{te}$ - $\boldsymbol{A}^{\boldsymbol{Y}}$
$\boldsymbol{X}^{\boldsymbol{E}}$	$\mathcal{X}_{in}$	$\mathcal{X}_{te}$

Table 3.1: Definition of Test Queries for different link prediction methods.

We further distinguish query types according to how much they generalize from the training data. The **input graph** is one large graph $G_{in} = (V, E)$. Suppose the nodes in the graph are divided into training nodes and test nodes that are unseen during the training process.

Given a subset of nodes $V' \subseteq V$, the (induced) subgraph $G\restriction V'$ comprises the subset and all links among them: $G\restriction V' = (V', E')$ where $E' = E \cap (V' \times V')$. The **train graph** G_{tr} is the subgraph induced by the training nodes V_{tr}, that is we remove the test nodes and their links, so $G_{tr} = G\restriction V_{tr}$. Similarly the **test graph** is the subgraph induced by the test nodes s V_{te}, so $G_{te} = G\restriction V_{te}$. We write $\mathcal{A}_{in}, \mathcal{A}_{tr}, \mathcal{A}_{te}$ for the set of link variables defined by the input, train, and test graphs respectively.

In a **fully-inductive query**, target links are from the test graph only. Evidence links comprise all test graph links that are not targets. Evidence nodes comprise all the nodes that are in the test graph.

In a **semi-inductive query**, target links involve at least one test node. Evidence links comprise all links that are not targets (both training and test links). Evidence nodes comprise all the nodes that are in the input graph. Figure 3.1 illustrates fully and semi-inductive queries and table 3.1 defines their format precisely using our PGQ language.

3.2 Model Definition

The method can be applied with any encoder-decoder graph neural network architecture that outputs node representations. As discussed below (Section. 3.2.1), conditional VAE inference techniques are utilized for query answering. Therefore the VGAE is trained using the variational ELBO objective [6,11]. In order to answer inductive queries involving unseen nodes, the VGAE is trained without incorporating node IDs [6, Ch.5.1.1].

Let $\boldsymbol{Z}$ be an $N \times d$ matrix that represents latent node embeddings. Like other VGAE models, links are generated independently given node embeddings:

$$p_\theta(\boldsymbol{A}|\boldsymbol{Z}) = \prod_{u,v} p_\theta(\boldsymbol{A}[u,v]|\boldsymbol{Z}[u], \boldsymbol{Z}[v]),$$

where $p_\theta : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ is a trainable **decoder**.

The graph **encoder** $q_\varphi(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{A})$ is implemented by a GNN that takes as input an attributed graph and returns latent node embeddings. Existing graph neural networks require all attribute values to be specified for all nodes, which is why we consider only attribute-complete queries. The training loss is the variational ELBO for link reconstruction [11]:

$$\mathcal{L}(\theta, \varphi) = E_{\boldsymbol{Z} \sim q_\varphi(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})}\big[\ln p_\theta(\boldsymbol{A}|\boldsymbol{Z})\big] - KL\big(q_\varphi(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{A})||p(\boldsymbol{Z})\big). \tag{3.1}$$

where $KL(.||.)$ is Kullback-Leibler divergence which measures difference between two probability distributions. $p(\boldsymbol{Z})$ is prior distribution of $\boldsymbol{Z}$ which is usually taken as a standard Gaussian distribution. The approximate posterior $q_\varphi(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{A})$ is the graph encoder. The reconstruction likelihood $p_\theta(\boldsymbol{A}|\boldsymbol{Z})$ is computed by the decoder of the VGAE model.

Equation (3.1) is the training objective used in our experiments unless otherwise noted.

3.2.1 Graph Query Answering with Dynamic Conditional VGAEs

Below describes a solution for the following *inference problem definition*:

- **Given** a single fixed graph generative model M, trained as in section 3.2, defining a distribution $\mathbb{P}_M$ over graphs
- and an attribute-complete link prediction query (Section 3.1) with target $\boldsymbol{Y} = \boldsymbol{y}$ and evidence $\boldsymbol{E} = \boldsymbol{e}$, chosen dynamically by the user,
- **output** an estimate of $\mathbb{P}_M(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{E} = \boldsymbol{e})$.

The first step is to dynamically convert the generative VGAE M into a CVGAE for the given query. The second is to apply CVAE prediction methods to estimate the target query probability.

3.2.2 From VGAE to CVGAE

Sohn et al. introduced the CVAE as the conditional version of VAE for estimating conditional probabilities [20]. Assume that the query target is a list $\boldsymbol{Y} = \boldsymbol{y}$ comprising m assignments $Y_i = y_i, i = 1, \ldots, m$ where $Y_i = \boldsymbol{A}[u_i, v_i]$ and y_i is either 0 or 1. Then the conditional probability for a VGAE model is given by

$$\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{E} = \boldsymbol{e}) = \int_{\boldsymbol{Z}} \mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{Z}) p(\boldsymbol{Z}|\boldsymbol{E} = \boldsymbol{e}) d\boldsymbol{Z} \tag{3.2}$$

where we have used the fact that given embeddings for nodes u_i and v_i, the link $\boldsymbol{A}[u_i, v_i]$ is independent of other links and node attributes. $\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{Z})$ is implemented by the

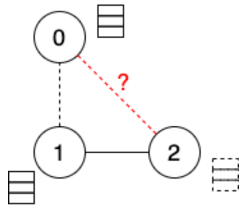


Figure 3.2: Example of a query

VGAE decoder p_θ. To approximate the conditional distribution of node embeddings given the evidence, we use the graph encoder

$$p(\boldsymbol{Z}|\boldsymbol{E}=\boldsymbol{e}) := q_\varphi(\boldsymbol{Z}|\boldsymbol{A}^{\boldsymbol{E}}, \boldsymbol{X}^{\boldsymbol{E}}). \tag{3.3}$$

In terms of the CVAE model [20], the graph encoding $q_\varphi(\boldsymbol{Z}|\boldsymbol{A}^{\boldsymbol{E}}, \boldsymbol{X}^{\boldsymbol{E}})$ represents the prior network. The encoding $q_\varphi(\boldsymbol{Z}|\boldsymbol{A}^{\boldsymbol{E},\boldsymbol{Y}}, \boldsymbol{X}^{\boldsymbol{E},\boldsymbol{Y}})$ represents the recognition network, which is used in importance sampling (see Section 3.2.3).

To compute the prior and recognition graph encodings, we construct evidence and target adjacency/feature matrices as in algorithm 1. For both evidence and target matrices, the unspecified $*$ link values are translated into into zero. This follows common practice in graph neural networks [11]. One justification for this approach is that many graph message passing algorithms propagate information only along existing links, so setting the links to zero means that the unknown links have no contribution to the final prediction. The attribute matrices and binary adjacency matrices can be processed by graph neural networks.

