

A APPENDIX / SUPPLEMENTAL MATERIAL

A.1 DATASET INFORMATION

To evaluate all the methods we utilize 5 datasets used in previous studies of generative models Kipf and Welling (2016); Yun *et al.* (2019); Hao *et al.* (2020).

- **Cora** (Kipf and Welling, 2016) is a citation dataset that consists of nodes that represent machine-learning papers divided into seven classes (the subjects of the papers) and links that represent citation between them. The target for node classification is the subject of the papers. This dataset has 5,429 links, 2,708 nodes with an average node degree of 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. The target for node classification is predicting one of three labels corresponding to the conferences where the papers were published. This dataset has 18,929 links, 8,993 nodes with an average node degree of 2.209.
- **IMDb** (Yun *et al.*, 2019) is a movie dataset with three types of nodes (movies, actors, and directors) and it uses the genre of movies as their labels. The target for node classification is predicting one of three genres of movies. This dataset has 19,120 links, 12,772 nodes with an average node degree of 2.9.
- **CiteSeer** (Kipf and Welling, 2016) is also a citation dataset that consists of nodes that represent machine-learning papers divided into six classes (the topics of the papers) and links that represent citation between them. The target for node classification is the topic of the publications. This dataset has 4,732 links, 3,327 nodes with an average node degree of 2.7.
- The **UW** dataset models academic relationships at a university, where persons can be both students and professors and key relations include AdvisedBy (a student advised by a professor) and TaughtBy (a student taking a course taught by a professor). The dataset contains 278 person entities and 132 course entities. The classification target is the phase of study of the student, which can be one of three labels: *pre-quals*, *post-quals*, or *post-generals*. If the person is not a student, the label is 0.
- **Photo & Computers** are datasets from the Amazon co-purchase graph (McAuley *et al.*, 2015). In these datasets, 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 (Hao *et al.*, 2020).

Data Preprocessing Following previous work (Kipf and 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.

A.2 RESULTS FOR HETEROGENEOUS ACM AND IMDB

We present graph generation results for the original heterogeneous versions of IMDb and ACM, which feature different types of nodes and links. For ease of comparison, we repeat the results for homogeneous versions from the main paper. Heterogeneous versions are distinguished by an asterisk. We observe large percentage improvements from rule enhancement, especially for IMDb.

Table 4: Mean \pm Std for Count Distance (CD \downarrow) and Graph Realism (MMD \downarrow) with Improvements (in scientific notation, so "e" represents $\times 10^n$).

Dataset	Count Distance (MSE)			Graph Realism (MMD)		
	VGAE+	VGAE+R	Improv. (%)	VGAE+	VGAE+R	Improv. (%)
ACM	$1.24e5 \pm 4.86e3$	$2.81e4 \pm 3.54e3$	77.34	$3.31e20 \pm 1.32e20$	$3.29e18 \pm 3.08e18$	99.01
ACM*	$3.65e5 \pm 3.32e3$	$1.47e4 \pm 2.54e3$	95.97	$1.9e15 \pm 1.32e15$	$9.8e15 \pm 0.98e15$	-415.79
IMDb	$5.86e5 \pm 1.96e4$	$3.23e5 \pm 1.34e5$	44.81	$2.94e23 \pm 6.29e22$	$4.58e22 \pm 3.25e22$	84.42
IMDb*	$1.37e5 \pm 2.23e2$	$1.43e4 \pm 1.98e2$	89.56	$1.9e20 \pm 2.88e18$	$7.8e17 \pm 1.32e17$	99.59

For node classification on heterogeneous datasets, the class labels are as follows: For IMDB*, predict the genre of the movie nodes, and for ACM*, predict the area of a paper. As with the homogeneous versions, rule enhancement does not lead to substantive improvements on these datasets, because the rules are not informative for node classification in these datasets.

Table 5: AUC and F1 Score for link prediction on heterogeneous ACM and IMDB (higher is better).

Dataset	Model	AUC	F1 Score
ACM*	VGAE+R	0.976 ± 0.004	0.9184 ± 0.011
	VGAE+	0.979 ± 0.003	0.9123 ± 0.011
IMDb*	VGAE+R	0.774 ± 0.006	0.632 ± 0.008
	VGAE+	0.769 ± 0.011	0.615 ± 0.014

A.3 NODE CLASSIFICATION BASELINE METHODS

Table 6 compares the rule-enhanced VGAE+R with the following node classification baselines. We focused on baselines that are similar to generative models in that they aim to support multiple prediction tasks. The baselines apply to homogeneous graphs only, so we homogenized ACM and IMDB (Mahmoudzadeh *et al.*, 2024) and omitted UW.

Generalizable Graph Masked AutoEncoder (GiGaMAE) (Shi *et al.*, 2023) introduces a novel graph encoder based on aligning different node embeddings that respectively encode structural and feature information. **MVGRL** is an inductive self-supervised approach for learning representations of nodes and graphs by contrasting different structural views of graphs (Hassani and Khasahmadi, 2020). $\mathcal{G}^2\text{Pxy}$ (Zhang *et al.*, 2023) generates proxy nodes to support classification. For all methods we used the authors’ code to train a node classifier and generate class labels.

Dataset	Model	AUC	F1-score
Cora	VGAE+R	0.965 ± 0.013	0.887 ± 0.016
	GiGaMAE	0.920	0.856
	$\mathcal{G}^2\text{P}_{xy}$	0.921	0.781
	MVGRL	0.888	0.886
CiteSeer	VGAE+R	0.903 ± 0.008	0.794 ± 0.042
	GiGaMAE	0.842	0.798
	$\mathcal{G}^2\text{P}_{xy}$	0.850	0.781
	MVGRL	0.807	0.710
Computers	VGAE+R	0.915 ± 0.022	0.827 ± 0.047
	GiGaMAE	0.941	0.770
	$\mathcal{G}^2\text{P}_{xy}$	0.680	0.578
	MVGRL	0.983	0.816
Photo	VGAE+R	0.991 ± 0.003	0.972 ± 0.002
	GiGaMAE	0.963	0.569
	$\mathcal{G}^2\text{P}_{xy}$	0.773	0.513
	MVGRL	0.963	0.960
ACM	VGAE+R	0.761 ± 0.084	0.525 ± 0.014
	GiGaMAE	0.823	0.440
	$\mathcal{G}^2\text{P}_{xy}$	0.742	0.661
	MVGRL	0.708	0.803
IMDb	VGAE+R	0.829 ± 0.006	0.697 ± 0.008
	GiGaMAE	0.890	0.457
	$\mathcal{G}^2\text{P}_{xy}$	0.654	0.541
	MVGRL	0.788	0.653

Table 6: Node classification results comparing rule-enhanced graph generation against baselines.

A.4 FORMULA LEARNING

In principle, any Markov Logic Network structure learning method can be used with our moment-matching training objective. We deployed the Factorbase system (Qian and Schulte, 2015) for several reasons.

1. Interpretability: The first phase of Factorbase uses Bayesian network structure learning method to discover a *causal graph* that provides a perspicuous visual representation of all associations learned from the data.
2. Scalability: Factorbase scales to larger and more complex datasets than other MLN structure learning method, on the order of 100K nodes.
3. Quality: Research has shown that the formulas discovered by Factorbase support high quality inferences in relational prediction tasks.

Factorbase has three main phases.

