SeeGera: Self-supervised Semi-implicit Graph Variational Auto-encoders with Masking

Xiang Li
East China Normal University
Shanghai, China
xiangli@dase.ecnu.edu.cn

Tiandi Ye
East China Normal University
Shanghai, China
52205903002@stu.ecnu.edu.cn

Caihua Shan
Microsoft Research Asia
Shanghai, China
caihuashan@microsoft.com

Dongsheng Li
Microsoft Research Asia
Shanghai, China
dongsheng.li@microsoft.com

Ming Gao[*]
East China Normal University
Shanghai, China
mgao@dase.ecnu.edu.cn

ABSTRACT

Generative graph self-supervised learning (SSL) aims to learn node representations by reconstructing the input graph data. However, most existing methods focus on unsupervised learning tasks only and very few work has shown its superiority over the state-of-the-art graph contrastive learning (GCL) models, especially on the classification task. While a very recent model has been proposed to bridge the gap, its performance on unsupervised learning tasks is still unknown. In this paper, to comprehensively enhance the performance of generative graph SSL against other GCL models on both unsupervised and supervised learning tasks, we propose the SEEGERA model, which is based on the family of self-supervised variational graph auto-encoder (VGAE). Specifically, SEEGERA adopts the semi-implicit variational inference framework, a hierarchical variational framework, and mainly focuses on feature reconstruction and structure/feature masking. On the one hand, SEEGERA co-embeds both nodes and features in the encoder and reconstructs both links and features in the decoder. Since feature embeddings contain rich semantic information on features, they can be combined with node embeddings to provide fine-grained knowledge for feature reconstruction. On the other hand, SEEGERA adds an additional layer for structure/feature masking to the hierarchical variational framework, which boosts the model generalizability. We conduct extensive experiments comparing SEEGERA with 9 other state-of-the-art competitors. Our results show that SEEGERA can compare favorably against other state-of-the-art GCL methods in a variety of unsupervised and supervised learning tasks.

KEYWORDS

Graph neural networks, graph self-supervised learning, variational graph auto-encoder

[*]Corresponding author

1 INTRODUCTION

Self-supervised learning (SSL) [2, 6, 9, 36] has attracted significant attention recently. By extracting and employing supervisions from data itself, SSL can heavily reduce the dependence of neural network models on the labeled data, which is costly to obtain. To facilitate graph-based learning, SSL has been applied on graph-structured data. For example, it can learn representations for nodes (e.g., web pages in search engines), and detect the anomalies on webs (e.g., malicious users) [16]. Recently, graph contrastive learning (GCL), as one of the main SSL types, has experienced a surge [10, 26, 29, 35]. The core idea of GCL is to first construct positive and negative pairs for nodes, and then maximize the similarity between positive pairs while minimizing that between negative ones[1].

Despite the success, existing GCL methods suffer from two main problems. On the one hand, negative samples are needed in most contrastive objectives, which generally construct one positive and K negative samples for each node. However, these models are easily affected by the value of K. When K is small, the model cannot learn sufficient discriminative information, which degrades the model effectiveness; otherwise, there could lead to a large number of false-negative samples and slow convergence. Generally, K is set empirically and there lack theoretical supports. On the other hand, for the rest of methods based on positive pairs only, they are easily trapped into a degenerate solution [40], where all the output embeddings of nodes collapse to a constant. To tackle the issue, additional strategies are necessary, such as asymmetric dual encoders with momentum updates and exponential moving average [21, 27]. Recently, some studies [14] have showed that although these training strategies can alleviate collapse to some extent, they may still cause collapse in partial dimensions of the representation, which leads to worse performance.

