Rule-Enhanced Graph Learning

Ali Khazraee [1] **Abdolreza Mirzaei** [1 2] **Majid Farhadi** [2 3] **Parmis Naddaf** [2] **Kiarash Zahirnia** [2]
Mohammad Salameh [4] **Kevin Cannons** [4] **Richard Mar** [2] **Mingyi Wu** [2] **Oliver Schulte** [2]

Abstract

Knowledge-enhanced graph learning is one of the current frontiers for neural models of graph data. In this paper, we propose a new approach to enhancing deep generative models with domain knowledge that is represented by first-order logic rules. First-order logic provides an expressive formalism for representing interpretable knowledge about relational structures. Our approach builds on ideas from statistical-relational learning (SRL), a field of machine learning that aims to combine first-order logic with statistical models. One of the fundamental concepts in SRL is *rule moment matching*: constrain model training such that the expected instance count of each rule matches its observed instance count. Our conceptual contribution is to adapt this idea for deep generative models by maximizing the (approximate) model likelihood subject to the rule moment matching constraint. Our algorithmic contribution is a novel method for computing the expected rule instance count of a Variational Graph Autoencoder (VGAE), based on matrix multiplication. Empirical evaluation on four benchmark datasets shows that rule moment matching improves the quality of generated graphs substantially (by orders of magnitude on standard graph quality metrics).

[1] Electrical and Computer Engineering Department, Isfahan University of Technology, Iran. [2]School of Computer Science, Simon Fraser University, Canada. [3]Department of Electrical and Computer Engineering, University of Alberta, Canada. [4]Huawei Technologies Ltd., Canada.. Correspondence to: Ali Khazraee <a.khazraee@ec.iut.ac.ir>, Abdolreza Mirzaei <mirzaei@sfu.ca>, Oliver Schulte <oschulte@cs.sfu.ca>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria.

1. Introduction

Generative models for graphs based on graph neural networks (GNNs) have achieved great success in modeling complex graphs (Hamilton, 2020). One of the current research frontiers is enhancing graph learning with domain knowledge (Tian et al., 2024). Different enhancement methodologies are appropriate for different types of knowledge (e.g., knowledge from models, humans, external sources). In this paper, we consider leveraging knowledge in the form of a *first-order logic knowledge base* (Russell & Norvig, 2010), comprising a set of if-then rules. An example rule would be "If person X works in city Y, then X lives in city Y (with probability p)" (see Appendix Figure 5(a)).

Advantages. Logical rules have several advantages for enhancing graph learning. (1) Expressiveness: First-order rules are one of the most common formalisms for representing domain knowledge in AI and database systems (Russell & Norvig, 2010). (2) Interpretability: If-then rules are easily understood by users and domain experts. In this work we derive rules from a *causal graph*, which facilitates an intuitive qualitative interpretation in terms of causal relationships, and allows the set of rules to be visualized perspicuously. (3) Learnability: The field of statistical-relational learning has developed statistical methods for learning rules from a heterogeneous training graph, known as *structure learning*. (4) User Control: Users can control the behavior of the final graph generation system by specifying and/or rejecting rules. We utilize a *mixed-initiative* structure learning method, where users can partially specify a causal graph, and the system completes it based on data. (5) Graph Realism and Data Efficiency: When the first-order rules capture valid patterns, constraining graph learning to match them leads to more realistic graphs. Even when GNNs have the expressive power to capture these patterns through embeddings, presenting them to the GNN explicitly as constraints speeds up learning because less data is needed to learn them.

Approach. Figure 1 shows our system components. We build on fundamental ideas from statistical-relational learning (SRL) and show how they can be combined with deep graph generative models (GGMs). SRL is an area of machine learning that combines first-order logic with statistical

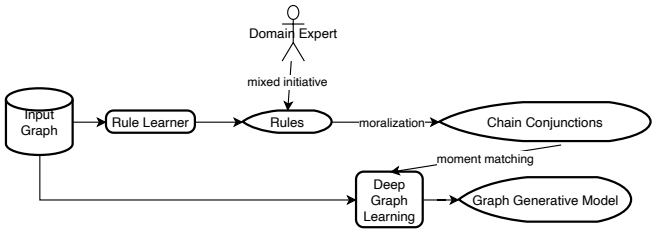


Figure 1. System Overview for Rule-Enhanced Graph Learning

models (Raedt et al., 2016). A fundamental concept of SRL is *rule moment matching* (Domingos & Lowd, 2019; Russell, 2015; Kuzelka et al., 2018). The general idea is that a rule can be viewed as specifying a *motif* or subgraph pattern with an **instance count** in a given graph (for illustration see Appendix Figure 5). Rule moment matching requires that *the expected rule instance count from a model should match the observed rule count in a training graph.* Our novel GGM training objective is to *maximize the GGM likelihood subject to rule moment matching*.

Our algorithmic contribution is a *differentiable new matrix multiplication method* for computing observed and expected rule instance counts. We show that for every rule (satisfying a minor syntactic constraint), there is a corresponding sequence of adjacency matrices, such that i) the observed rule count is obtained by multiplying data adjacency matrices, and ii) the expected rule count is obtained by multiplying expected adjacency matrices. A well-known special case is calculating the number of triangles in a graph through the third power $\mathbf{A}^3$ of its adjacency matrix.

Evaluation. Our methodology uses an A-B design where we compare training a variational graph auto-encoder (VGAE) (Kipf & Welling, 2016) with and without rule moment matching, on five benchmark datasets. To learn rules, we deploy the Factorbase system, which uses Bayesian network structure learning to find a comprehensive set of probabilistic rules (Schulte & Khosravi, 2012; Qian & Schulte, 2015). We find that rule-enhanced VGAEs score better than standard VGAEs on several metrics: (1) They generate *more realistic graphs*, by orders of magnitude, as measured by SOTA graph quality metrics (F1 MMD) (Thompson et al., 2022; O'Bray et al., 2022). (2) On the downstream task of *node classification*, the rule-enhanced VGAE node embeddings improve accuracy compared to standard VGAE, whereas the link prediction accuracy remains the same.

Contributions Our main contributions can be summarized as follows.

- A new objective function for enhancing generative graph learning with domain knowledge represented by logical rules.
- A new matrix multiplication algorithm for counting the number of rule instances in a graph.
- A new algorithm for estimating the expected number of rule instances for a Variational Graph Autoencoder model, based on matrix multiplication.
- Two extensions of the previous VGAE model:
  - VGAE+ generates node features and edge types.
  - VGAE+R uses the new objective function to train a VGAE+ model that matches rule instance counts.

2. Related Work

Our work falls under the heading of *neuro-symbolic AI*, a cutting-edge field of AI that aims to combine symbolic formalisms, such as first-order logic, with neural network learning. For surveys of neuro-symbolic AI, please see (Raedt et al., 2020; Garcez & Lamb, 2023). Within neuro-symbolic AI, our approach belongs to the family of *semantic loss* frameworks where the training objective is enhanced with symbolic knowledge, but the trained system is a standard NN model (in our case, a deep GGM) that does not utilize rules at test time. In contrast, rule-based approaches utilize rules at test time, for example to perform symbolic reasoning (Raedt et al., 2020; Qu et al., 2021). Compared to other semantic loss approaches, our main novelty is that we incorporate knowledge expressed in first-order logic, rather than the less powerful formalism of propositional logic (Xu et al., 2018; Garcez & Lamb, 2023).