1. Causal Graph Discovery.
2. Rule Extraction from the learned causal graph.
3. Moralization: convert learned rules to conjunctions.

Both rule extraction and moralization are computationally straightforward linear-time operations. Moralization is a method for converting learned rules to conjunctions. Previous research in statistical-relational learning (Kazemi *et al.*, 2014; Khosravi *et al.*, 2012) has shown that moralization is a strong method for converting learned rules to useful features for generative graph models. Note that therefore *our matrix multiplication algorithm can be applied to any set of learned rules after moralization*. This means that our algorithm for rule enhancement can be applied to any set of learned rules, not only those obtained from MLN structure learning. We next explain in the phases of Factorbase formula discovery and illustrate the results on our dataset.

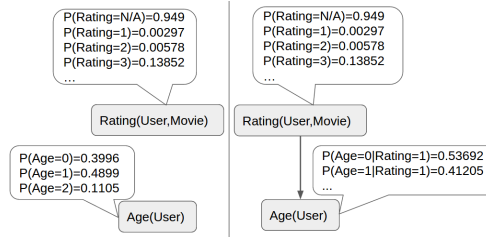


Figure 5: Example First-Order Causal Graphs: left = B_1 with graph \mathcal{G}_1 , right = B_1^+ with graph \mathcal{G}_1^+ .

Causal Graph Discovery A **causal graph (CG) structure** is a directed acyclic graph G (DAG) whose nodes comprise a set of random variables Pearl (2000). The interpretation is that the parents of a node represent its direct causes Pearl (2000). A **causal Bayesian network** B is a structure G together with a set of parameter values of the form $P(\text{child_value}|\text{parent_values})$, which specify the distribution of a child node given an assignment of values to its parent node. A first-order causal graph Wang *et al.* (2008); Kimmig *et al.* (2014) (FOCG) is a CG whose nodes are first-order terms. The CG parameters specify the distribution of a child node given an assignment of values to its parent node. Figure 5 shows two parametrized FOCGs. The right FOCG connects the rating of a movie to the age of the rater. In the left FOCG, ratings are independent of ages. The process of structure learning searches for statistically significant connections between first-order terms to introduce or remove edges (Schulte and Qian, 2018; Schulte and Gholami, 2017).

We show the causal graphs learned by Factorbase for three datasets. We find that many of the learned rules capture intuitively plausible domain constraints, such as homophily: if one paper cites another, then they are likely from the same research area. For example in the graph for the Cora dataset, the parents of the target label of a paper (node) are paper features 1 and 2, the citation relationship between a paper and the target paper (denoted “edges_table”), and the label of the citing paper. The

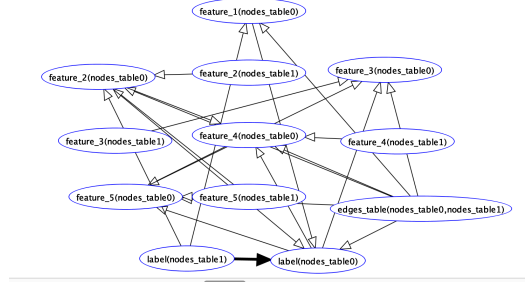


Figure 6: The causal graph for the Cora dataset. We have highlighted the homophily edge that shows that if paper 0 cites paper 1, then the label of paper 0 correlates with the label of paper 1.

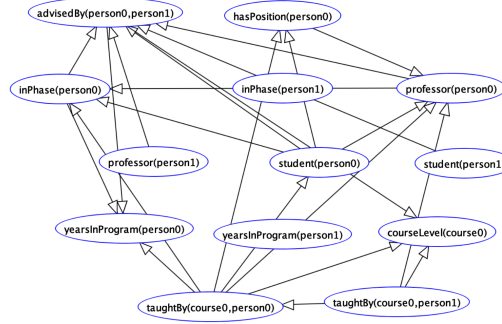


Figure 7: The causal graph for the UW dataset.

graph defines a rule for each combination of the 4 parent nodes, and each of the 7 possible child node values (paper classes).

ACM and CiteSeer are citation networks like Cora with a similar graph, which we omit. Figure 7 shows the causal graph for the UW dataset. The graph captures a number of relationships that make sense in the university domain. For example whether a person teaches a course correlates with whether they have a position.

Figure 8 shows the causal graph for the IMDb dataset. The graph captures many correlations between features of movies and actors. An interesting aspect of the graph is that for the target label (genre of the movie), its only parents are other features of the movie to be classified (features 1,2,4). This suggests that relational features, such as the features of actors who appear in the movie, should not add information beyond movie features. We verified this directly by building a classifier based on movie features only, which performed better than the GNN that uses relational features.

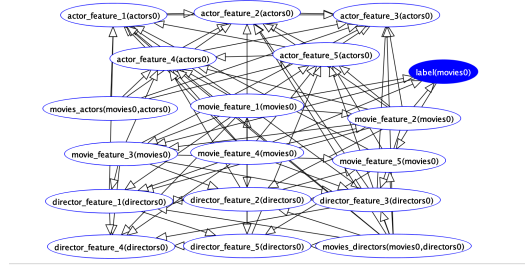


Figure 8: The causal graph for the IMDb dataset.

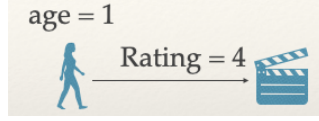


Figure 9: The FO conjunction $Age(User) = 1, Rating(User, Movie) = 4$ can be viewed as a two-node motif. The conjunction is obtained from the rule $Age(User) = 1 \rightarrow Rating(User, Movie) = 4$ by moralization.

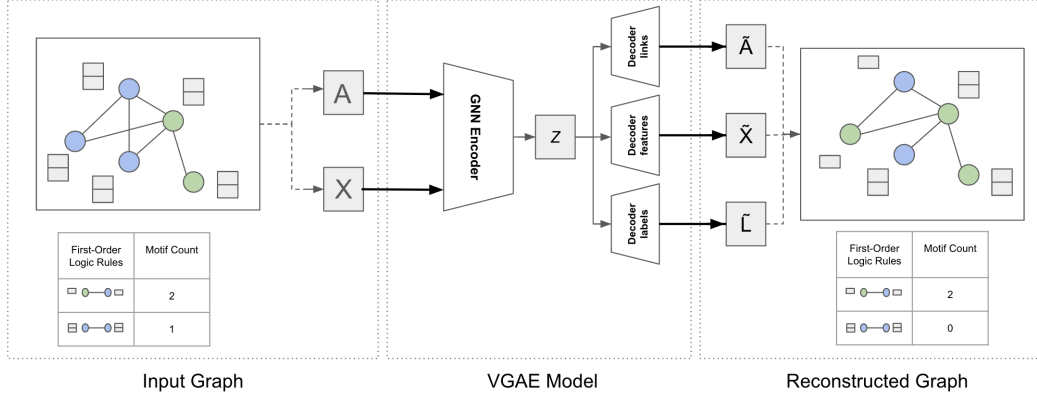


Figure 10: Encoder-Decoder Training Architecture for the VGAE+ Model.

Rule Extraction from Causal Graphs A rule is of the form $B \rightarrow H$ where the body B is an FO conjunction and the head H is a single FO literal. An example rule is

$$Age(User) = 1 \rightarrow Rating(User, Movie) = 4.$$

This rule expresses the knowledge that the age level of a user influences their ratings.