- $\boldsymbol{A}^{\boldsymbol{E}} = \begin{bmatrix} 1 & * & * \\ * & 1 & 1 \\ * & 1 & 1 \end{bmatrix}$
- $\boldsymbol{A}^{\boldsymbol{E},\boldsymbol{Y}} = \begin{bmatrix} 1 & * & 1 \\ * & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- $\boldsymbol{X}^{\boldsymbol{E}} = \begin{bmatrix} 0.2 & 0.4 & 0.8 & 0.21 \\ 0.67 & 0.2 & 0.34 & 0.1 \\ * & * & * & * \end{bmatrix}$
- $\boldsymbol{X}^{\boldsymbol{E},\boldsymbol{Y}} = \begin{bmatrix} 0.2 & 0.4 & 0.8 & 0.21 \\ 0.67 & 0.2 & 0.34 & 0.1 \\ * & * & * & * \end{bmatrix}$

Algorithm 1 Construct evidence and target adjacency and attribute matrices

```
Input: Given a query $\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{E} = \boldsymbol{e})$
Output: $\boldsymbol{A^E}$ , $\boldsymbol{A^{E,Y}}$, $\boldsymbol{X^E}$, $\boldsymbol{X^{E,Y}}$
    for all the links $[u, v]$ do
        if $\boldsymbol{e}$ contains $\boldsymbol{A}[u, v] = 1(0)$ then
            $\boldsymbol{A^E}[u, v] = 1(0)$
        else
            $\boldsymbol{A^E}[u, v] = *$
        end if
        if either $\boldsymbol{e}$ or $\boldsymbol{y}$ contains $\boldsymbol{A}[u, v] = 1(0)$. then
            $\boldsymbol{A^{E,Y}}[u, v] = 1(0)$
        else
            $\boldsymbol{A^{E,Y}}[u, v] = *$
        end if
    end for
    for All the nodes $u$ in $\boldsymbol{e}$ do
        if $\boldsymbol{e}$ contains $\boldsymbol{X}[u] = \boldsymbol{x}$ then
            $\boldsymbol{X^E}[u] = \boldsymbol{x}[u]$
        else
            $\boldsymbol{X^E}[u] = *$
        end if
        if either $\boldsymbol{e}$ or $\boldsymbol{y}$ contains $\boldsymbol{X}[u] = \boldsymbol{x}$ then
            $\boldsymbol{X^{E,Y}}[u] = \boldsymbol{x}[u]$
        else
            $\boldsymbol{X^{E,Y}}[u] = *$
        end if
    end for
```

3.2.3 From CVGAE to Query Probabilities

Sohn et al. describe three methods for applying a given CVAE to compute conditional probabilities: deterministic, Monte Carlo, and importance sampling [20]. We evaluate the same methods for graph queries. The following defines the inference methods.

Recall that $q_\varphi(\boldsymbol{Z}|\boldsymbol{A^E}, \boldsymbol{X^E})$ computes the prior mean $\mu[u]$ and covariance $\sigma[u]$ for each node u. So we have $p(\boldsymbol{Z}[u]|\boldsymbol{E} = \boldsymbol{e}) \sim N(\mu[u], \sigma[u])$.

Deterministic inference computes the mean of the node embeddings using the prior network, then applies the decoder to the mean to compute a target probability:

$$\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}|\boldsymbol{E} = \boldsymbol{e}) \approx \prod_{i=1}^{m} p(Y_i = y_i|\mu[u_i], \mu[v_i]). \tag{3.4}$$

Monte Carlo inference samples S node embeddings from the prior network, then applies the decoder to each sample. For every target link $[u, v]$ in $\boldsymbol{y}$, the prior joint distribution over both node embeddings is given by:

$$p(\boldsymbol{Z}[u], \boldsymbol{Z}[v] | \boldsymbol{E} = \boldsymbol{e}) \approx p(\boldsymbol{Z}[u] | \boldsymbol{E} = \boldsymbol{e}) \times p(\boldsymbol{Z}[v] | \boldsymbol{E} = \boldsymbol{e}) \sim N(\mu[u], \sigma[u]) \times N(\mu[v], \sigma[v]).$$

We now sample from the joint prior distribution to estimate:

$$\begin{aligned}&\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{E} = \boldsymbol{e}) = \frac{1}{S} \sum_{s=1}^{S} \prod_{i=1}^{m} p(Y_i = y_i | \boldsymbol{Z}^s[u_i], \boldsymbol{Z}^s[v_i]) \\ &\text{where } \boldsymbol{Z}^s[u_i], \boldsymbol{Z}^s[v_i] \sim p(\boldsymbol{Z}[u_i], \boldsymbol{Z}[v_i] | \boldsymbol{E} = \boldsymbol{e}).\end{aligned}$$

Importance Sampling inference samples embeddings from the recognition network and applies importance weights to each sample.

The original CVAE paper proposed as an inference method an importance sampling scheme that uses the approximate posterior computed from the recognition network as the proposal distribution [20]. We use the graph encoder $q_\varphi(\boldsymbol{A}^{\boldsymbol{E},\boldsymbol{Y}}, \boldsymbol{X}^{\boldsymbol{E},\boldsymbol{Y}})$ as a recognition network to compute the posterior mean and covariance $\mu^q[u], \sigma^q[u]$ for each node u. So we have:

$$q(\boldsymbol{Z}[u] | \boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Y} = \boldsymbol{y}) \sim N(\mu^q[u], \sigma^q[u]). \tag{3.5}$$

The posterior joint distribution over both node embeddings is given by:

$$\begin{aligned}&q(\boldsymbol{Z}[u], \boldsymbol{Z}[v] | \boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Y} = \boldsymbol{y}) = q(\boldsymbol{Z}[u] | \boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Y} = \boldsymbol{y}) \times q(\boldsymbol{Z}[v] | \boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Y} = \boldsymbol{y}) \\ &\sim N(\mu^q[u], \sigma^q[u])) \times N(\mu^q][v], \sigma^q[v])).\end{aligned}$$

The importance sampling scheme of Sohn et al. samples from the joint posterior distribution to estimate:

$$\begin{aligned}&\mathbb{P}(\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{E} = \boldsymbol{e}) \approx \frac{1}{S} \sum_{s=1}^{S} \prod_{i=1}^{m} p(Y_i = y_i | \boldsymbol{Z}^s[u_i], \boldsymbol{Z}^s[v_i]) \times \frac{p(\boldsymbol{Z}^s[u_i], \boldsymbol{Z}^s[v_i] | \boldsymbol{E} = \boldsymbol{e})}{q(\boldsymbol{Z}^s[u_i], \boldsymbol{Z}^s[v_i] | \boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Y} = \boldsymbol{y})} \\ &\text{where } \boldsymbol{Z}^s[u_i], \boldsymbol{Z}^s[v_i] \sim q(\boldsymbol{Z}[u_i], \boldsymbol{Z}[v_i] | \boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Y} = \boldsymbol{y}).\end{aligned} \tag{3.6}$$

Chapter 4

Experiments

We describe benchmark datasets used and the test queries prepared, then the compare query answering methods and their evaluation metrics.