Deep graph generative models. The closest predecessor to our work is the constrained VGAE model of (Ma et al., 2018) where a VGAE likelihood is maximized subject to a constraint of the form $g(\boldsymbol{\theta}) = 0$. While this general form covers rule matching, the work of Ma et al. does not incorporate first-order logic for specifying graph patterns, nor does it address computing pattern counts.

In principle the rule-enhanced likelihood objective can be used for maximum likelihood training with any deep graph generative model. We selected VGAEs as our base model for several reasons. (1) They are a well-established and widely used GGM. (2) They support learning from a single large graph, rather than from a set of graphs (Faez et al., 2021). Rule learners also utilize the single-graph setting (Qian & Schulte, 2015; Meilicke et al., 2024), so the VGAE input data are compatible with the rule learner input data. (3) As we show in this paper, the conditional link independence assumptions of VGAEs facilitates the computation of expected rule instance counts. We believe that extending rule moment matching to other generative models is a fruitful topic for future research.

Maximum Entropy Moment Matching. Kuzelka et al. (2018) show that a distribution P over graphs maximizes entropy subject to rule moment matching if and only if P is defined by a log-linear model known as a **Markov Logic Network** (MLN) with maximum likelihood weights $\boldsymbol{w}$. While the maximum entropy objective is based on rule counts only, our constrained likelihood objective can also capture local graph patterns that complement the global graph statistics represented by rule counts. For example, matching the number of observed triangles in a graph is unlikely to capture community structure, or which nodes have special properties such as centrality.

3. Variational Graph Auto-Encoder Model and Training

We describe our generative VGAE+ model, which augments a VGAE to generate features and labels, with training and implementation details. Figure 2 shows the VGAE+ training architecture.

3.1. Data Format

A graph is a pair of (V, E) where V is a set of nodes of size $|V| = n$ and $E \subseteq V \times V$ is a set of edges. Node features are summarized in $n \times f$ matrix $\mathbf{X}$ and node labels in a $n \times L$ matrix $\mathbf{L}$ where the u-th row of $\mathbf{L}$ is a one-hot encoding of the label of node u. Different edge types are represented by a set of adjacency matrices $\mathbf{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_T\}$. The notation $\mathbf{A}_r[u, v] = 1$ indicates that there is a link $u \rightarrow_r v$ of type r from node u to node v.

3.2. VGAE+ Model

Let $\boldsymbol{z}$ be an $n \times d$ matrix that represents latent node embeddings. In the VGAE model, links are generated independently given node embeddings. Following the GraphVAE approach (Simonovsky & Komodakis, 2018), we generate node classes and node features independently as well given node embeddings. We thus utilize three decoder models (see Figure 2):

$$
\begin{aligned}
p_\theta(\mathbf{A}|\boldsymbol{z}) &= \prod_{r=1}^{T} \prod_{u,v} p_{\theta_r}(\mathbf{A}_r[u, v]|\boldsymbol{z}[u], \boldsymbol{z}[v]) \\
p_\psi(\boldsymbol{X}|\boldsymbol{z}) &= \prod_{u} p_\psi(\boldsymbol{X}[u]|\boldsymbol{z}[u]) \\
p_\phi(\boldsymbol{L}|\boldsymbol{z}) &= \prod_{u} p_\phi(\boldsymbol{L}[u]|\boldsymbol{z}[u])
\end{aligned}
\tag{1}
$$

where $p_\theta : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1]$ is a trainable **link decoder**, p_ψ is a trainable **feature decoder**, and p_ϕ is a trainable **label decoder**.

Let $D = (\boldsymbol{X}, \mathbf{A})$ be the attributed training graph. The graph **encoder** $q_\phi(\boldsymbol{z}|\boldsymbol{X}, \mathbf{A})$ is implemented by a GNN that takes as input an attribute graph and returns latent node embeddings. For compatibility with baseline methods, the encoder does not receive node labels as input, but adding them is straightforward.

A VGAE+R model is trained using the **rule-matching variational ELBO objective** that extends the standard VGAE ELBO (Kipf & Welling, 2016; Hamilton, 2020):

$$
\begin{aligned}
\mathcal{L}(\theta, \psi, \phi) = KL\big(q_\phi(\boldsymbol{z}|\boldsymbol{X}, \mathbf{A})||p(\boldsymbol{z})\big) - E_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{X}, \mathbf{A})} \\
[\alpha \times \ln p_\theta(\mathbf{A}|\boldsymbol{z}) + \beta \times \ln p_\psi(\boldsymbol{X}|\boldsymbol{z}) + \gamma \times \ln p_\phi(\boldsymbol{L}|\boldsymbol{z}) \\
+\lambda/k \sum_{i=1}^{k} \rho(n_i(D), E[n_i(\mathcal{G})|\boldsymbol{z}])]
\end{aligned}
\tag{2}
$$

where $\rho(count_1, count_2) \geq 0$ is a differentiable distance metric that maps two counts to a non-negative number, such that $\rho(count_1, count_2) = 0$ if and only if $count_1 = count_2$. The notation $E[n_i(\mathcal{G})|\boldsymbol{z}]) = E_{\mathcal{G} \sim P(\mathcal{G}|\boldsymbol{z})}[n_i(\mathcal{G})]$ denotes the expected rule instance count, conditional on the node embeddings $\boldsymbol{z}$. The hyperparameters α, β and γ weight the importance of different reconstruction tasks. The hyperparameter λ controls the importance of moment matching and can be interpreted as a Lagrange multiplier. The next section discusses how the instance count $n_i(\mathcal{G})$ is defined.

Implementation of VGAE+ model For the encoder, we used an RGCN (Schlichtkrull et al., 2018), with the Pytorch Geometric implementation. The node embedding dimension was 64, the number of RGCN layers 2, with separate layers for estimating the Gaussian posterior mean and the Gaussian posterior standard deviation for each node. We trained the RGCN for 300 epochs. In our experiments, we set $\lambda = 0$ as the VGAE+ baseline training objective, and $\lambda = 1$ as the training objective for a VGAE+R model that leverages rules.

4. First-Order Logic Rules and Instance Counts

A rule is of the form $\boldsymbol{B} \rightarrow H$ where $\boldsymbol{B}$ is a conjunction of literals known as the rule *body* and H is a single literal known as the head. Most rule learners are based on *discriminative* learning, building on classification techniques to search for bodies that predict the head. A question researched in statistical-relational learning is how to convert a set of predictive rules to a set of graph features or graph statistics that support *generative* graph modelling. For log-linear MLNs, the recommended answer is to convert each

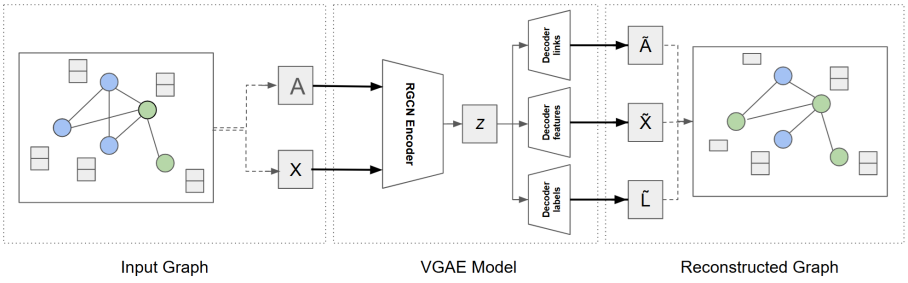


Figure 2. Encoder-Decoder Training Architecture for the VGAE+ Model

rule to a conjunction $\phi = (\boldsymbol{B}, H)$ (Domingos & Richardson, 2007, 12.5.3), (Kazemi et al., 2014; Khosravi et al., 2012). This is known as the **moralization** procedure because it is a logical analogue of the moralization algorithm for converting a directed graphical model to an equivalent undirected graphical model (e.g., converting a Bayesian network to a Markov random field). While a moment matching objective can in principle be applied with any method for extracting graph statistics from a set of rules, we follow the moralization approach and utilize first-order conjunctions. We briefly review concepts from first-order logic; the Appendix provides full formal definitions.