A single rule $B \rightarrow H$ is defined by a combination $parent_values \rightarrow child_value$, for each child node value and possible combination of parents values (Khosravi *et al.*, 2012). For example, the causal graph of Figure 5(right) defines the rule

$$Rating(User, Movie) = 3 \rightarrow Age(User) = 1).$$

Given 6 possible rating values and 5 age levels, the link $Rating(User, Movie) \rightarrow Age(User)$ defines $6 \times 5 = 30$ rules.

As the Factorbase system can output many rules, we use

$2n(\mathbf{B}, H) \ln(n(H|\mathbf{B})/n(H)) - \ln n(H)$ as a *rule quality metric* for pruning. The conditional count is given by $n(H|\mathbf{B}) = n(H, \mathbf{B})/n(\mathbf{B})$. 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 and Gholami, 2017). The quality metric can be interpreted as a positive local BIC model selection score. We keep all rules whose quality metric is above 0. Rule pruning reduces the number of rules for scalability, and also simulates the impact of a domain expert selecting the most important rules.

Moralization: From rules to conjunctions Most rule learners are based on *discriminative* learning, building on classification techniques to search for bodies that predict the head. A question researched in SRL is how to convert a set of rules to a set of graph features/statistics that support *generative* graph modelling. The recommended answer is *moralization*: convert each rule to a conjunction $\phi = (B, H)$ (Domingos and Richardson, 2007, 12.5.3), (Kazemi *et al.*, 2014). Figure 9 illustrates how moralization converts a rule to a conjunction, and how a conjunction represents a motif.

A.5 DETAILS ON THE VGAE+R MODEL

Figure 10 shows the VGAE+R architecture. The graph **encoder** $q_\phi(z|\mathbf{X}, A)$ 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. Following Mahmoudzadeh *et al.* (2024), we used a multi-layer graph attention network (GAT) and an RGCN (Schlichtkrull *et al.*, 2018) for heterogeneous datasets. The encoders were configured in the same way as Mahmoudzadeh *et al.* (2024).

The decoders generate links, node features, and node labels independently as follows.

$$p_{\eta}(\mathbf{A}|\mathbf{z}) = \prod_{r=1}^T \prod_{u,v} p_{\eta_r}(\mathbf{A}_r[u, v]|\mathbf{z}[u], \mathbf{z}[v]) \quad (5)$$

$$p_{\psi}(\mathbf{X}|\mathbf{z}) = \prod_u p_{\psi}(\mathbf{X}[u]|\mathbf{z}[u]) \quad (6)$$

$$p_{\phi}(\mathbf{L}|\mathbf{z}) = \prod_u p_{\phi}(\mathbf{L}[u]|\mathbf{z}[u]), \quad (7)$$

Jaeger and Schulte (2020) provide a strong theoretical foundation for the independent decoder model. Following Mahmoudzadeh *et al.* (2024), we use a stochastic blocks model for the link decoder Equation (5), and MLPs for the feature and label decoders Equations (6) and (7). When training a model for graph generation, we use the training objective Equation (2), without the node labels. When training a model for node classification, the encoder does not see the node labels, but the decoder has access to the training node labels to evaluate the node label term Equation (7). At test time, the trained encoder is applied to the available graph information to produce node embeddings for each test node, and then the trained decoder is applied to output class probabilities.

Hyperparameters We used the ADAM optimizer for training, with default hyperparameter values. For *graph generation* we obtained the best results for both the VGAE+ and the VGAE+R model by setting all ELBO hyperparameters α, β, γ to 1, similar to Simonovsky and Komodakis (2018). The motif loss weight λ was also set to 1 except for Computers, where $\lambda = 2$ yielded better results. Both models were trained for 100 epochs on all datasets.

For *node classification*, for the VGAE+ baseline model with $\lambda = 0$, we follow Mahmoudzadeh *et al.* (2024) and use Bayesian optimization to fix the ELBO hyperparameters α, β, γ . For the VGAE+R model, after fixing the ELBO hyperparameters, we set the motif loss weight λ empirically based on a validation set. We found that different datasets require different training regimes to build good node classifiers. Specifically, the number of training epochs were as follows for both VGAE+ and VGAE+R models: 700 for Cora and CiteSeer, 1000 for IMDB, Computers, ACM, Photo, and 100 for UW.

For the learning curves, we trained the model for 1000 epochs at each sample size on the Citeseer dataset. To obtain standard deviations, we initialized training with five different seeds for node classification, and three different seeds for graph generation.

Computing Resources The experiments were conducted using a GPU cluster. The compute resources varied based on dataset size. For large datasets such as Computers and Photos, experiments were run on NVIDIA A100 GPUs with 80GB VRAM, 64GB RAM, and 16 CPU threads. For IMDB and ACM, NVIDIA A40 GPUs with 48GB VRAM, 64GB RAM, and 16 CPU threads were used. For smaller datasets such as Cora, Citeseer, and UW, experiments were conducted on NVIDIA GeForce GTX 1080 Ti GPUs. Each dataset required approximately 20GB of storage for generated graphs. Given the training regime described, generating measurements for one dataset took approximately 4 to 5 hours.

A.6 FIRST-ORDER SEMANTIC LOSS VS. PROPOSITIONAL SEMANTIC LOSS

In this section we show how the semantic loss Equation (4) with the distance function

$$|\ln n_i(D) - \ln E_{\theta}[n_i|\mathbf{z}]| \quad (8)$$

reduces to the semantic loss function of Xu *et al.* (2018) in the case of a propositional formula ϕ_i . For a given graph, a propositional formula is either true or false in the graph. Xu *et al.* give the

example of an "exactly-one" constraint that says that each node is assigned exactly to exactly one class. Translating into our notation, a propositional formula has an instance count of 1 if it is satisfied in a graph, and 0 otherwise:

$$n_i(\mathcal{G}) \leq 1 \text{ for propositional } \phi_i \quad (9)$$

The basis idea of Xu *et al.* is to view a propositional formula as a constraint that a model should satisfy. There the semantic loss function should maximize the probability of the formula under the model. The probability of the formula is the probability of the graphs that satisfy it. The steps in the proof are as follows.

$$p_{\theta}(\phi_i|\mathbf{z}) = \sum_{\mathcal{G}} P_{\theta}(\mathcal{G}|\mathbf{z}) n_i(\mathcal{G}) = E_{\theta}[n_i|\mathbf{z}] \quad (10)$$

$$-\ln p_{\theta}(\phi_i|\mathbf{z}) = -\ln E_{\theta}[n_i|\mathbf{z}] \quad (11)$$

$$|\ln n_i(D) - \ln E_{\theta}[n_i|\mathbf{z}]| = |0 - \ln E_{\theta}[n_i|\mathbf{z}]| = |-\ln p_{\theta}(\phi_i|\mathbf{z})| = -\ln p_{\theta}(\phi_i|\mathbf{z}) \quad (12)$$

Equation (10) holds because due to the zero-one condition (9) the probability of a formula is the same as the expected instance count, which immediately implies Equation (11). For Equation (12), since formula ϕ_i is assumed true in *all* graphs, in particular it will be true in the observed graph D . Therefore $n_i(D) = 1$. The last equality holds because for $p_{\theta} \leq 0$, we have $-\ln p_{\theta}(\phi_i|\mathbf{z}) \geq 0$, so $-\ln p_{\theta}(\phi_i|\mathbf{z}) = |-\ln p_{\theta}(\phi_i|\mathbf{z})|$. The leftmost term in Equation (12) is our first-order semantic loss, and the rightmost is the propositional first-order semantic loss of Xu *et al.*, which establishes our claim.