4.1 Datasets

To evaluate all the methods we utilize 6 datasets used in previous studies of generative models [8, 11, 26]. Table 4.1 presents statistics of these datasets. Edge density refers to the number of links in a graph divided by the total number of possible links in that graph. It is a measure of the connectedness or compactness of the graph. The average node degree in a graph is the average of the degrees of all the nodes in the graph.

- **Cora** is a citation dataset that consists of nodes that represent machine-learning papers divided into seven classes and links that represent citation between them. This dataset has 5,429 links, 2,708 nodes with an average node degree 3.8. [18].
- **ACM** is a citation dataset. It has three types of nodes (paper, author, and venue) and four types of links. This dataset has 18,929 links, 8,993 nodes with an average node degree 2.209 [27].
- **IMDb** is a movie dataset with three types of nodes (movies, actors, and directors) and it uses genre of movies as their labels. This dataset has 19,120 links, 12,772 nodes with an average node degree 2.9 [27].
- **CiteSeer** is also a citation dataset which consists of nodes that represent machine-learning papers divided into six classes and links that represent citation between them. This dataset has 4,732 links, 3,327 nodes with an average node degree 2.7 [18].
- **Photo & Computers** are datasets from the Amazon co-purchase graph [13], where nodes represent goods, links indicate that two goods are frequently bought together, node features are bag-of-words encoded product reviews, and class labels are given by the product category. For Photo dataset the number of links is 238,162 links, and

Dataset	**Nodes**	**Links**	**Edge Density**	**Average Node Degree**
Cora [11]	2,708	5,429	0.00074	3.8
ACM [26]	8,993	18,929	0.00093	2.2
IMDb [26]	12,772	19,120	0.00046	2.9
CiteSeer [11]	3,327	4,732	0.00171	2.7
Photo [8]	7,650	238,162	0.01629	36.7
Computers [8]	13,752	491,722	0.01041	36.7

Table 4.1: Statistics of the datasets.

number of nodes is 7,650 with an average node degree of 36.7. While Computers dataset has 491,722 links, 13,752 nodes with an average node degree 36.7 [8].

4.2 Data Preprocessing

We define an $N \times N$ adjacency matrix $\boldsymbol{A}$ such that its element $\boldsymbol{A}[u, v] = 1$ if there is an link from node u to node v and 0 otherwise. We also define a node feature matrix $\boldsymbol{X}$, which is of size N by the number of features for each node. Following previous link prediction studies [11], we add self-loops and make all links undirected (i.e., if the training data contains an adjacency, $v \to u$, we ensure it also contains $u \to v$).

4.3 Test Query Design

Since our input data are undirected graphs, we enforce the constraint that if a query target contains a link indicator $A[u, v]$, the evidence does not contain $A[v, u]$. We split the input graph nodes into 80% seen nodes and 20% unseen test nodes.

Single-Link Queries. Single-link queries specify a single link target (i.e., $m = 1$). For fully-inductive single-link queries, we randomly select 100 positive and 100 negative test links from the test node graph as targets. Then we predict each test link given all the other links in the test node graph. In the semi-inductive setting, we select 100 positive and 100 negative links that at least one end of the link in the test set. Then we predict each test link given all the other links in the input graph.

Multi-Link Queries. Multi-link test queries specify a set of links to be predicted jointly. In the fully-inductive setting, we predict, for each test node u, which other test nodes v are its neighbors. We randomly select a set of 100 test nodes as target nodes. For each target test node, we predict jointly which other test nodes are its neighbors. In the semi-inductive setting, we predict, for each test node, which other nodes (both training and test) are its neighbors given all the other links in the input graph.

Other Test Queries The query language of Section 3.1 is expressive and allows us to formulate various other types of useful queries. Examples include the following.

- In practice, the goal is often to predict a set of target links from a relatively small set of relevant evidence links. For example, we may use only observed links in the k-hop neighborhood of target links as evidence.
- Transductive queries aim to predict links among observed nodes.
- We could predict links based on attributes alone, as in [8].
- We could remove node features (by setting them to a constant value) to assess their usefulness.

The learning and inference procedures in this thesis support these and other kinds of queries. We use the inductive test queries described above to limit the scope of our evaluation, and because they are the most relevant to current link prediction research. Using inference from a single model for other link prediction queries is a topic for future research.

4.4 Metrics

For all of our evaluations we report the following metrics:

- **Area Under the Curve (AUC)** measures the area underneath the Receiver Operating Characteristic (ROC) curve. This metric measures the performance of the model across all possible classification thresholds.
- **Average Precision (AP)** measures the weighted mean of precisions over all possible classification thresholds.
- **Precision** measures how many observations predicted as positive are correctly predicted.
- **Recall** measures from all positive observations how many of them were classified as positive.
- **Hit-Rate (HR)** is computed as follows: (1) For each method, find the set T the of top 20% links as ranked by the method. (2) Find the number of ground-truth links in T. (3) Divide by the total number of links in T.

For single-link queries, the ranking metrics become standard classification metrics. The metrics are computed over all single-link queries. In the case of multi-link predictions, for each link prediction query, we apply the same metrics, but on a per-query basis, that is, restricted to the links mentioned in the target query. We report the mean of each metric over all queries.

4.5 CVAE-Based Methods

We evaluate CVAE inference based on two VGAE architectures trained with the ELBO (3.1). As a strong *decoder*, we utilize a non-linear node transformation function, Stochastic Block Model (SBM), [14]. SBM is a popular model for community detection tasks, where the goal is to identify the underlying groups or communities of nodes in a network. It is also commonly used as a benchmark model for evaluating the performance of graph analysis algorithms and is defined as:

$$p_\theta(\boldsymbol{Z}[u], \boldsymbol{Z}[v]) = \boldsymbol{Z}[u]^\top \Lambda \boldsymbol{Z}[v], \tag{4.1}$$

where $\Lambda \in \mathbb{R}^d \times \mathbb{R}^d$, is the block matrix for d blocks in the Stochastic Block Model [6, Sec.8.3].

We compare two well-known *encoder* GNN architectures. The output dimension for both encoders is 128.

- **VGAE-GCN** Graph Convolutional Neural Network(GCN) is a popular encoder [11]. GCNs use convolutional operations, similar to those in image processing, to aggregate information from a node's neighbors and update the node's representation. This makes them especially useful for tasks such as node classification and link prediction in graph-structured data.

- **VGAE-GAT** Graph Attention Network(GAT) is based on the idea of attention mechanisms, where each node in the graph attends to its neighbors with different levels of importance during message passing [22]. GATs use multi-head attention to compute the importance of each neighbor based on learned weights. GATs have been shown to achieve state-of-the-art performance on various graph-related tasks, such as node classification and link prediction.