4.1. First-Order Conjunctions

A **functor** represents a node attribute/label or relationship of a given type. We assume that all functors are Boolean. A unary functor f represents a node attribute/label. The notation $f(u)$ indicates that node u has the attribute represented by f. A binary functor R_r represents a relation of type r. The notation $R_r(u, v)$ indicates the existence of a link of type r between nodes u and v.

A **node variable** ranges over node indices. Note that a node variable is not a random variable, but is a placeholder for a generic node index, analogous to a variable in a programming language. For example, if $node$ is a node variable, the assignment $node = u$ assigns node index u to the node variable. A **literal** is of the form $R(U, V), \neg R(U, V), f(U), \neg f(U)$. A **grounding** $\boldsymbol{U} = \boldsymbol{u}$ assigns a node index from a list $\boldsymbol{u}$ to each node variable in a list $\boldsymbol{U}$. A graph $\mathcal{G}$ specifies a Boolean variable for each ground literal. We write $I_{\mathcal{G}}(\ell_i(\boldsymbol{U} = \boldsymbol{u}))$ for the **indicator function** that returns 1 if a ground literal is true in a graph, 0 otherwise.

A **conjunction** is a list of literals $\phi = \ell_1, \ldots, \ell_s$. A graph $\mathcal{G}$ **satisfies** a ground conjunction $\phi(\boldsymbol{U} = \boldsymbol{u})$ if it satisfies each ground literal in the conjunction, where $\boldsymbol{U}$ includes all node variables in the conjunction. A grounding is **valid** if (i) a node variable is assigned the same node in each occurrence, and (ii) two distinct node variables $U \neq V$ are assigned two distinct nodes. The **instance count** for a conjunction ϕ in a graph $\mathcal{G}$ returns the number of valid groundings that satisfy the conjunction:

$$n_\phi(\mathcal{G}) = \sum_{\boldsymbol{U}=\boldsymbol{u} \text{ is valid}} \prod_{i=1}^{s} I_{\mathcal{G}}(\ell_i(\boldsymbol{U} = \boldsymbol{u})).$$

Example. For the graph $\mathcal{G}$ shown in Figure 3, $Taughtby(Course, Professor)$ is a relationship literal. A valid grounding for this literal is $Taughtby(Course =$ "$Deep\ Learning$", $Professor =$ "$Jane$"$)$. Because there is a TaughtBy link between deep learning and Dr. Jane in the graph $\mathcal{G}$, we have $I_{\mathcal{G}}(Taughtby(Course =$ "$Deep\ Learning$", $Professor =$ "$Jane$"$)) = 1$. On the other hand, there is no $TaughtBy$ link between Deep Learning and Tom in the graph so $I_{\mathcal{G}}(\neg Taughtby(Course =$ "$Deep\ Learning$", $Professor =$ "Tom"$)) = 1$. Similarly $I_{\mathcal{G}}(Intelligent(Student =$ "$Jack$"$)) = 1$ because Jack is intelligent.

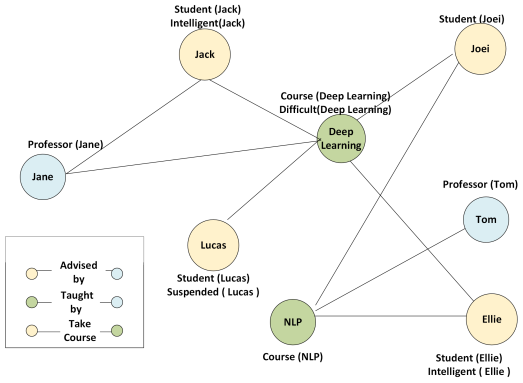


Figure 3. An example graph containing unary and binary literals.

4.2. Expected Instance Counts

For given node embeddings $\boldsymbol{z}$, a VGAE+ model assigns a probability $p_{\boldsymbol{z}}(\ell(\boldsymbol{U} = \boldsymbol{u}))$ to each ground literal. Since literals are independent given $\boldsymbol{z}$, the probability of a con-

junction is just the product of the conjunct probabilities. We can therefore define the **probabilistic instance count** by replacing the 0/1 indicator values with probabilities:

$$n_\phi(\tilde{\mathcal{G}}_{\boldsymbol{z}}) = \sum_{\boldsymbol{U}=\boldsymbol{u} \text{ is valid}} \prod_{i=1}^{s} p_{\boldsymbol{z}}(\ell_i(\boldsymbol{U}=\boldsymbol{u}))$$

where $\tilde{\mathcal{G}}_{\boldsymbol{z}}$ is the **expected graph**, which is a probabilistic graph where each link and attribute is assigned its decoder probability $p_{\boldsymbol{z}}$; see Appendix Figure 6 for illustration.

Proposition 4.1. *The expected conjunction count given a set of node embeddings can be computed as the conjunction count in the expected graph:* $E[n_\phi(\mathcal{G})|\boldsymbol{z}] = n_\phi(\tilde{\mathcal{G}}_{\boldsymbol{z}})$

The next section presents a matrix multiplication algorithm for obtaining instance counts in both the observed and the expected graphs.

5. Matrix Multiplication for Instance Counting

We describe our method briefly and informally; a formal specification is in Appendix A.4. A conjunction with P relationship literals $\ell_1(U_1, V_1), \ldots, \ell_P(U_P, V_P)$ is a **chain** if $U_{i+1} = V_i$ for every i. A conjunction is a chain conjunction if its binary literals form a chain. To aid intuition, note that a chain conjunction defines a *graph motif*: The relationship chain defines a path template (or metapath (Sun & Han, 2012)) that specifies which types of links are present (absent) in the path. The unary functors specify constraints on what type of nodes participate in the motif.

Example. $\phi = \{TakesCourse(Student = \text{“}Jack\text{”}, Course = \text{“}Deep\ Learning\text{”}), Taughtby(Course = \text{“}Deep\ Learning\text{”}, Professor = \text{“}Jane\text{”})\}$ is an example of a grounded chain conjunction. For the graph shown in Figure 3, $I_{\mathcal{G}}\{\phi\} = 1$ because student “Jack” takes the course “Deep Learning” taught by professor “Jane”.

The input to our counting algorithm is an input graph $\mathcal{G}$ and a centered chain conjunction $\{\ell_1(U_1, V_1), \ldots, \ell_P(U_P, V_P), \ell_1(W_1), \ldots, \ell_Q(W_Q)$.

A chain is **centered** if the only node variable that appears twice non-consecutively is the first one (formally, if $U_i = V_j$ and $j \neq i+1$, then $U_i = U_1$). In our experiments, we found that all learned rules were centered. We discuss below how our method can be extended to non-centered chains. We construct a sequence of matrices as follows.

1. For a positive relationship literal $\ell = R(U, V)$ the *initial adjacency matrix* is $\mathbf{A}_\ell = \mathbf{A}_r$, the adjacency matrix for relation R. For a negative relationship literal $\ell = \neg R(U, V)$ the initial adjacency matrix is $\mathbf{A}_\ell = \neg\mathbf{A}_r$ where $\neg\mathbf{A}_r[u, v] = 1 - \mathbf{A}_r[u, v]$ for all node indices u, v.