A.7 EXPANDED FIRST-ORDER LOGIC DEFINITIONS

We add to the definitions in the main body to introduce concepts we need in our proofs and in the full description of matrix multiplication algorithm. We make this section self-contained for ease of reference. 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 $I_{\mathcal{G}}(\neg R(U = u, V = v)) = 1 - I_{\mathcal{G}}(R(U = u, V = v))$.

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 $\ell_1(U_1, V_1), \dots, \ell_P(U_P, V_P)$ comprising P relationship literals where U_i and V_i are node variables.
2. A list $\ell_1(W_1), \dots, \ell_Q(W_Q)$ 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 $\ell_1(U_1, V_1), \dots, \ell_P(U_P, V_P)$ 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 $i = 2, \dots, P$. 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:

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

Conjunction Counts. For compactness, write a grounding as $\langle U = u, V = v, W = w \rangle$ so the indicator function returns a 0/1 value for $I_{\mathcal{G}}(\langle U = u, V = v, W = w \rangle, EQ)$. A grounding $\langle U = u, V = v, W = w \rangle$ 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:

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

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

A.8 PROOF OF PROPOSITION 2

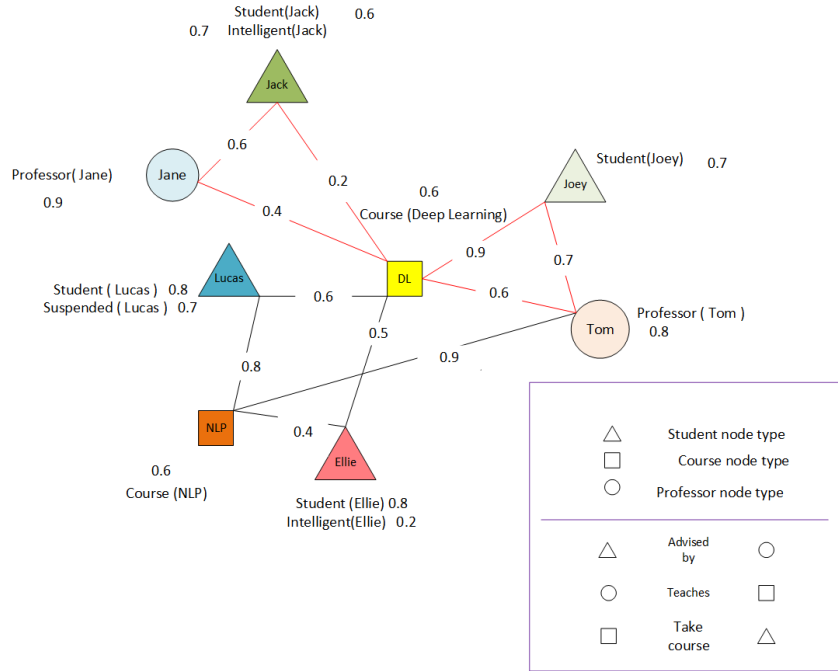


Figure 11: An example expected graph corresponding to Figure 4.

The output of the VGAE+ model is an expected graph like Figure 11. The following proposition states that the expected counts can be computed based on a expected graph. We use the full first-order logic definitions of Appendix A.7.

Proposition 2. *The expected conjunction count given a set of node embeddings can be computed as the conjunction count in the expected graph: $E_{\mathcal{G} \sim P(\mathcal{G}|\mathbf{z})}[n_\phi(\mathcal{G})] = n_\phi(\tilde{\mathcal{G}}_{\mathbf{z}})$*

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

$$\begin{aligned}
& E_{\mathcal{G} \sim P(\mathcal{G}|\mathbf{z})}[n_\phi(\mathcal{G})] \\
&= E\left[\sum_{\langle \mathbf{u}, \mathbf{v}, \mathbf{w} \rangle \in Valid_{EQ}} \prod_{i=1}^P I_{\mathcal{G}}(\ell_i(U_i = u_i, V_i = v_i)) \prod_{j=1}^Q I_{\mathcal{G}}(\ell_j(W_j = w_j))\right] \\
&= E\left[\sum_{\langle \mathbf{u}, \mathbf{v}, \mathbf{w} \rangle \in Valid_{EQ}} \prod_{i=1}^P l_i^{u_i v_i} \prod_{j=1}^Q l_j^{w_j}\right] \\
&= \sum_{\langle \mathbf{u}, \mathbf{v}, \mathbf{w} \rangle \in Valid_{EQ}} E\left[\prod_{i=1}^P l_i^{u_i v_i} \prod_{j=1}^Q l_j^{w_j}\right] \\
&= \sum_{\langle \mathbf{u}, \mathbf{v}, \mathbf{w} \rangle \in Valid_{EQ}} \prod_{i=1}^P E[l_i^{u_i v_i}] \prod_{j=1}^Q E[l_j^{w_j}] \\
&= \sum_{\langle \mathbf{u}, \mathbf{v}, \mathbf{w} \rangle \in Valid_{EQ}} \prod_{i=1}^P p_{\tilde{\mathcal{G}}_{\mathbf{z}}}(\ell_i(U_i = u_i, V_i = v_i)) \prod_{j=1}^Q p_{\tilde{\mathcal{G}}_{\mathbf{z}}}(\ell_j(W_j = w_j)) = n_\phi(\tilde{\mathcal{G}}_{\mathbf{z}})
\end{aligned} \tag{13}$$

Equation (13) follows because the expectation of a product of independent random variables is the product of their expectations. The random variables $l_i^{u_i v_i}$ and $l_j^{w_j}$ 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 \mathbf{z} , any two different ground literals are independent. \square

A.9 MATRIX MULTIPLICATION METHOD

A.9.1 EXAMPLE FOR EXPECTED INSTANCE COUNT

Figure 12 illustrates the matrix multiplication method for the expected graph in our example.



Figure 12: The matrix multiplication sequence for our example conjunction and Figure 11 graph.

A.9.2 FULL SPECIFICATION OF MATRIX MULTIPLICATION METHOD

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 $\{\ell_1(U_1, V_1), \dots, \ell_P(U_P, V_P), \ell_1(W_1), \dots, \ell_Q(W_Q), EQ\}$ 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 $\phi' = \ell_1(U_1, V_1), \dots, \ell_k(U_k, V_k)$ is a conjunction of length k . Let

$$O_{k+1}(\phi) = \begin{cases} O_k(\phi') \bar{\mathbf{A}}_{k+1}, & \text{if } U_1 = V_{k+1} \notin EQ \\ (O_k(\phi') \bar{\mathbf{A}}_{k+1}) \circ I, & \text{if } U_1 = V_{k+1} \in EQ \end{cases} \quad (14)$$

A.9.3 CORRECTNESS PROOF (PROPOSITION 3)

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 the following 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.

Proposition 3. Let $\{\ell_1(U_1, V_1), \dots, \ell_P(U_k, V_k), \ell_1(W_1), \dots, \ell_Q(W_Q), EQ\}$ be a chain conjunction of length k .

1. For an input graph \mathcal{G} , the (u, v) -th entry of O_k counts the number of groundings of ϕ in \mathcal{G} where $U_1 = u$ and $V_P = v$. Therefore $n_\phi(\mathcal{G}) = \sum (O_k(\phi))$.