4.6 Importance Sampling With the Recognition Network

While both approximate posteriors and proposal distributions are traditionally denoted by the symbol q, using the posterior as a proposal is problematic because it can lead to information leakage. For PQGs, if we sample embeddings from $q(\boldsymbol{Z}[u], \boldsymbol{Z}[v]|\boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Y} = \boldsymbol{y})$, they will incorporate the information from $\boldsymbol{Y} = \boldsymbol{y}$, making it easy for the decoder the predict that $\boldsymbol{Y} = \boldsymbol{y}$, the ground-truth value. We used 30 samples for both normalized and unnormalized methods.

To confirm this diagnosis further, we evaluated a **normalized importance sampling scheme for single-link queries** without information leakage: first estimate a probability score for $\boldsymbol{Y} = 1$, and another for $\boldsymbol{Y} = 0$. Then we apply softmax to normalize the scores for a final probability prediction. In symbols, the normalized IS estimates are computed as follows.

$$\begin{aligned}
P_0 &= \frac{1}{S}\sum_{s=1}^{S} p(Y=1|\boldsymbol{Z}^s[u],\boldsymbol{Z}^s[v]) \times \frac{p(\boldsymbol{Z}^s[u],\boldsymbol{Z}^s[v]|\boldsymbol{E}=\boldsymbol{e})}{q(\boldsymbol{Z}^s[u],\boldsymbol{Z}^s[v]|\boldsymbol{E}=\boldsymbol{e},Y=0)} \\
P_1 &= \frac{1}{S}\sum_{s=1}^{S} p(Y=1|\boldsymbol{Z}^s[u],\boldsymbol{Z}^s[v]) \times \frac{p(\boldsymbol{Z}^s[u],\boldsymbol{Z}^s[v]|\boldsymbol{E}=\boldsymbol{e})}{q(\boldsymbol{Z}^s[u],\boldsymbol{Z}^s[v]|\boldsymbol{E}=\boldsymbol{e},Y=1)} \\
\mathbb{P}(Y=1|\boldsymbol{E}=\boldsymbol{e}) &= \mathit{softmax}(P_0,P_1)
\end{aligned} \tag{4.2}$$

4.7 Experimental Setup

We used the same VGAE implementation as the original VGAE paper [11]. We trained our models using 100 epochs, which was sufficient to ensure convergence of the training loss. The latent embedding dimension was set to 128 [11]. We reported results using GCN and GAT encoders. The GCN encoder is a two-layer GCN[64, 128]. The GAT encoder is also a two layer GAT with dimensions [32,128]. For each GAT layer, we used 4 heads as suggested by the original paper [22]. GAT layers concatenate embeddings for each head so we chose a smaller initial embedding dimension.

4.8 Baselines

1. **SEAL** is a well-known method which was introduced for transductive link prediction [29]. We train SEAL inductively which is similar to the approach of GraIL [21], used for attribute graphs rather than knowledge graphs.

2. **DEAL** was introduced for both transductive and inductive cold-start link prediction tasks [8]. Since this method only takes the attribute matrix as input, we apply it only in the fully-inductive setting.

3. **Classifier methods** use an inductive encoder, then train an SBM link prediction model that takes the trained node embeddings as input. Comparing classification methods with deterministic CVAE inference evaluates the importance of end-to-end training of the link predictor.

    - **Classifier-GraphSage** Uses the well-known GraphSage embeddings, trained with the unsupervised objective [7].
    - **Classifier-GAT** Uses the GAT embedding system, trained with the same micro-F$_1$ score as in the original paper [22].

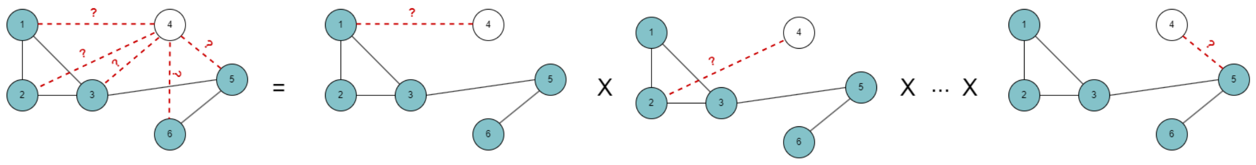


Figure 4.1: Predicting multiple links by computing independent link probabilities for each target link.

4.8.1 Baseline Setup

For all 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 nodes in all experiments.

We extend the single-link prediction methods to predict multi-links by computing independent link probabilities for each target link given the evidence, then multiplying them to approximate the multi-link joint probability. Figure 4.1 shows this process or in symbols:

$$\mathbb{P}(\boldsymbol{A}[u, v_1], \ldots, \boldsymbol{A}[u, v_m] | \boldsymbol{E}) \approx \prod_{i=1}^{m} \mathbb{P}(\boldsymbol{A}[u, v_i] | \boldsymbol{E}). \tag{4.3}$$

where u is a target node and $v_1 \ldots v_m$ are all the possible neighbors of u.

To compute ranking metrics for neighborhood queries, we require a score to each potential neighbor u of a test node v. For the baseline methods, we use the probability of the link $\boldsymbol{A}[u, v]$, computed either by independent prediction. For the VGAE sampling methods, we use the link probability averaged over all samples (cf. Section 3.2.3).

Chapter 5

Results

We report five Area Under Curve (AUC), Average Precision (AP), precision, recall, and Hit-Rate (HR) metrics [1] for VGAE and our baselines on six datasets. The results were computed with a classification threshold of 0.5 for precision, recall, and HR. In all the tables, the first best result for each setting is indicated in **bold**, and the second best result is underlined.

5.1 Single-Link Queries

AUC and AP. Tables 5.1 and 5.2 show the results for single-link prediction results on AUC and AP respectively. All the VGAE models perform better than the baselines in almost all the datasets. They all get 0.8 or higher performance in both AUC and AP. There is no clear distinction between deterministic and Monte Carlo sampling methods. We believe this is because we used 30 as the number of samples for Monte Carlo due to the limitations of our resources while it has been suggested to use at least 1000 samples [3]. We can also observe that VGAE models with GAT encoder perform better than or at least as good the VGAE models with GCN encoders. This is because GAT encoders have an attention mechanism that puts more weight on the important links. We can see that GAT encoders perform slightly worse for the ACM and CiteSeer datasets compared to the GCN encoders. This is because while GAT encoders are designed to handle graphs with varying node degrees, but if the degree distribution is highly skewed or if the graph has a high degree of sparsity, the GAT encoder may not be able to effectively capture important patterns in the data. As shown in table 4.1, ACM and CiteSeer have the smallest average node degree. As expected the embed+classify methods perform much worse than all the other methods which shows the value of end-to-end training. It is also noticeable that while SEAL performs better than the Embed+Classify methods, it cannot generalize well to the fully-inductive link prediction settings and its performance drops by a lot in almost all the datasets. Its performance is also dependant on the density of the graphs. For example, it performs much better with the Photo and Computers dataset which have a higher edge density. As with single-link queries,