2. Say that a unary functor $\ell(W)$ matches a relationship literal $\ell(U, V)$ if $U = W$ or $V = W$. For every relationship literal $\ell(U, V)$ and for every matching grounding $W = w = U$, multiply the row $\mathbf{A}_\ell[w, :]$ by the indicator $I_{\mathcal{G}}(W = w)$ (probability for expected counts).

   Similarly for the columns, if $W = w = V$, multiply the column $\mathbf{A}_\ell[w, :]$ by the indicator $I_{\mathcal{G}}(W = w)$ If a node w violates a unary functor constraint, its row resp. column entries will be zeroed out, so we refer to the resulting matrices as the *masked* matrices $\bar{\mathbf{A}}_\ell$.

3. Define the sequence of matrix multiplications $O_k, k = 1, \ldots, P$ as follows.
   (a) O_1 is the masked adjacency matrix for literal L_1.
   (b) Inductively, $O_{k+1} = O_k \bar{\mathbf{A}}_{\ell_{k+1}}$. If $V_{k+1} = U_1$, we zero out the non-diagonal entries.

The next proposition shows that the instance count for the relationship chain can be obtained through summing over the entries in the constructed matrix product.

Proposition 5.1. *The (u, v)-th entry of O_k counts the number of groundings of a centered chain conjunction ϕ of length k in a graph $\mathcal{G}$ where $U_1 = u$ and $V_P = v$. Therefore*

$$n_\phi(\mathcal{G}) = \sum(O_k(\phi)).$$

Example The element (i, j) in A^3 represents the number of distinct paths of exactly 3 steps connecting node i to node j. Our algorithm uses different adjacency matrices to capture multiple relationship types. For example, multiplying three different adjacency matrices A, B, and C, e.g., ABC, gives the number of length-3 paths where the first step is from A, the second from B, and the third from C. This generalization is more flexible than just computing A^3, as it allows analyzing complex networks with diverse connections between nodes. For example, if A, B, and C represent friendships, co-workers, and family, respectively, then ABC counts paths where the first step is a friendship, the second a co-worker relationship, and the third a family connection. This provides a richer understanding of node connectivity and reachability in the network.

Our construction and the correctness proof extend to probabilistic adjacency matrices and unary factors, which allows us to compute expected motif counts (see Proposition 4.1). Another extension is to the case of nested conjunctions, which have no crossing equalities. Say that two variable equalities $U_{k_1} = V_{k_2}$ and $U_{k_3} = V_{k_4}$ cross if $k_1 < k_3 < k_2$ and $k_2 < k_4$. A nested chain comprises centered chains, so construction recursively computes instance counts.

6. Evaluation

We detail our methodology and then discuss our empirical results. We will make our code available on-line.

6.1. Experimental Design.

We describe our benchmark datasets, comparison methods, and how evaluation metrics are computed.

6.1.1. Datasets

We utilize datasets from previous studies of GGMs (Kipf & Welling, 2016; Yun et al., 2019; Hao et al., 2020). Cora, ACM, and CiteSeer are citation networks, IMDb is a movie dataset. Appendix A.5 presents dataset statistics.

Data Preprocessing Following previous work (Kipf & Welling, 2016), for GNN message passing we add self-loops and make all links undirected (i.e., if the training data contains an adjacency, $v \rightarrow u$, it also contains $u \rightarrow v$.) Cora and CiteSeer are homogeneous datasets, whereas ACM and IMDb are heterogeneous datasets. Rule learning is applied to the original data.

6.1.2. Evaluation Metrics

We compare rule-enhanced VGAE+ training with standard VGAE+ training, using three main metrics. In the following, we refer to a complete dataset as the **input graph**. Our evaluation metrics measure performance on two downstream tasks, node classification and link prediction, and graph realism, the quality of generated graphs.

Count Distance We report the root mean squared distance between the observed motif counts and the expected motif counts in the reconstructed graph. Formally, let $\tilde{D}$ be a reconstruction of the input graph D obtained by sampling a node embedding matrix $\boldsymbol{z}$ from the encoder posterior $q_\phi(\boldsymbol{z}|D)$ and and then applying the decoder model Equation (1) to $\boldsymbol{z}$. We report the mean squared distance $(1/k \sum_{i=1}^{k} [n_i(D) - n_i(\tilde{D})]^2)^{1/2}$ as the **count distance** (CD).

Graph Realism measures how similar graphs generated by the model are to observed graphs. How to quantitatively assess generated graphs has been studied in recent papers (O'Bray et al., 2022; Thompson et al., 2022; Shirzad et al., 2022). The general approach proceeds in two stages: 1) For a graph $\mathcal{G}$, extract a real-valued **descriptor** vector $\phi(\mathcal{G})$. 2) Measure the similarity $\mu(\mathcal{G}, \hat{\mathcal{G}})$ of an observed graph $\mathcal{G}$ and a generated graph $\hat{\mathcal{G}}$ by applying a distance/kernel on real-valued vectors to their descriptors $\phi(\mathcal{G})$ and $\phi(\hat{\mathcal{G}})$. The similarity of a set of observed graphs and a set of generated graphs can be quantified as the similarity of their descriptor sets using Maximum Mean Distance (MMD). The SOTA descriptor function utilizes a *reference embedding network* GNN $\mathcal{E}$. The embedder $\mathcal{E}$ is obtained from random weights or pretraining and is therefore independent of any of the models under evaluation.

We adapt the GNN-based approach to the setting of learning from a single training graph D as follows. We compare the training graph to generated expected graphs $\tilde{\mathcal{G}}_1, \ldots \tilde{\mathcal{G}}_m$. An expected graph $\tilde{\mathcal{G}}_i$ is generated by sampling a node embedding $\boldsymbol{z}_i$ from the prior distribution $p(\boldsymbol{z})$, then applying the decoder model Equation (1) to $\boldsymbol{z}_i$. We apply an embedder $\mathcal{E}$ with random weights to the training graph resp. generated expected graphs to obtain embeddings e resp. $\hat{e}_1, \ldots, \hat{e}_m$. The random GNN option does not require multiple training graphs. The message-passing mechanism of GNNs naturally extends to weighted graphs, so we can apply the GNN embedder to expected graphs directly, rather than sampling binary adjacency matrices/features from the expected graph. To quantify the similarity of the generated embeddings to the training graph embedding e, we utilize the MMD metric with a linear kernel, which is recommended by (Thompson et al., 2022) for measuring the realism of generated graphs. Unlike Count Distance, the MMD metric is completely independent of the training objective.

Link Prediction To compute a link prediction score, following (Kipf & Welling, 2016) we randomly divide the links in the input graph into training links and test links (80%/20%). The training graph includes the training links and all nodes from the input graph. At test time, we use the node embeddings from the training graph to predict the test edges (Kipf & Welling, 2016).

Node Classification To compute a node classification score, we randomly divide the nodes in the input graph into training and test nodes (80%/20%). The training graph is the input graph but with the test node labels removed. At test time, we run the encoder on the input graph to obtain node embeddings for all nodes, then apply the decoder to predict node labels for the test nodes.