2. For an expected graph \tilde{G}_z , the (u, v) -th entry of O_k counts the expected number of groundings of ϕ where $U_1 = u$ and $V_P = v$. Therefore $n_\phi(\tilde{G}_z) = \sum(O_k(\phi))$.

Proof. We give the proof for clause 1, counting observed counts in an input graph \mathcal{G} . The argument for expected counts computed from an expected graph is exactly parallel.

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) $I_G(\ell(U = u, V = v)) = 1$. All and only such pairs have the entry $\bar{A}[u, v] = 1$ 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 $\sum(\bar{A})$.

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{A} , or equivalently $\sum(\bar{A} \circ I)$. This establishes the base case.

Inductive Step: Assume the proposition holds for k and consider the matrix O_{k+1} computed by Equation (14). 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{A}_{k+1} represent v s related by ℓ_{k+1} to w . So, (u, w) -th entry of $O_k \bar{A}_{k+1}$ 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 $O_k \bar{A}_{k+1}$ or equivalently, $(O_k \bar{A}_{k+1}) \circ I$. Thus the total number of satisfying groundings is given by $\sum(O_{k+1}(\phi))$ in either case, which establishes the inductive hypothesis. \square

Extensions

- Our counting method is based on a **sorting algorithm**. A sorting algorithm is a procedure that takes a set of relationship literals

$$\ell_1(U_1, V_1), \dots, \ell_P(U_P, V_P)$$

and determines if there exists a permutation π such that the literals can be arranged to form a *chain conjunction*. Specifically, it seeks to satisfy the equality constraints given by

$$V_{\pi(i-1)} = U_{\pi(i)} \quad \text{for } i = 2, \dots, P.$$

Example

Consider the following conjunction of relationship literals:

$$\begin{aligned} & \text{AdvisedBy}(\text{Student}, \text{Professor}), \\ & \text{TaughtBy}(\text{Course}, \text{Professor}), \\ & \text{Registered}(\text{Student}, \text{Course}) \end{aligned}$$

This set of relationships is not a chain conjunction because there is no permutation of the literals that satisfies the necessary chain equality constraints. However, we can transform this into a chain conjunction using the reverse relations:

$$\begin{aligned} & \text{AdvisedBy}(\text{Student}, \text{Professor}), \\ & \text{Teaches}(\text{Professor}, \text{Course}), \\ & \text{TakeCourse}(\text{Course}, \text{Student}) \end{aligned}$$

In this case, the literals can be rearranged to form a chain conjunction, satisfying the equality constraints. Here *Teaches* is the reverse of *TaughtBy* and *TakeCourse* is the reverse of *Registered*.

- Our algorithm and proof can be extended 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 is composed of centered chains, so our matrix multiplication algorithm can be used to recursively compute instance counts.

A.10 GNN-BASED GRAPH REALISM METRIC FOR EVALUATING GRAPH GENERATION

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 their real-valued descriptors vectors $\phi(\mathcal{G})$ 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 and is therefore independent of any of the models under evaluation.

We adapt the SOTA 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 $\tilde{\mathcal{G}}_1, \dots, \tilde{\mathcal{G}}_m$. An expected graph $\tilde{\mathcal{G}}_i$ is generated by sampling a node embedding z_i from the prior distribution $p(z)$, then applying the decoder model eq. (5) to z_i . Following Thompson *et al.* (2022), we apply an embedder \mathcal{E} with random weights to the training graph resp. generated expected graphs to obtain embeddings e resp. $\hat{e}_1, \dots, \hat{e}_m$. 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. To quantify the similarity of the generated embeddings $\hat{e}_1, \dots, \hat{e}_m$ to the training graph embedding e , we utilize the MMD metric with a linear kernel, which is recommended by Thompson *et al.* (2022). We used their code to compute the MMD metric results.

A.11 LEARNING CURVES

We report a learning curve experiment to examine the effect of rule knowledge on data efficiency. The idea is to simulate the impact of a domain expert providing the model with a strong set of rules. After learning an informative set of rules on the entire training graph, we sample 20% of node labels as test labels and reserve the other 80% as training node labels. Then we sample $x\%$ of the training node labels for training the VGAE with and without rules. We report the predictive accuracy on the test labels, after training the VGAE with and without rules on $x = 25\%, 50\%, 75\%, 100\%$ of training labels. The models are tested on the remaining $100 - x\%$ of nodes. UW is too small to obtain a meaningful learning curve.

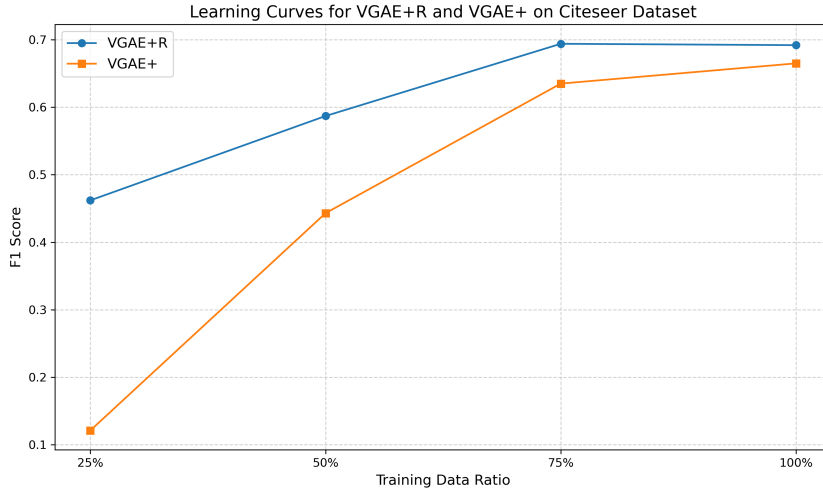


Figure 13: Learning curve for the CiteSeer dataset.

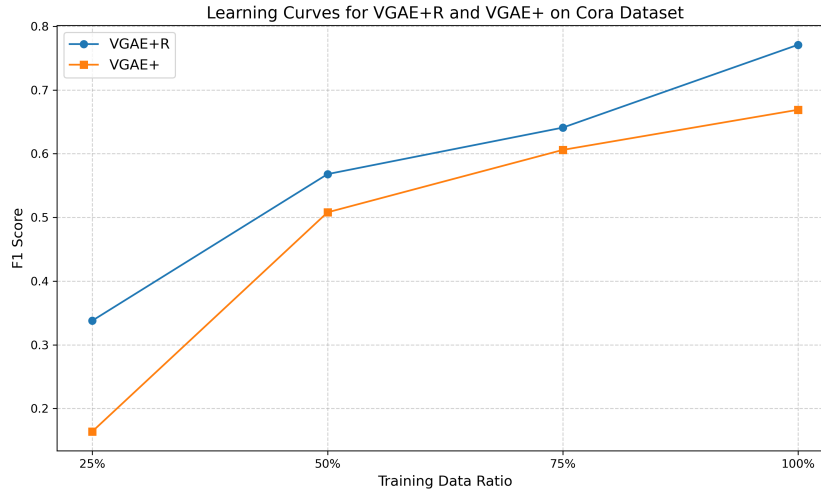


Figure 14: Learning curve for the Cora dataset.