		AUC					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	0.855	<u>0.957</u>	0.850	0.937	0.942	0.954
	VGAE-GCN (Monte Carlo)	0.863	**0.961**	<u>0.857</u>	**0.941**	<u>0.952</u>	0.948
	VGAE-GAT (Deterministic)	**0.894**	0.906	**0.860**	<u>0.940</u>	0.944	0.950
	VGAE-GAT (Monte Carlo)	<u>0.892</u>	0.900	<u>0.857</u>	0.935	**0.968**	<u>0.960</u>
	Extended SEAL	0.756	0.666	0.854	0.627	0.924	**0.972**
	Classify (GAT)	0.667	0.870	0.796	0.731	0.796	0.835
	Classify (GraphSage)	0.561	0.593	0.492	0.504	0.552	0.444
Fully	VGAE-GCN (Deterministic)	0.765	**0.984**	0.885	0.910	0.859	0.923
	VGAE-GCN (Monte Carlo)	0.765	<u>0.976</u>	<u>0.887</u>	0.904	0.898	**0.961**
	VGAE-GAT (Deterministic)	**0.792**	0.935	0.865	<u>0.917</u>	**0.973**	0.935
	VGAE-GAT (Monte Carlo)	<u>0.790</u>	0.947	0.859	**0.923**	<u>0.965</u>	<u>0.954</u>
	Extended SEAL	0.695	0.645	0.636	0.579	0.840	0.939
	Classify (GAT)	0.525	0.764	0.671	0.700	0.802	0.733
	Classify (GraphSage)	0.695	0.564	0.440	0.521	0.572	0.515
	DEAL	0.733	0.957	**0.948**	0.828	0.861	0.829

Table 5.1: AUC results for single-link prediction.

DEAL is highly accurate on the node-heterogeneous datasets ACM and IMDb where nodes can have different types.

Precision. Table 5.3 shows the result of precision performance of the models for single-link prediction. Same as AUC and AP, precision is high in all the VGAE models with the lowest value being 80.2% for Computers in the VGAE-GCN (Monte Carlo) semi-inductive case and the highest being 100% for VGAE-GAT (Deterministic) fully-inductive case . SEAL and the embed+classify methods perform much worse, specifically ranging from 40 to high 70s except for SEAL in photos and Computers. As explained before, this is because these two datasets are denser and SEAL is very dependent on the existence of edges. DEAL reports 100% for all datasets in the case of fully-inductive single-link prediction.. This means that all the links that DEAL predicts to be one, actually exist.

Recall. Table 5.4 shows the result of recall performance of the models for single-link prediction. Compared to precision, recall is lower in almost all the models. Especially when generalized from semi-link prediction to fully-link prediction.. Table 5.4 confirms that DEAL is conservative and only predicts the links that are highly likely positive, with recall of under 20% in most of the datasets. The VGAE methods show higher variability on recall, and thus appear to be sensitive to the 0.5 classification threshold. Their recall scores are generally competitive with SEAL and score higher than the embed+classify methods.

		AP					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	0.891	<u>0.965</u>	0.827	<u>0.953</u>	0.911	0.940
	VGAE-GCN (Monte Carlo)	0.897	**0.968**	0.836	**0.955**	<u>0.952</u>	0.925
	VGAE-GAT (Deterministic)	**0.912**	0.923	**0.851**	<u>0.953</u>	0.938	0.942
	VGAE-GAT (Monte Carlo)	<u>0.908</u>	0.917	<u>0.844</u>	0.951	**0.964**	<u>0.956</u>
	Extended SEAL	0.829	0.737	**0.851**	0.748	0.931	**0.976**
	Classify (GAT)	0.676	0.866	0.786	0.754	0.800	0.837
	Classify (GraphSage)	0.526	0.605	0.527	0.511	0.551	0.481
Fully	VGAE-GCN (Deterministic)	0.790	**0.983**	0.900	**0.932**	0.870	0.912
	VGAE-GCN (Monte Carlo)	0.775	<u>0.973</u>	<u>0.901</u>	<u>0.922</u>	0.886	0.951
	VGAE-GAT (Deterministic)	<u>0.806</u>	0.944	0.882	**0.932**	<u>0.969</u>	0.941
	VGAE-GAT (Monte Carlo)	**0.812**	0.956	0.878	**0.932**	**0.952**	<u>0.955</u>
	Extended SEAL	0.787	0.701	0.647	0.697	0.897	**0.959**
	Classify (GAT)	0.534	0.781	0.667	0.695	0.799	0.757
	Classify (GraphSage)	0.486	0.554	0.471	0.532	0.627	0.549
	DEAL	0.780	0.967	**0.957**	0.861	0.852	0.844

Table 5.2: AP results for single-link prediction.

Hit-Rate. Based on table 5.5, we find that DEAL achieves a very high HR through conservative link prediction. On single-link queries, both VGAE methods and SEAL achieve a high HR. The classification methods score worse than either the VGAE methods or SEAL. Comparing semi-inductive and fully-inductive link prediction methods, the VGAE methods tend to generalize better than SEAL. For example on CiteSeer single-link queries, VGAE-GCN(Deterministic) stays the same at 100% , while SEAL declines from 97.4% to 83.8%.

5.2 Multi-Link Queries

AUC and AP. Based on tables 5.6 and 5.7, all VGAE methods do very well, with both AUC and AP ranging from 0.7 to above 0.9 on almost all the datasets for both semi and fully inductive multi-link predictions. The Embed+Classify methods again do much worse, typically below 0.6. The VGAE methods also outperform SEAL on every dataset except for AUC on Cora fully inductive multi-link prediction. The AUC of SEAL is exceptionally high on Cora and exceptionally low on ACM, which shows that its multi-link is highly data-dependent. Similarly its generalization ability depends strongly on the data: excellent on Cora and good on Photo and Computers, and very poor on ACM, IMDb, and CiteSeer. This spread likely arises because test graphs vary in their density, and SEAL does best with highly dense graphs. The VGAE models are performing better than DEAL in both AUC and AP in all the datasets except for IMDb.

		Precision(%)					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	95.0	97.6	85.2	93.0	90.1	85.6
	VGAE-GCN (Monte Carlo)	96.7	**98.5**	84.6	**93.9**	87.8	87.9
	VGAE-GAT (Deterministic)	**96.9**	91.7	92.2	93.0	85.2	80.2
	VGAE-GAT (Monte Carlo)	94.1	92.0	**93.7**	92.7	86.2	86.6
	Extended SEAL	67.2	55.9	57.1	69.8	**97.0**	**90.7**
	Classify (GAT)	65.2	76.1	76.4	69.4	73.4	76.7
	Classify (GraphSage)	48.3	75.0	54.8	45.4	55.5	42.8
Fully	VGAE-GCN (Deterministic)	93.9	97.7	92.6	94.7	90.3	86.3
	VGAE-GCN (Monte Carlo)	88.6	96.3	88.5	90.9	81.4	95.2
	VGAE-GAT (Deterministic)	90.3	97.3	**100.0**	94.4	88.0	88.7
	VGAE-GAT (Monte Carlo)	93.5	96.1	97.6	91.7	86.2	89.7
	Extended SEAL	72.0	58.8	57.6	75.7	99.5	97.8
	Classify (GAT)	50.0	79.3	71.1	72.9	88.4	78.9
	Classify (GraphSage)	53.3	47.6	38.4	55.5	73.3	52.7
	DEAL	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Table 5.3: Precision results for single-link prediction.