6.1.3. Comparison Methods.

Rule Learning We used the Factorbase system (Qian & Schulte, 2015; Schulte & Khosravi, 2012) with default settings. Factorbase is the most scalable system for learning first-order generative graphical models with state-of-the-art predictive performance. Factorbase outputs directed probabilistic rules $\boldsymbol{B} \rightarrow H; p$, where H is a literal, $\boldsymbol{B}$ is a conjunction, and p, is a conditional probability $P(H|\boldsymbol{B})$.

As a *rule quality metric* for pruning, we use $2n(\boldsymbol{B}, H) \ln(n(H|\boldsymbol{B})/n(H)) - \ln n(H)$. Here the counts n are computed with respect to the training data D. The

Table 1. Count Distance with and without rule-enhancement.

DATASET	CORA	CITESEER	IMDB	ACM
VGAE+	4233.6	4085.6	137612.3	36572.1
VGAE+R	**797.2**	**523.56**	**14373.3**	**12984.5**

Table 2. Graph Realism MMD F1 metric.

DATASET	CORA	CITESEER	IMDB	ACM
VGAE+	2.8E+12	6.8E+11	1.7E+19	**1.9E+15**
VGAE+R	**1.2E+10**	**2.6E+9**	**7.8E+15**	4.5E+15

conditional count is given by $n(H|\boldsymbol{B}) = n(H, \boldsymbol{B})/n(\boldsymbol{B})$. This metric computes the increase in the log-probability of the head given the body, relative to the prior probability of the head, together with a BIC-type correction for sample size (Schulte & Gholami, 2017). We keep all rules whose quality metric is above 0, which can be interpreted as a positive local BIC model selection score. Rule pruning reduces the number of rules for scalability, and also simulates the impact of a domain expert selecting the most important rules.

VGAE+R model For the distance metric ρ that compares an expected rule count with an observed row count we use

$$\rho(n_i(D), E[n_i(\mathcal{G})|\boldsymbol{z}]) = |\ln n_i(D) - \ln E[n_i(\mathcal{G})|\boldsymbol{z}])|$$

The magnitude of conjunction counts grows exponentially with the number of node variables in the conjunction. Comparing expected counts on a log-scale decreases the impact of the number of variables and improves numeric stability.

6.2. Experimental Results

We first evaluate the quality of the generated graphs, then the VGAE effectiveness on downstream tasks.

6.2.1. COUNT DISTANCE AND GRAPH REALISM

Table 1 shows the difference between expected and observed rule instance counts.

Both methods show large absolute distances because a VGAE model does not match graph data well; in particular, it produces overly dense graphs (Orbanz & Roy, 2014). However *in comparison*, we observe a very large improvement in the match between expected and observed counts, by at least a factor of 5 depending on the dataset. This shows that VGAE training without the moment matching constraint is far from matching rule counts.

On the graph realism metric shown in Table 2, we again find large absolute distances with the training set, and very large improvements through rule learning. An exception is the ACM dataset where the standard VGAE generates more realistic graphs, though not by an order of magnitude. Overall we conclude that unconstrained VGAE training does not match the instantiation counts of the learned rules, and that moment matching has a large impact on the graphs generated.

6.2.2. DOWNSTREAM TASKS

On the downstream task of node classification, Table 3 shows a substantive improvement on 2 out of 4 datasets, Cora and Citeseer. For example on Cora, the F1 score improves by 13 points, and on Citeseer by 3 points. On IMDB there is no big difference, and on ACM, the node classification score decreases. This is likely related to the fact that ACM is the only data set for which rule matching decreases graph realism (cf. Table 2).

Table 3. Node classification results for various datasets.

Dataset	Metric	Model	
		VGAE+	VGAE+R
Cora	Precision	0.7952	**0.8556**
	F1	0.7178	**0.8440**
	Recall	0.8284	**0.8542**
Citeseer	Precision	0.7265	**0.7579**
	F1	0.6958	**0.7251**
	Recall	0.7233	**0.7639**
IMDB	Precision	0.6328	**0.6337**
	F1	**0.6282**	0.6219
	Recall	**0.6297**	0.6250
ACM	Precision	**0.9675**	0.9323
	F1	**0.8623**	0.7925
	Recall	**0.9624**	0.9396

For the downstream task of link prediction, we use AUC (Area Under Curve) and AP (Average Precision), standard metrics for class-imbalanced binary labels. AUC measures the area underneath the receiver operating characteristic (ROC) curve and AP calculates the average of the precisions over all possible classification thresholds (Davis & Goadrich, 2006). We observe no substantial difference between the vanilla and the rule-enhanced VGAE models. Inspecting the learned rules, we find that many of them express homophily constraints on links, e.g. that if one paper cites another, then they are likely from the same research area. We believe that the standard message-passing mechanism adequately captures homophily constraints through node embeddings even without an explicit moment matching constraint.

Table 4. Link prediction results for various datasets and edge types.

Dataset	Model type	ACM		Cora	Citeseer	IMDB	
		Author - Paper	Paper - Subject	Paper - Paper	Paper - Paper	Actor - Movie	Director - Movie
AUC	VGAE+	0.9748	0.9741	0.9209	**0.9067**	0.9032	0.8869
	VGAE+R	**0.9769**	0.9741	**0.9328**	0.8810	**0.9074**	**0.8908**
AP	VGAE+	0.9945	0.9961	0.9064	0.9064	0.9201	0.8949
	VGAE+R	**0.9963**	**0.9973**	**0.9355**	**0.9156**	**0.9255**	**0.8950**

Learning Curve We report a learning curve experiment to examine the effect of rule knowledge on data efficiency. After learning an informative set of rules S on the entire training graph, we sample $x\%$ of nodes and use them to train the standard VGAE+ model and the VGAE+ model enhanced by the rules S. The models are tested on the remaining $100 - x\%$ of nodes. The idea is to simulate the impact of a domain expert providing the model with a strong set of rules. As Figure 4 illustrates, on the CiteSeer dataset the rules make learning substantially more efficient. We omit learning curves for other datasets due to space constraints.

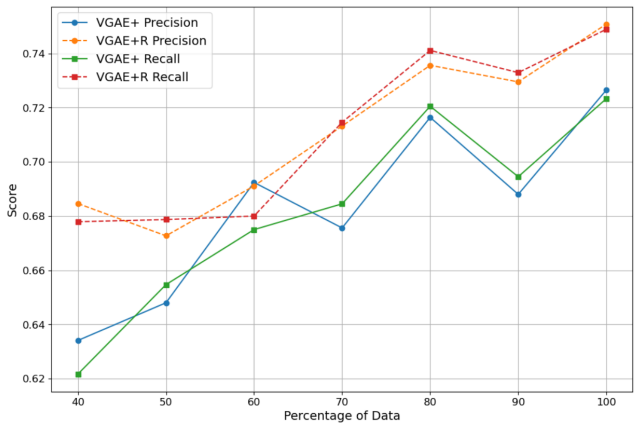


Figure 4. A node classification learning curve for the Citeseer Dataset. The rule-enhanced VGAE model achieves both higher precision and higher recall for almost every dataset size.

7. Conclusion

We proposed a new objective function for training a deep graph generative model (DGGM) to incorporate domain knowledge expressed by logical rules: Maximize the data likelihood subject to a moment matching constraint, which requires the expected rule instance counts under a model to match the observed rule instance count. Our main algorithmic contribution is a new differentiable matrix multiplication method for computing both observed and expected counts. For the observed counts, matrix multiplication is applied to the data adjacency matrices directly. For the expected counts, matrix multiplication is applied to the expected adjacency matrices under the model. In empirical evaluation, we found that rule matching improves the quality of the graphs generated by a Variational Graph Auto-Encoder (VGAE) model by orders of magnitude, both with respect to expected counts and with respect to a standard metric of graph realism. On downstream tasks that apply the trained GGM, rule matching substantially improves node classification accuracy, and makes little difference to link prediction performance.