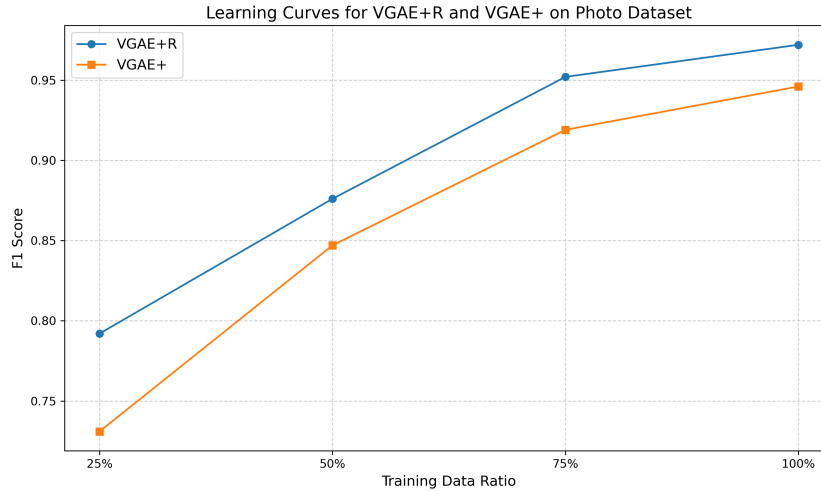


Figure 15: Learning curve for the Photo dataset.

A.12 LOSS CURVES

In this section we show that rule matching has a big impact on training, by in effect initializing the GGM model in a part of weight space that encodes the rule knowledge.

A.12.1 LABEL LOSS CURVES

In this section, we compare the label loss component for each model across different datasets. As shown in Figures 19, 20, 21, 22, and 23, the label loss for VGAE+R is generally lower than that for VGAE+ across most datasets, suggesting that VGAE+R may yield better accuracy in node classification.

A.12.2 TOTAL LOSS CURVES

In this section, we present the total loss curves for both the VGAE+ and VGAE+R models across multiple datasets. While it is not possible to directly compare the loss values between the two models, the downward trend in loss during training for both models indicates successful convergence. Figures

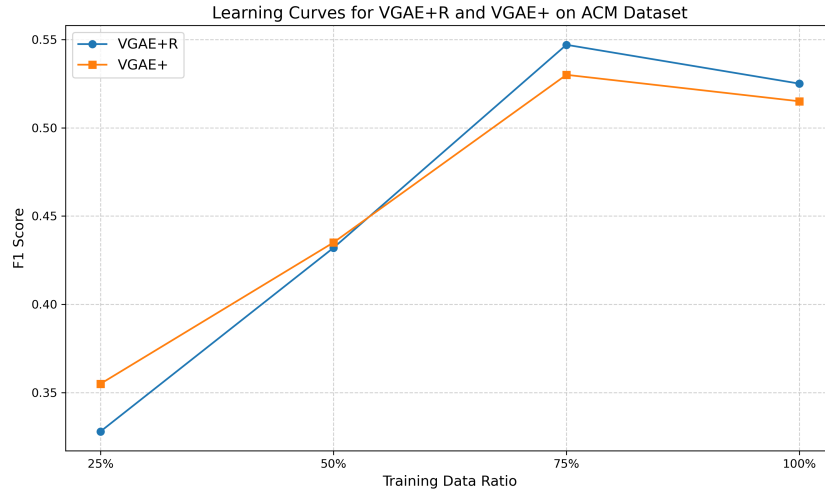


Figure 16: Learning curve for the ACM dataset.

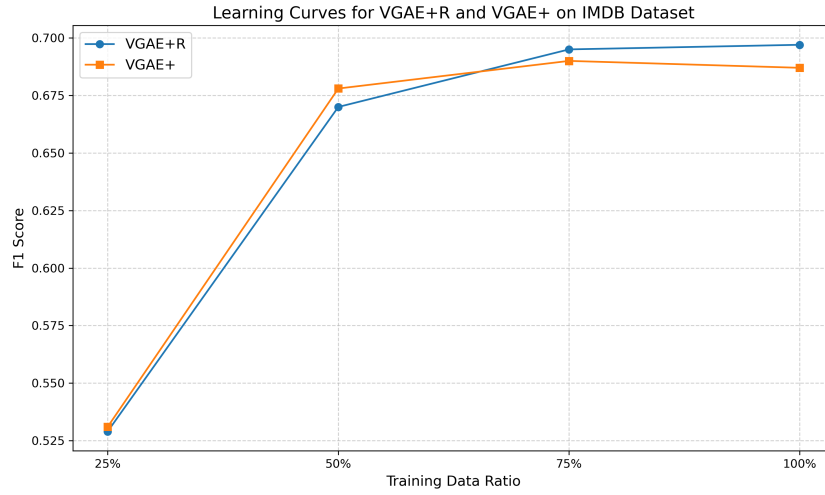


Figure 17: Learning curve for the IMDB dataset.

24, 25, 26, 27, and 28 illustrate the total loss for each dataset, confirming the models' progress throughout the training process.

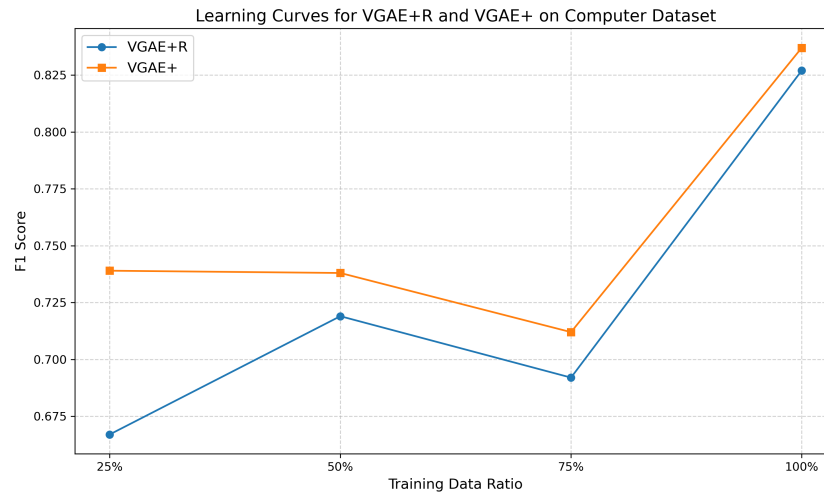


Figure 18: Learning curve for the Computers dataset.