		Recall(%)					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	57.0	81.0	46.0	**80.0**	64.0	89.0
	VGAE-GCN (Monte Carlo)	59.0	79.0	44.0	77.0	87.0	87.0
	VGAE-GAT (Deterministic)	63.0	66.0	47.0	**80.0**	92.0	**97.0**
	VGAE-GAT (Monte Carlo)	64.0	69.0	45.0	77.0	**99.0**	**97.0**
	Extended SEAL	**69.9**	81.1	**99.3**	54.6	78.5	95.2
	Classify (GAT)	47.0	**83.0**	65.0	50.0	61.0	66.0
	Classify (GraphSage)	15.0	12.0	17.0	10.0	15.0	15.0
Fully	VGAE-GCN (Deterministic)	31.0	**84.0**	50.0	**71.0**	65.0	69.0
	VGAE-GCN (Monte Carlo)	35.0	70.0	44.0	68.0	63.0	85.0
	VGAE-GAT (Deterministic)	28.0	72.0	43.0	67.0	**95.0**	81.0
	VGAE-GAT (Monte Carlo)	29.0	73.0	40.0	66.0	94.0	**87.0**
	Extended SEAL	**56.0**	69.5	**83.7**	41.8	54.1	84.2
	Classify (GAT)	14.0	46.0	42.0	27.0	46.0	45.0
	Classify (GraphSage)	16.0	10.0	10.0	10.0	22.0	19.0
	DEAL	03.0	60.0	52.0	03.0	17.0	10.0

Table 5.4: Recall results for single-link prediction.

		Hit-Rate(%)					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	**100.0**	**100.0**	90.0	**100.0**	**100.0**	97.5
	VGAE-GCN (Monte Carlo)	97.5	**100.0**	**92.5**	**100.0**	**100.0**	95.0
	VGAE-GAT (Deterministic)	97.5	**100.0**	**92.5**	**100.0**	97.5	**100.0**
	VGAE-GAT (Monte Carlo)	97.5	**100.0**	**92.5**	**100.0**	**100.0**	97.5
	Extended SEAL	**98.1**	84.3	<u>90.2</u>	<u>97.4</u>	<u>98.1</u>	<u>99.5</u>
	Classify (GAT)	65.0	<u>90.0</u>	82.5	85.0	73.4	76.7
	Classify (GraphSage)	48.3	61.9	56.6	45.4	55.5	42.8
Fully	VGAE-GCN (Deterministic)	93.9	**100.0**	<u>97.5</u>	**100.0**	95.0	92.5
	VGAE-GCN (Monte Carlo)	88.6	**100.0**	**100.0**	**100.0**	95.0	97.5
	VGAE-GAT (Deterministic)	90.3	<u>97.5</u>	**100.0**	**100.0**	97.5	97.5
	VGAE-GAT (Monte Carlo)	93.5	**100.0**	**100.0**	**100.0**	97.5	**100.0**
	Extended SEAL	<u>99.2</u>	77.7	77.3	<u>83.8</u>	<u>99.6</u>	<u>99.7</u>
	Classify (GAT)	50.0	82.5	77.5	75.0	88.4	78.9
	Classify (GraphSage)	53.3	47.6	38.4	55.5	73.3	52.7
	DEAL	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Table 5.5: Hit-Rate @ 20% results for single-link prediction.

Precision. Table 5.8 shows the result of precision performance of the models for multi-link prediction. All methods do worse on the harder multi-link problems compared to single-link prediction specially on the Cora and IMDb datasets. This could be because the link density of these datasets are very small and removing all the neighbors of a node can impact the performance of the models by a lot. As before the VGAE methods perform better than SEAL and embed+classify methods. This is while DEAL gets 100% precision on all the datasets due to being very conservative with predicting links.

Recall. Based on table 5.9, recall is also lower in comparison to the single-link prediction setup. Missing a lot of links in the multi-link prediction makes the methods much weaker in predicting all the positive links. We can see that VGAE methods are performing better than the baselines on most of the datasets while SEAL is performing better for Cora, ACM, and IMDb. Recall for DEAL is also very low (below 5% in 3 datasets) which means it misses a large proportion of actual positive links.

Hit-Rate. Table 5.10 shows the result of Hit-Rate for multi-link prediction methods. The VGAE methods still perform well in the multi-link setup. We find that DEAL achieves a very high HR through conservative link prediction. SEAL tends to deteriorate much more for multi-link queries in comparison to single-link prediction. The embed+classify methods score worse than both the VGAE methods and SEAL. Comparing semi-inductive and fully-inductive methods, the VGAE methods tend to generalize better than SEAL.

		AUC					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	0.721	<u>0.958</u>	0.827	**0.951**	0.757	0.761
	VGAE-GCN (Monte Carlo)	0.711	**0.959**	**0.836**	0.930	<u>0.802</u>	**0.841**
	VGAE-GAT (Deterministic)	<u>0.737</u>	0.925	0.814	<u>0.932</u>	0.759	0.615
	VGAE-GAT (Monte Carlo)	**0.749**	0.937	0.815	0.925	**0.812**	<u>0.775</u>
	Extended SEAL	0.693	0.681	<u>0.830</u>	0.679	0.506	0.450
	Classify (GAT)	0.491	0.699	0.632	0.470	0.453	0.380
	Classify (GraphSage)	0.543	0.535	0.467	0.512	0.528	0.507
Fully	VGAE-GCN (Deterministic)	0.639	0.959	<u>0.860</u>	<u>0.923</u>	0.783	0.786
	VGAE-GCN (Monte Carlo)	0.631	<u>0.960</u>	0.835	**0.929**	**0.817**	**0.858**
	VGAE-GAT (Deterministic)	0.670	0.946	0.830	0.898	0.749	0.683
	VGAE-GAT (Monte Carlo)	<u>0.691</u>	**0.967**	0.830	0.922	<u>0.799</u>	<u>0.822</u>
	Extended SEAL	**0.821**	0.260	0.336	0.375	0.657	0.687
	Classify (GAT)	0.394	0.621	0.587	0.450	0.537	0.468
	Classify (GraphSage)	0.557	0.513	0.459	0.532	0.554	0.541
	DEAL	0.678	0.953	**0.935**	0.837	0.759	0.757

Table 5.6: AUC results for multi-link prediction.