The main limitation of our current approach is scaling the matrix multiplication algorithm for expected counts. While multiplying sparse 0/1 adjacency matrices is fast, expected adjacency matrices are dense and slow to multiply. A possible approach would be to deploy approximation algorithms from the related problem of weighted model counting (van Bremen & Kuzelka, 2020).

Our novel approach to rule-enhanced graph learning opens several avenues for future research, including (1) Extracting different rule statistics for moment matching (e.g. conditional probabilities from if-then rules). (2) End-to-end joint learning of a rule set and a GGM through moment matching. (3) Enhancing different types of GGMs with rule matching (e.g. auto-regressive methods).

In sum, rule moment matching presents a novel approach to neuro-symbolic AI that combines logical rules with deep graph learning. Our experiments show great potential for enhancing deep graph generative models with rule-based knowledge.

Acknowledgments

This research was supported by grants to the senior author from the SFU-Huawei Joint Lab and the Natural Sciences and Engineering Research Council of Canada.

References

Davis, J. and Goadrich, M. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 233–240, New York, NY, USA, 2006. Association for Computing Machinery.

Domingos, P. and Lowd, D. Unifying logical and statistical ai with markov logic. *Communications of the ACM*, 62 (7):74–83, 2019.

Domingos, P. and Richardson, M. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning*. MIT Press, 2007.

Faez, F., Ommi, Y., Baghshah, M. S., and Rabiee, H. R. Deep graph generators: A survey. *IEEE Access*, 9: 106675–106702, 2021.

Garcez, A. d. and Lamb, L. C. Neurosymbolic ai: The 3 rd wave. *Artificial Intelligence Review*, pp. 1–20, 2023.

Hamilton, W. L. *Graph representation learning*, volume 14. Morgan & Claypool Publishers, 2020.

Hao, Y., Cao, X., Fang, Y., Xie, X., and Wang, S. Inductive link prediction for nodes having only attribute information. *arXiv preprint arXiv:2007.08053*, 2020.

Kazemi, S. M., Buchman, D., Kersting, K., Natarajan, S., and Poole, D. Relational logistic regression. In *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.

Khosravi, H., Schulte, O., Hu, J., and Gao, T. Learning compact Markov logic networks with decision trees. *Machine Learning*, 89(3):257–277, 2012.

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

Kuzelka, O., Wang, Y., Davis, J., and Schockaert, S. Relational marginal problems: Theory and estimation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 1–8. AAAI Press, 2018.

Ma, T., Chen, J., and Xiao, C. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 7113–7124, 2018.

Meilicke, C., Chekol, M. W., Betz, P., Fink, M., and Stuckeschmidt, H. Anytime bottom-up rule learning for large-scale knowledge graph completion. *The VLDB Journal*, 33(1):131–161, 2024.

O'Bray, L., Horn, M., Rieck, B., and Borgwardt, K. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. In *International Conference on Learning Representations*, 2022.

Orbanz, P. and Roy, D. M. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):437–461, 2014.

Qian, Z. and Schulte, O. Factorbase: Multi-relational model learning with sql all the way. In *Data Science and Advanced Analytics (DSAA)*, pp. 1–10, 2015.

Qu, M., Chen, J., Xhonneux, L.-P., Bengio, Y., and Tang, J. RNNLogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations*, 2021.

Raedt, L. D., Kersting, K., Natarajan, S., and Poole, D. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis lectures on artificial intelligence and machine learning*, 10(2):1–189, 2016.

Raedt, L. d., Dumančić, S., Manhaeve, R., and Marra, G. From statistical relational to neuro-symbolic artificial intelligence. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4943–4950, 2020.

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

Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, proceedings 15*, pp. 593–607. Springer, 2018.

Schulte, O. and Gholami, S. Locally consistent bayesian network scores for multi-relational data. In *IJCAI*, pp. 2693–2700, 2017.

Schulte, O. and Khosravi, H. Learning graphical models for relational data via lattice search. *Machine Learning*, 88 (3):331–368, 2012.

Shirzad, H., Hassani, K., and Sutherland, D. J. Evaluating graph generative models with contrastively learned features. *Advances in Neural Information Processing Systems*, 2022.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018*, pp. 412–422. Springer, 2018.

Sun, Y. and Han, J. *Mining Heterogeneous Information Networks: Principles and Methodologies*, volume 3. Morgan & Claypool Publishers, 2012.

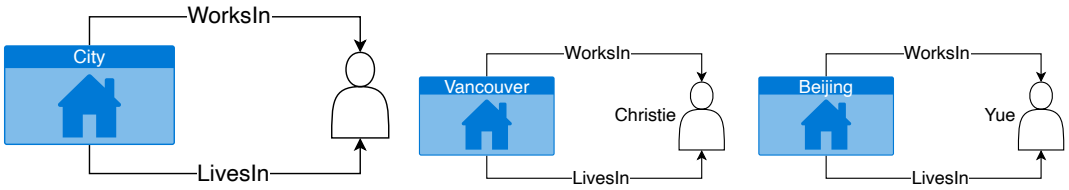
Thompson, R., Knyazev, B., Ghalebi, E., Kim, J., and Taylor, G. W. On evaluation metrics for graph generative models. In *International Conference on Learning Representations*, 2022.

Tian, Y., Pei, S., Zhang, X., Wang, W., Tong, H., and Chawla, N. V. (eds.). *Knowledge-enhanced Graph Learning*, 2024. Workshop at AAAI. URL https://www.yijuntian.com/tutorial.

van Bremen, T. and Kuzelka, O. Approximate weighted first-order model counting: Exploiting fast approximate model counters and symmetry. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4252–4258, 7 2020.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Van den Broeck, G. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 5502–5511. PMLR, 2018.

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. *In Proceedings of the Neural Information Processing Systems*, 32, 2019.



(a) Motif/Subgraph Pattern (b) Two example instances of the rule.

Figure 5. A motif for the rule "If X works in city Y, then X also lives in city Y" with example instances.

A. Appendix

A.1. Rules, Conjunctions, and Motifs

Figure 5 shows how moralization converts a rule to a conjunction, and how a conjunction represents a motif.

A.2. First-Order Logic Definitions

A positive **relationship literal** is of the form $R(U, V)$. A **negative** relationship literal is of the form $\neg R(U, V)$. A generic relationship literal (positive or negative) is denoted as $\ell(U, V)$. A **ground** relationship literal is of the form $\ell(U = u, V = v)$ where u and v are two node indices. Similar to specifying arguments for variables in a programming language, grounding specifies arguments for node variables. Positive and negative unary literals are defined similarly.

For a positive ground relationship literal $\ell(U = u, V = v) = R(U = u, V = v)$, the indicator $I_{\mathcal{G}}(\ell(U = u, V = v)) = 1$ if the two nodes u and v are linked by edge type R in graph G. For a negative ground relationship literal, $\ell(U = u, V = v) = \neg R(U = u, V = v)$, we have $I_{\mathcal{G}}(\neg R(U = u, V = v)) = 1 - I_{\mathcal{G}}(R(U = u, V = v))$.