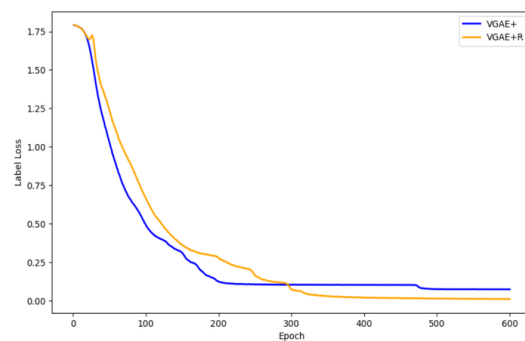


Figure 19: Label loss component for CiteSeer dataset during model training

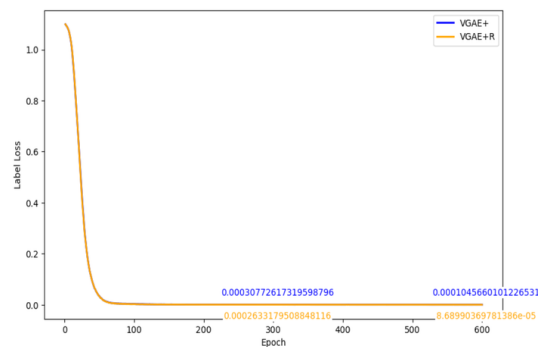


Figure 20: Label Loss component for ACM dataset during model training

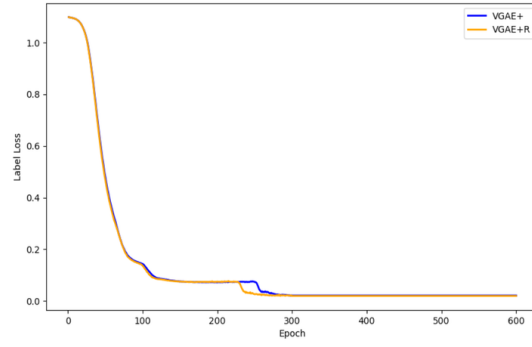


Figure 21: Label Loss component for IMDB dataset during model training

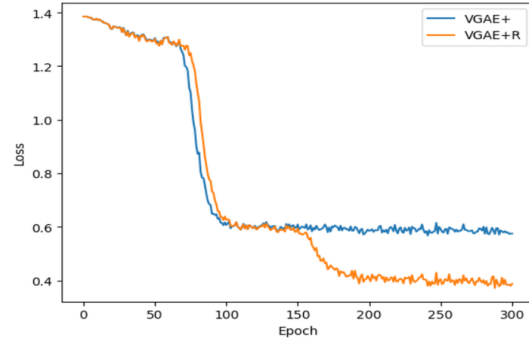


Figure 22: Label Loss component for UW dataset for person node type during model training

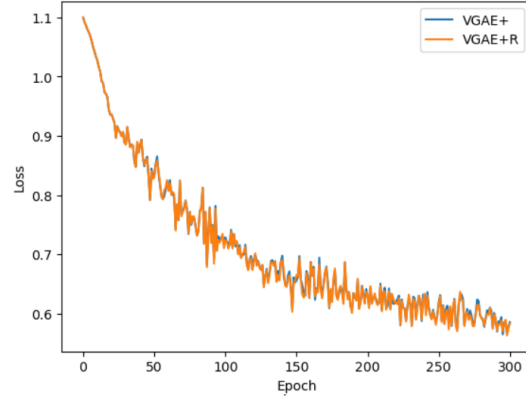


Figure 23: Label Loss component for UW dataset for course node type during model training

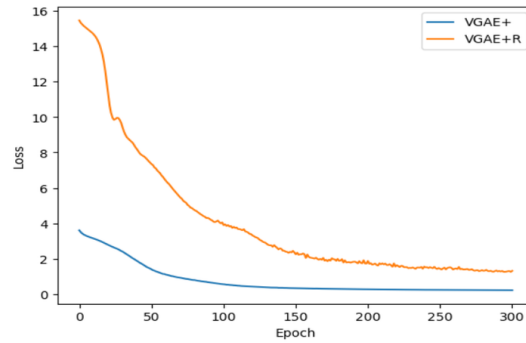


Figure 24: Total loss for Cora dataset during model training

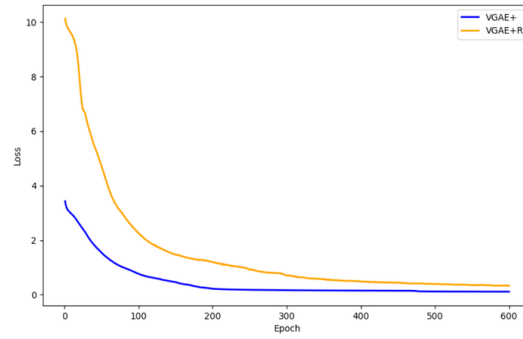


Figure 25: Total Loss for CiteSeer dataset during model training

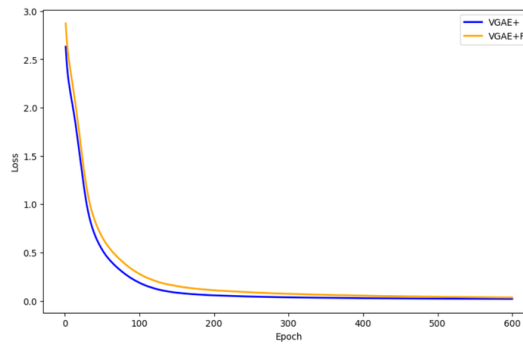


Figure 26: Total Loss for ACM dataset during model training

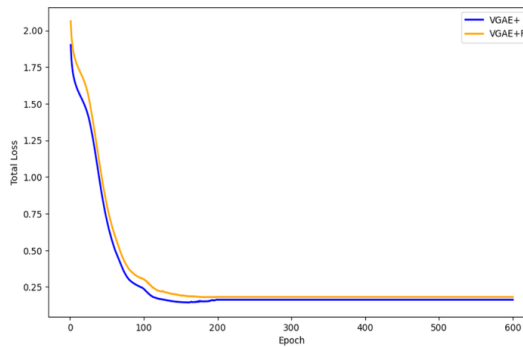


Figure 27: Total Loss for IMDb dataset during model training

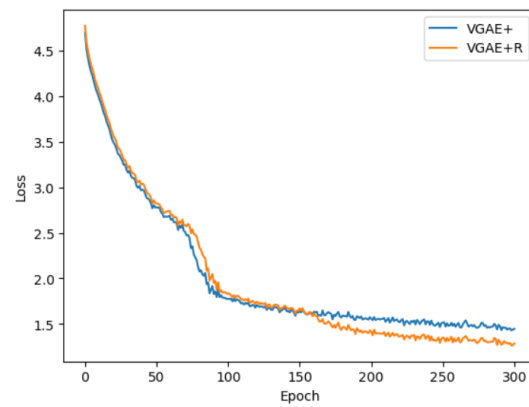


Figure 28: Total Loss for UW dataset during model training

Table 7: Graph quality comparison of VGAE+ and VGAE+R on six datasets using MMD metrics (degree, clustering, 4-orbit, spectral, diameter). Lower values indicate generated graphs are closer to observed ones; VGAE+R shows consistent improvements.

Dataset	Metric	VGAE+	VGAE+R
CiteSeer	Degree MMD	0.752	0.604
	Clustering MMD	0.733	1.672
	4-Orbit MMD	0.549	0.538
	Spectral MMD	0.506	0.431
	Diameter MMD	2.000	1.877
Cora	Degree MMD	0.768	0.707
	Clustering MMD	1.680	1.648
	4-Orbit MMD	1.400	1.375
	Spectral MMD	0.514	0.461
	Diameter MMD	1.979	1.944
IMDB	Degree MMD	0.756	0.521
	Clustering MMD	1.950	1.870
	4-Orbit MMD	0.599	0.523
	Spectral MMD	0.516	0.389
	Diameter MMD	1.894	1.760
ACM	Degree MMD	0.770	0.664
	Clustering MMD	1.930	1.880
	4-Orbit MMD	0.756	0.692
	Spectral MMD	0.571	0.524
	Diameter MMD	1.944	1.955
Photos	Degree MMD	0.727	0.547
	Clustering MMD	1.760	1.810
	4-Orbit MMD	1.090	1.260
	Spectral MMD	0.606	0.399
	Diameter MMD	1.941	1.305
Computers	Degree MMD	0.756	0.227
	Clustering MMD	1.952	1.867
	4-Orbit MMD	0.788	0.767
	Spectral MMD	0.516	0.326
	Diameter MMD	1.894	0.706

A.13 GRAPH REALISM EVALUATION

Our main metric of graph quality is the GNN-based graph realism MMD measure, which is the current state-of-the-art method for comparing generated graphs with observed graphs. This metric was introduced in prior work (Thompson *et al.*, 2022) and relies on a reference GNN R to embed entire graphs. Importantly, R is independent of the models under evaluation. In our experiments, we used a reference GNN with random weights, as recommended by Thompson *et al.* (2022), ensuring that no domain-specific information was encoded. A pre-trained reference GNN could also be used, but we did not adopt that setting in this paper.