In conclusion, our empirical results show that VGAE inference from a single model provides accurate probabilities and good generalization for inductive link prediction queries.

5.3 Importance Sampling Results

Unnormalized Importance Sampling (UIS) uses the estimation formula of Equation (3.6). Normalized Importance Sampling (NIS) uses the estimation formula of Equation (4.2).

For Importance sampling, we report AUC and AP results both for normalized and unnormalized methods for single-link prediction. Normalizing multi-link predictions is a much more complex task which requires to set the value of all the target links to zero and one independently and run the model separately for all the possible combinations. This process of running the model multiple times is going to be costly. Future work can focus on developing more efficient and effective methods for normalizing the importance sampling inference for multi-link predictions.

Table 5.11 shows that unnormalized importance sampling method performs very well on AUC and much higher compared to table 5.1 where deterministic and Monte Carlo sampling methods were used. For unnormalized importance sampling the performance on AUC for both GCN and GAT encoders on all the datasets are above 0.90 (except for VGAE-GAT semi-link prediction on Computers). Comparing the unnormalized importance sampling to normalized importance sampling shows that importance sampling does in fact has some

		AP					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	0.759	<u>0.963</u>	<u>0.849</u>	**0.954**	0.735	<u>0.743</u>
	VGAE-GCN (Monte Carlo)	0.755	**0.964**	**0.861**	0.941	**0.788**	**0.824**
	VGAE-GAT (Deterministic)	0.757	0.938	0.837	0.943	0.705	0.590
	VGAE-GAT (Monte Carlo)	**0.764**	0.947	0.841	<u>0.945</u>	<u>0.778</u>	0.721
	Extended SEAL	<u>0.761</u>	0.502	<u>0.849</u>	0.711	0.658	0.627
	Classify (GAT)	0.599	0.748	0.708	0.584	0.481	0.458
	Classify (GraphSage)	0.643	0.652	0.619	0.642	0.566	0.542
Fully	VGAE-GCN (Deterministic)	0.718	0.969	<u>0.893</u>	<u>0.949</u>	0.781	0.787
	VGAE-GCN (Monte Carlo)	0.711	<u>0.971</u>	0.877	**0.952**	**0.848**	**0.851**
	VGAE-GAT (Deterministic)	0.747	0.962	0.875	0.930	0.732	0.687
	VGAE-GAT (Monte Carlo)	**0.756**	**0.974**	0.875	0.946	0.790	0.802
	Extended SEAL	<u>0.748</u>	0.545	0.577	0.544	0.741	0.775
	Classify (GAT)	0.628	0.737	0.743	0.655	0.583	0.534
	Classify (GraphSage)	0.723	0.678	0.659	0.707	0.622	0.616
	DEAL	0.651	0.968	**0.952**	0.864	<u>0.836</u>	<u>0.844</u>

Table 5.7: AP results for multi-link prediction.

information leakage. The normalized version of importance sampling avoids the information leakage, but performs slightly worse in comparison to the other sampling methods.

		Precision(%)					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	44.5	<u>88.8</u>	**73.3**	<u>81.1</u>	70.6	69.2
	VGAE-GCN (Monte Carlo)	40.5	**91.0**	<u>70.3</u>	**82.6**	<u>81.6</u>	**94.4**
	VGAE-GAT (Deterministic)	<u>45.0</u>	81.8	64.7	80.6	74.3	<u>76.7</u>
	VGAE-GAT (Monte Carlo)	44.6	83.7	65.3	74.3	**98.8**	58.0
	Extended SEAL	**50.3**	42.6	38.5	50.1	04.5	11.0
	Classify (GAT)	28.7	52.4	39.7	11.4	26.0	28.7
	Classify (GraphSage)	33.1	31.2	35.6	26.2	54.2	52.0
Fully	VGAE-GCN (Deterministic)	<u>34.3</u>	<u>88.6</u>	61.3	<u>71.8</u>	72.6	69.2
	VGAE-GCN (Monte Carlo)	30.9	83.7	<u>62.4</u>	69.8	87.1	<u>96.2</u>
	VGAE-GAT (Deterministic)	28.2	83.0	54.6	66.5	83.4	80.3
	VGAE-GAT (Monte Carlo)	28.7	78.5	55.0	67.0	<u>99.9</u>	64.0
	Extended SEAL	29.5	37.5	42.3	18.8	05.0	11.5
	Classify (GAT)	13.3	36.2	29.4	08.9	36.2	34.8
	Classify (GraphSage)	27.2	21.3	19.3	21.5	61.2	62.4
	DEAL	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Table 5.8: Precision results for multi-link prediction.

		Recall(%)					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	22.5	74.7	**42.5**	**76.1**	**40.0**	**37.6**
	VGAE-GCN (Monte Carlo)	21.2	<u>77.2</u>	<u>41.9</u>	<u>74.9</u>	<u>39.6</u>	<u>27.9</u>
	VGAE-GAT (Deterministic)	<u>25.3</u>	63.1	39.8	73.4	20.9	02.7
	VGAE-GAT (Monte Carlo)	24.0	62.8	38.5	67.5	13.6	01.9
	Extended SEAL	**47.3**	**82.4**	21.0	53.3	02.5	11.2
	Classify (GAT)	16.9	44.6	36.7	05.9	10.1	13.7
	Classify (GraphSage)	11.8	13.6	13.1	08.1	19.2	20.1
Fully	VGAE-GCN (Deterministic)	**21.3**	**84.0**	44.5	**65.7**	**44.6**	**41.0**
	VGAE-GCN (Monte Carlo)	20.2	79.0	<u>44.6</u>	<u>65.1</u>	<u>40.8</u>	<u>30.7</u>
	VGAE-GAT (Deterministic)	18.1	70.5	40.2	61.5	23.6	12.3
	VGAE-GAT (Monte Carlo)	18.1	68.7	37.8	60.5	13.2	05.7
	Extended SEAL	17.0	70.6	**79.4**	16.3	13.3	05.1
	Classify (GAT)	11.8	32.3	29.3	06.0	11.2	15.5
	Classify (GraphSage)	18.8	14.8	10.9	13.2	24.0	24.4
	DEAL	06.0	67.8	50.0	04.0	09.8	12.5

Table 5.9: Recall results for multi-link prediction.