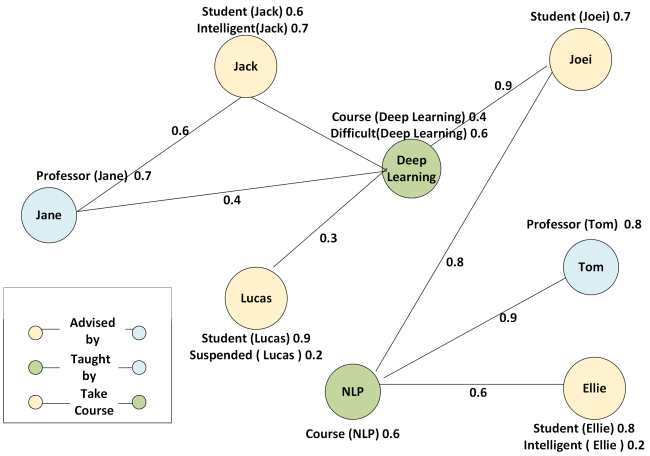


Figure 6. An example expected graph corresponding to Figure 3

A conjunction is a list of literals. Intuitively, a conjunction is a template for a motif or frequently occurring subgraph. The indicator function specifies which nodes satisfy the literal/conjunction. Our formal definitions are as follows.

A **conjunction** ϕ comprises three elements:

1. A list $\ell_1(U_1, V_1), \ldots, \ell_P(U_P, V_P)$ comprising P relationship literals where U_i and V_i are node variables.
2. A list $\ell_1(W_1), \ldots, \ell_Q(W_Q)$ comprising Q unary literals, where W_i is a node variable.
3. A set of equality constraints EQ of the form $D_k = E_k$ for any two node variables D_k and E_k that appear in the list of relationship literals or unary literals. Formally, EQ is a set of unordered pairs of node variables.

This definition is equivalent to the definition in the main paper: all node variables are assumed to occur exactly once in a conjunction, and the equality constraints specify which node variables must be mapped to the same node indices. Representing equality constraints in an explicit list facilitates the statement of our matrix multiplication algorithm.

A conjunction with P relationship literals $\ell_1(U_1, V_1), \ldots, \ell_P(U_P, V_P)$ is a **chain conjunction** if there is a permutation π of the literals such that the equality constraints comprise $V_{\pi(i-1)} = U_{\pi(i)}$ for $i = 2, \ldots, P$. A conjunction ϕ has $2P + Q$ parameters (i.e., node variables). Specifying a list of $2P + Q$ node indices as arguments to the conjunction returns a **grounded conjunction.** A grounded chain conjunction corresponds to a path in the graph where each consecutive pair of nodes is connected by an edge.

The indicator function for a grounded conjunction is given by:

$$I_{\mathcal{G}}(\ell_1(U_1 = u_1, V_1 = v_1), \ldots, \ell_P(U_P = u_P, V_P = v_P), \ell_1(W_1 = w_1), \ldots, \ell_Q(W_Q = w_Q)) = \prod_{i=1}^{P} I_{\mathcal{G}}(\ell_i(U_i = u_i, V_i = v_i)) \prod_{j=1}^{Q} I_{\mathcal{G}}(\ell_j(W_j = w_j)) \tag{3}$$

Conjunction Counts. For compactness, write a grounding as $\langle \boldsymbol{U} = \boldsymbol{u}, \boldsymbol{V} = \boldsymbol{v}, \boldsymbol{W} = \boldsymbol{w} \rangle$ so the indicator function returns a 0/1 value for $I_{\mathcal{G}}(\langle \boldsymbol{U} = \boldsymbol{u}, \boldsymbol{V} = \boldsymbol{v}, \boldsymbol{W} = \boldsymbol{w} \rangle, EQ)$. A grounding $\langle \boldsymbol{U} = \boldsymbol{u}, \boldsymbol{V} = \boldsymbol{v}, \boldsymbol{W} = \boldsymbol{w} \rangle$ is **valid** for a set of equality constraints EQ if for any two assignments $D = d$ and $E = e$ we have $d = e$ if and only if $(D = E) \in EQ$. Thus node variables constrained to be equal must be assigned the same node, and otherwise must be assigned different nodes. We write $Valid_{EQ}$ for the set of valid groundings. The **instance count** for a conjunction ϕ in a graph $\mathcal{G}$ returns the number of valid groundings that satisfy the conjunction:

$$n_\phi(\mathcal{G}) = \sum_{\langle \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \rangle \in Valid_{EQ}} I_{\mathcal{G}}(\langle \boldsymbol{U} = \boldsymbol{u}, \boldsymbol{V} = \boldsymbol{v}, \boldsymbol{W} = \boldsymbol{w} \rangle)$$

Here, we evaluate the indicator function for each combination of $\langle \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \rangle$, and sum up the values for all combinations to obtain the desired count of satisfying groundings.

A.3. Proof of Proposition 4.1

The output of VGAE+ model is an expected graph like Figure 6. The following proposition states how the expected counts can be computed based on a expected graph.

Proposition A.1. *The expected conjunction count given a set of node embeddings can be computed as the conjunction count in the expected graph:* $E_{\mathcal{G} \sim P(\mathcal{G}|\boldsymbol{z})}[n_\phi(\mathcal{G})] = n_\phi(\bar{\mathcal{G}}_{\boldsymbol{z}})$

Proof. Without loss of generality, assume the conjunction is of the form $\ell_1(U_1 = u_1, V_1 = v_1), \ldots, \ell_P(U_P = u_P, V_P = v_P), \ell_1(W_1 = w_1), \ldots, \ell_Q(W_Q = w_Q)$ with equality constraints EQ. Define the random variables $l_i^{uv}, i = 1, \ldots, P$ to return the indicator value $I_{\mathcal{G}}(\ell_i(U_i = u, V_i = v)$ and $l_j^w, j = 1, \ldots, Q$ to return the indicator value $I_{\mathcal{G}}(\ell_j(W_j = w))$. Then

$$
\begin{aligned}
E_{\mathcal{G}\sim P(\mathcal{G}|\boldsymbol{z})}[n_\phi(\mathcal{G})] = E[\sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} \prod_{i=1}^{P} I_{\mathcal{G}}(\ell_i(U_i = u_i, V_i = v_i)) \prod_{j=1}^{Q} I_{\mathcal{G}}(\ell_j(W_j = w_j))] \\
= E[\sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} \prod_{i=1}^{P} l_i^{\boldsymbol{u}_i\boldsymbol{v}_i} \prod_{j=1}^{Q} l_j^{\boldsymbol{w}_j}] \\
= \sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} E[\prod_{i=1}^{P} l_i^{\boldsymbol{u}_i\boldsymbol{v}_i} \prod_{j=1}^{Q} l_j^{\boldsymbol{w}_j}] \\
= \sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} \prod_{i=1}^{P} E[l_i^{\boldsymbol{u}_i\boldsymbol{v}_i}] \prod_{j=1}^{Q} E[l_j^{\boldsymbol{w}_j}] \qquad (4) \\
= \sum_{\langle \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\rangle \in Valid_{EQ}} \prod_{i=1}^{P} p_{\tilde{\mathcal{G}}_{\boldsymbol{z}}}(\ell_i(U_i = \boldsymbol{u}_i, V_i = \boldsymbol{v}_i)) \prod_{j=1}^{Q} p_{\tilde{\mathcal{G}}_{\boldsymbol{z}}}(\ell_j(W_j = \boldsymbol{w}_j)) \\
= n_\phi(\tilde{\mathcal{G}}_{\boldsymbol{z}})
\end{aligned}
$$

Equation (4) follows because the expectation of a product of independent random variables is the product of their expectations. The random variables $l_i^{\boldsymbol{u}_i\boldsymbol{v}_i}$ and $l_j^{\boldsymbol{w}_j}$ are independent because the (in)equality constraints ensure that in a valid grounding, no two different literals are ground to the same ground literal. And conditional on the node embeddings $\boldsymbol{z}$, any two different ground literals are independent. □

A.4. Matrix Multiplication Method

We next formulate the proposition that for every chain conjunction, there is a corresponding sequence of matrix multiplication operations such that: for every input graph $\mathcal{G}$, applying the operation sequence to the graph edge label tensor returns the instance count. In this formulation, we use these facts: 1) Every ground positive (negative) relationship literal corresponds to a link present (absent) in the graph. 2) A grounded chain conjunction corresponds to a path in the graph where each consecutive pair of nodes is connected by a present/absent link. Intuitively, a chain conjunction is a template for a motif or frequently occurring subgraph. The indicator function specifies which nodes satisfy the literal/conjunction. Our formal definitions are as follows.