For completeness, we also report statistic-based MMD measures (degree, clustering, 4-orbit, spectral, diameter), which have been widely used in earlier work on auto-regressive graph generation such as GraphRNN. These complementary metrics show consistent benefits from incorporating first-order rules.

Table 7 presents the performance of VGAE+ (baseline) and VGAE+R (rule-enhanced) across six benchmark datasets. Across both GNN-based and classical MMD measures, lower values indicate closer alignment between generated and observed graphs, and VGAE+R achieves lower scores in all but one setting (clustering on Citeseer), demonstrating the positive effect of rule-based enhancement.

Table 8: Count Distance between rule values in the test set and generated graphs. Lower values are better.

Dataset	VGAE+	VGAE+R
Cora	1403.530	18.020
CiteSeer	459.560	24.760
IMDB	25762.900	1082.700
ACM	474.360	118.900
Photo	6046.740	1212.110
Computers	29567.100	1006.970
UW	763257.950	744808.390

Table 9: Runtime and memory usage of VGAE+ and VGAE+R across benchmark datasets.

Dataset	#Nodes	#Rules	VRAM (GB)	Runtime VGAE+	Runtime VGAE+R	GPU
Cora	2708	138	12.2	0m 22.9s	0m 44.7s	GTX 1080 Ti
CiteSeer	3327	193	11.7	1m 34.2s	2m 18.9s	Titan X
IMDB	12772	31	38.8	3m 36.7s	4m 08.5s	A40 (48 GB)
ACM	8993	62	41.2	2m 07.5s	2m 34.4s	A40 (48 GB)
Photos	7650	207	80.0	1m 36.1s	2m 40.4s	A100 (80 GB)
Computers	13752	58	79.5	4m 30.0s	5m 24.2s	A100 (80 GB)
UW	410	111	0.175	0m 15.0s	0m 16.0s	GTX 1080 Ti

A.14 COUNT DISTANCE EVALUATION BASED ON PRIOR EMBEDDINGS

For the Count Distance metric, prior work on VAE-based graph generation has often evaluated fit to the training or test data using the ELBO as an indirect measure of generation quality (Simonovsky and Komodakis, 2018). In our setting, Count Distance similarly measures how well generated graphs reproduce rule instance counts observed in the data. One might expect a neural GGM to implicitly match motif counts as part of modeling the training distribution, but our results show that a standard VGAE does not. Adding our explicit semantic loss substantially reduces this discrepancy.

we also report results when sampling directly from the prior $p(z)$. As shown in Table 8, the semantic loss improves not only fit to training data but also pure generation under $p(z)$. Lower values indicate closer alignment between generated and observed rule counts, and VGAE+R consistently achieves much lower distances than VGAE across all datasets.

A.15 SCALABILITY AND RUNTIME ANALYSIS

Our core algorithm relies on dense matrix multiplications over probabilistic adjacency matrices. As discussed in Section 5, this raises natural concerns about scalability. In this appendix, we provide additional details on the computational resources required and the observed runtime overhead of the semantic loss.

Table 9 summarizes, for each benchmark graph, the number of nodes, number of rules, peak GPU memory usage, runtime for 100 training epochs with VGAE+ and VGAE+R, and the GPU model used. Storage needs for generating graphs ranged from 1GB for the smallest dataset to 70GB for the largest, and end-to-end experiments (training + generation + evaluation) took between 4 and 5 hours for the biggest dataset.

Modern GPUs render dense matrix multiplications surprisingly efficient: thousands of cores execute multiply-add operations in parallel, while optimized libraries such as cuBLAS tile large matrices into on-chip shared-memory blocks. Tensor Cores and fused multiply-add instructions further accelerate throughput, often at reduced precision. As a result, the additional multiplications from our semantic loss add only a modest wall-clock overhead.

For example, runtime on the UW graph increases by just 1 second (15s to 16s). For mid-sized graphs such as ACM and IMDB, the slowdown is about 20% and 15%, respectively. Larger graphs such as Photos incur slowdowns of $1.95\times$ and $1.67\times$, yet all runs complete in under six minutes. Even the

Table 10: Robustness of VGAE+R to noisy/incomplete rules on the Cora dataset.

Condition	Degree	Clust.	4-Orbit	Spectral	Diam.	CountDist (train)	CountDist (test)	Graph Realism	AUC (%)	F1 (%)
50% Rules Deleted	0.733	1.640	1.398	0.517	1.914	58991.850	35.190	1.745e20	94.5	85.0
VGAE+R (full rules)	0.707	1.648	1.375	0.461	1.944	26800.000	18.000	6.840e17	96.5	88.0
Baseline VGAE+	0.768	1.680	1.400	0.514	1.979	51400.000	1403.000	1.400e19	86.0	69.0

13k-node Computers graph finishes in 5m 24s on an A100 GPU. On average, VGAE+R adds only a $1.4\times$ overhead while yielding substantial improvements in rule compliance and graph quality.

It is common practice to evaluate new objectives for graph generative models in settings that require a full dense adjacency matrix. Examples include GraphVAE (Simonovsky and Komodakis, 2018) and the more recent Digress diffusion model (Vignac *et al.*, 2023). These evaluations remain meaningful because many graph generation benchmarks consist of small to medium-sized graphs, such as molecules and proteins. A general lesson in the field is that all-at-once methods have fast training and generation for small to medium graphs, while scaling to very large graphs typically requires autoregressive approaches (Hamilton, 2020).

A.16 ROBUSTNESS TO NOISY OR INCOMPLETE RULES

Almost all rules we use are non-deterministic, meaning they allow for exceptions (e.g., “if X works in city Y , then X lives in city Y ,” which may not always hold). This flexibility is part of the power of first-order rules, which can hold in a graph to a degree rather than absolutely (Domingos and Richardson, 2007). If noise refers to uninformative rule bodies, we prune such rules during preprocessing, which increases robustness. If the rule set is incomplete—missing important rules—the system degrades gracefully toward the standard GGM likelihood. In the extreme case of an empty rule set, our objective reduces exactly to the standard VGAE loss. Since our implementation derives rules from a first-order Bayesian network that represents the full joint domain distribution (Schulte and Qian, 2015), the resulting rule sets are likely close to complete.

Table 10 reports results on the Cora dataset under three settings: (1) deleting the top 50% of rules, (2) using the full VGAE+R rule set, and (3) the baseline VGAE+. Metrics include generation quality (MMD measures), count distance on train/test graphs, graph realism ($\times 10^n$), and classification accuracy (AUC, F1). Results confirm that pruning rules degrades performance, but VGAE+R remains robust compared to the baseline VGAE+.