		Hit-Rate(%)					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (Deterministic)	46.3	90.4	**74.0**	83.3	75.6	71.8
	VGAE-GCN (Monte Carlo)	42.1	**93.2**	71.1	**84.6**	84.7	**95.8**
	VGAE-GAT (Deterministic)	45.9	83.3	65.6	83.3	74.6	76.7
	VGAE-GAT (Monte Carlo)	45.7	84.7	66.4	77.4	**99.0**	58.0
	Extended SEAL	**52.0**	11.5	38.0	48.0	04.0	10.0
	Classify (GAT)	34.2	52.3	36.9	5.9	26.0	24.0
	Classify (GraphSage)	32.9	31.9	35.2	27.0	55.1	52.0
Fully	VGAE-GCN (Deterministic)	34.7	91.7	61.1	73.8	77.6	71.8
	VGAE-GCN (Monte Carlo)	31.6	87.8	63.0	71.3	89.7	97.4
	VGAE-GAT (Deterministic)	28.9	84.8	54.5	68.3	84.1	80.2
	VGAE-GAT (Monte Carlo)	29.1	80.0	55.3	68.0	99.8	64.0
	Extended SEAL	29.0	23.6	30.0	14.0	05.5	11.5
	Classify (GAT)	14.1	28.4	29.7	08.7	36.2	34.5
	Classify (GraphSage)	27.6	19.9	19.7	21.0	61.8	62.9
	DEAL	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Table 5.10: Hit-Rate @ 20% results for multi-link prediction.

		AUC					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (UIS)	0.985	0.986	0.940	**0.970**	**0.926**	**0.924**
	VGAE-GCN (NIS)	0.824	0.888	0.820	0.947	0.891	0.865
	VGAE-GAT (UIS)	**0.987**	**0.984**	**0.968**	0.969	0.913	0.879
	VGAE-GAT (NIS)	0.855	0.910	0.798	0.930	0.924	0.914
Fully	VGAE-GCN (UIS)	0.973	**0.980**	0.979	**0.969**	0.925	**0.962**
	VGAE-GCN (NIS)	0.651	0.818	0.821	0.860	0.938	0.689
	VGAE-GAT (UIS)	**0.985**	**0.980**	**1.00**	0.962	**0.954**	0.955
	VGAE-GAT (NIS)	0.654	0.756	0.806	0.923	0.920	0.914

Table 5.11: AUC results for Normalized Importance Sampling (NIS) and Unnormalized Importance Sampling (UIS).

		AP					
		Cora	ACM	IMDb	CiteSeer	Photo	Computers
Semi	VGAE-GCN (UIS)	<u>0.952</u>	**0.954**	<u>0.844</u>	0.898	<u>0.812</u>	0.808
	VGAE-GCN (NIS)	0.865	0.902	0.792	**0.959**	**0.900**	**0.930**
	VGAE-GAT (UIS)	**0.955**	<u>0.951</u>	**0.899**	0.897	0.790	0.735
	VGAE-GAT (NIS)	0.706	0.699	0.788	<u>0.937</u>	**0.900**	<u>0.911</u>
Fully	VGAE-GCN (UIS)	<u>0.918</u>	**0.926**	<u>0.944</u>	<u>0.897</u>	0.818	0.890
	VGAE-GCN (NIS)	0.691	<u>0.804</u>	0.827	0.858	**0.891**	**0.928**
	VGAE-GAT (UIS)	**0.953**	**0.926**	**1.00**	0.895	0.878	0.872
	VGAE-GAT (NIS)	0.706	0.699	0.788	**0.937**	**0.900**	<u>0.911</u>

Table 5.12: AP results for Normalized Importance Sampling (NIS) and Unnormalized Importance Sampling (UIS).

Chapter 6

Conclusion and Future Work

6.1 Conclusion

A probabilistic graph query (PGQ) is a type of query that asks for the probability of a specific subgraph, given an evidence subgraph. Answering PGQs is a new and challenging task for generative graph models, as it requires modeling the joint probability distribution of the graph and the query subgraph. This is important because such models can be used in a production environment where multiple users pose a range of queries to be answered.

We showed how a trained VGAE model can be used to construct a Conditional Variational Auto-encoder (CVAE) to answer a given user's PGQ in a zero-shot manner, without retraining the model.

This thesis evaluated the proposed approach on six benchmark datasets and a range of link prediction queries. It was found that inductive link prediction from a single VGAE was more accurate than methods customized for single-link prediction on most datasets and queries, even for single-link queries. This demonstrates the effectiveness of the proposed PGQ answering system based on VGAE models and the potential for its application in real-world scenarios where multiple users pose diverse graph queries.

6.2 Future Work

An important direction for future research is to explore other generative graph models that can be used to answer PGQs. While VGAE models have shown promising results in the current study, other generative graph models such as auto-regressive models could be explored. Auto-regressive models have the advantage of being able to model dynamic changes in the graph over time. These models can be trained to model the probability distribution of the graph at each time step, conditioned on the graph at the previous time step.

Another valuable direction for future research is to improve the way the model distinguishes links that are specified as absent in the query from unspecified links. In the current study, the model sets absent links to non-existing, which can result in lower accuracy for PGQs that involve absent links. A more promising approach is to distinguish between absent links and unspecified links, by representing the unspecified links as missing values in the adjacency matrix.

Another promising direction for future research is to utilize the CVAE model for other prediction targets, such as node labels and features, and edge types and features. It would even be possible to generate a complete graph, which defines the graph completion task analogous to image completion. The CVAE model can be used to generate new graph structures that complete missing or unknown components of a graph, especially in scenarios where some or all of the attributes of the test nodes are missing or unknown. This has practical applications in areas such as recommendation systems and drug discovery.

Bibliography

[1] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[2] Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.

[3] George Fishman. *Monte Carlo: concepts, algorithms, and applications*. Springer Science & Business Media, 2013.

[4] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1433–1445, New York, NY, USA, 2018. Association for Computing Machinery.

[5] Mikhail Galkin, Zhaocheng Zhu, Hongyu Ren, and Jian Tang. Inductive logical query answering in knowledge graphs. In *Advances in Neural Information Processing Systems*, 2022.

[6] William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[7] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[8] Yu Hao, Xin Cao, Yixiang Fang, Xike Xie, and Sibo Wang. Inductive link prediction for nodes having only attribute information. *Structure*, 3(4):5, 2020.

[9] Bert Huang, Angelika Kimmig, Lise Getoor, and Jennifer Golbeck. Probabilistic soft logic for trust analysis in social networks. In *International Workshop on Statistical Relational Artificial Intelligence (StaRAI 2012)*, 2012.

[10] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[11] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[12] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In *NeurIPS*, 2019.

[13] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

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

[15] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf. 01 2007.

[16] Ryan A Rossi, Rong Zhou, and Nesreen K Ahmed. Deep inductive graph representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):438–452, 2018.

[17] Stuart Russell. Unifying logic and probability. *Communications of the ACM*, 58(7):88–97, 2015.

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

[19] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 412–422, Cham, 2018. Springer International Publishing.

[20] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28:3483–3491, 2015.

[21] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR, 2020.

[22] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster.

[23] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023.

[24] 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*, volume 32, 2018.

[25] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. 09 2022.

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

[27] Seongjun Yun, Minbyul Jeong, Sungdong Yoo, Seunghun Lee, S Yi Sean, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks: Learning meta-path graphs to improve gnns. *Neural Networks*, 2022.

[28] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[29] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[30] Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. In *International Conference on Learning Representations*, 2020.