The input to our counting algorithm is a chain conjunction $\{\ell_1(U_1, V_1), \ldots, \ell_P(U_P, V_P), \ell_1(W_1), \ldots, \ell_Q(W_Q), EQ\}$ and an input graph $\mathcal{G}$. The first step is to process the unary literals by masking adjacency matrix entries of nodes that do not satisfy all unary literals. The second step is to define inductively a sequence of matrices such that the instance count of the conjunction can be computed as the entry sum of the matrix product. We use $A \circ B$ to denote the element-wise matrix (Hadamard) product and I for the identity matrix of the appropriate dimension. A positive relationship literal $R(U, V)$ is **associated with** $\mathbf{A}_r$, the adjacency matrix for relation R. A negative relationship literal $\neg R(U, V)$ is associated with $\neg\mathbf{A}_r$ where $\neg\mathbf{A}_r[u, v] = 1 - \mathbf{A}_r[u, v]$ for all node indices u, v.

Step 1: Unary Literals Consider binary relationship literal $\ell_i(U, V)$ with associated $m \times n$ adjacency matrix $A_i(\mathcal{G})$. We search for every unary literal $\ell(W)$ where $(U = W) \in EQ$. For each such literal, we create a binary vector T of size m such that $T[w] := I_{\mathcal{G}}(\ell(W = w))$. Thus the entry $T[w]$ masks all the nodes that do not satisfy the unary literal. We apply the mask to the w row of matrix A_i, setting $\bar{\mathbf{A}}_i[w, :] = A_i[w, :] \circ T[w]$. where $A(w, :)$ represents the entire w row of matrix A. If the w entry of T is zero, the entire row of $\bar{\mathbf{A}}_i$ is set to 0. If the w entry of T is one, the entire row of A_i is copied to $\bar{\mathbf{A}}_i$.

Similarly, if a unary literal $\ell(W)$ exists where $(V = W) \in EQ$, we mask the corresponding column entries in the adjacency matrix A_i, and repeat the masking process for all such unary literals. We refer to the adjacency matrix that incorporates the unary functor constraints as the *masked adjacency matrix* $\bar{\mathbf{A}}_i$.

Step 2: Binary Literals A chain conjunction is centered if all equality constraints for the binary literals (other than the chain constraints) involve the first node variable, that is they are of the form $U_1 = E_k$. For a centered chain, we define a sequence of matrix multiplications as follows.

1. For a single literal conjunct $\phi = \ell(U, V)$ with associated masked matrix $\bar{\mathbf{A}}$, let

$$O_1(\phi) = \begin{cases} \bar{\mathbf{A}}, & \text{if } U = V \notin EQ \\ \bar{\mathbf{A}} \circ I, & \text{if } U = V \in EQ \end{cases}$$

   $\bar{\mathbf{A}} \circ I$ agrees with $\bar{\mathbf{A}}$ on the diagonal and is 0 off-diagonal.

2. Inductively, consider a conjunction ϕ of length k+1 in the form of $\phi = \phi', \ell_{k+1}(U_{k+1}, V_{k+1})$ where $\phi' = \ell_1(U_1, V_1), \ldots, \ell_k(U_k, V_k)$ is a conjunction of length k. Let

$$O_{k+1}(\phi) = \begin{cases} O_k(\phi')\bar{\mathbf{A}}_{k+1}, & \text{if } U_1 = V_{k+1} \notin EQ \\ (O_k(\phi')\bar{\mathbf{A}}_{k+1}) \circ I, & \text{if } U_1 = V_{k+1} \in EQ \end{cases} \tag{5}$$

Proposition A.2. *The (u, v)-th entry of O_k counts the number of groundings of a centered chain conjunction ϕ of length k in a graph $\mathcal{G}$ where $U_1 = u$ and $V_P = v$. Therefore*

$$n_\phi(\mathcal{G}) = \sum(O_k(\phi)).$$

Proof. Base case, $k = 1$. If $\phi = \{\ell(U, V)\}$, then the conjunction count is the number of pairs (u, v) such that (i) both groundings $U = u$ and $V = v$ satisfy all unary literals, and (ii) $I_{\mathcal{G}}(\ell(U = u, V = V)) = 1$. All and only such pairs have the entry $\bar{\mathbf{A}}[u, v] = 1$ in the masked adjacency matrix associated with $\ell(U, V)$.

Case 1: $(U = V) \notin EQ$. Then the number of satisfying groundings is simply given by $\sum(\bar{\mathbf{A}})$.

Case 2: $(U = V) \in EQ$. Then the satisfying groundings are of the form $U = u, V = u$, so their count is given by the matrix trace of $\bar{\mathbf{A}}$, or equivalently $\sum(\bar{\mathbf{A}} \circ I)$. This establishes the base case.

Inductive Step: Assume the proposition holds for k and consider the matrix O_{k+1} computed by Equation (5). By the inductive hypothesis, the (u, v)-th entry of O_k counts the number of instantiations of length k between vertices u and v that satisfy ϕ'. Now, the number of instantiations of length $k + 1$ between u and w equals the number of instantiations of length k from vertex u to each vertex v that has ℓ_{k+1} relation with w. The non-zero entries of column w of masked matrix $\bar{\mathbf{A}}_{k+1}$ represent vs related by ℓ_{k+1} to w. So, (u, w)-th entry of $O_k\bar{\mathbf{A}}_{k+1}$ gives the number of instantiations between u and w satisfying the centered conjunction and all equality constraints except possibly $U_1 = V_{k+1}$. Therefore for the case where $U_1 = V_{k+1} \notin EQ$, the matrix O_{k+1} satisfies the inductive hypothesis. For the case where $U_1 = V_{k+1} \in EQ$, we observe that the number of instantiations of length $k + 1$ from node u to u equals the (u, u) diagonal entry of $O_k\bar{\mathbf{A}}_{k+1}$ or equivalently, $(O_k\bar{\mathbf{A}}_{k+1}) \circ I$. Thus the total number of satisfying groundings is given by $\sum(O_{k+1}(\phi))$ in either case, which establishes the inductive hypothesis. □

A.5. Dataset Statistics

To evaluate all the methods we utilize 4 datasets used in previous studies of generative models (Kipf & Welling, 2016; Yun et al., 2019; Hao et al., 2020).

- **Cora** (Kipf & Welling, 2016) 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.
- **ACM** (Yun et al., 2019) 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.
- **IMDb** (Yun et al., 2019) 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.
- **CiteSeer** CiteSeer (Kipf & Welling, 2016) 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.