To address the shortcomings of GCL methods, generative graph SSL methods can be used instead. In particular, self-supervised

---

[1]Note that some GCL methods require positive pairs only and they only maximize the similarity between positive pairs.

graph auto-encoders (GAEs) [12], whose objective is to reconstruct the input graph data, have been widely studied. Existing methods mainly differ in their adopted reconstruction components, such as the adjacency matrix reconstruction [19], the node feature reconstruction [20] and a combination of both graph structure and node feature reconstruction [23]. However, most of these methods focus on the unsupervised learning tasks like link prediction and node clustering, and very few work has shown its superiority over the state-of-the-art GCL methods, especially on the classification task. While a masked GAE model GraphMAE [7] is very recently proposed to bridge the gap, its performance on the unsupervised learning tasks is still unexplored. Since the goal of SSL is to learn versatile representations, a further study on self-supervised GAE model that can achieve comprehensive superiority on both unsupervised and supervised learning tasks is needed. Further, although GraphMAE is an auto-encoding method, it is based on GAE and is essentially not a generative model. This also calls our attention back to the study of generative graph SSL model, such as variational graph auto-encoder (VGAE) [12].

Different from GAE, VGAE consists of an inference model and a generative model. Specifically, the inference model encodes observations (links and features) into latent variables (node embeddings) while the generative model decodes from these latent variables to reconstruct links. However, as pointed out in [7], node feature reconstruction is beneficial for learning high-quality representations. Therefore, the lack of feature reconstruction could degrade the model effectiveness. To solve the issue, most existing methods adopt MLP [8, 9] and GNNs [7, 20] as their decoders for feature reconstruction. However, they utilize node-level embeddings only and ignore feature-level embeddings that contain rich semantic information on node features and can be used to help feature reconstruction. Recently, CAN [18] is proposed to co-embed both nodes and features, and use the inner product of their embeddings as the decoder to recover node features. Despite the success, it has three main problems. First, the linear decoder is generally less powerful than MLP and GNNs, which restricts the model's capability in reconstructing node features. Second, it assumes the independence between node and feature embeddings in the variational inference stage, but practically these two types of embeddings are highly correlated. Third, it lacks structure/feature masking in the learning process, which has been shown to degrade the model's performance on the classification task [7].

In this paper, we study generative graph SSL and our goal is to enhance the family of self-supervised VGAE on graph representation learning in a variety of downstream tasks. Recently, semi-implicit variational inference (SIVI) [32], which is a hierarchical variational framework, has been applied to VGAE to model a wide range of underlying true posteriors with multi-modality, skewness and heavy tails [4]. We thus adopt the framework to remove the explicit Gaussian restriction on the variational distribution and mainly focus on the component of feature reconstruction and structure/feature masking. We propose a **Se**lf-supervised s**e**mi-implicit **G**raph variational auto-encod**er** with m**a**sking, namely, SeeGera. Specifically, the model co-embeds both nodes and features in the encoder and jointly reconstructs links and features in the decoder. Note that the feature embeddings can provide fine-grained information that is supplementary to the node embeddings when reconstructing node

features. Specifically, for each node, we take its feature values as weights and compute the weighted average of feature embeddings w.r.t. the node. The weighted embedding characterizes the affinities between the node and all the features. After that, we combine the weighted embedding with the node embedding, and feed the fused embedding into GNNs to reconstruct the node's features. Further, to generate node and feature embeddings in the encoder, we first assume the independence between them and propose the base SeeGera model. Then we upgrade the model by capturing the correlations between node and feature embeddings. Finally, we add an additional layer to the hierarchical variational framework to integrate SeeGera with the masking mechanism and boost the model performance. In summary, our main contributions are listed:

• We propose a generative graph SSL model SeeGera. To our knowledge, this is the first generative graph SSL method that is comprehensively compared with the SOTA GCL models in terms of both unsupervised and supervised learning tasks, and shows superiority.

• We present a novel feature reconstruction method that leverages both node and feature embeddings to provide fine-grained information for reconstructing features. We further introduce the structure/feature masking mechanism by adding an additional layer to the hierarchical variational framework.

• We conduct extensive experiments to evaluate the performance of SeeGera on two unsupervised learning tasks: link prediction and attribute inference, and one supervised learning task: node classification. Experimental results show that SeeGera can significantly outperform other competitors on both link prediction and attribute inference tasks, and perform comparably with them in node classification. This effectively verifies the power of generative graph SSL in graph representation learning.

2 RELATED WORK

In this section, we summarize the related work on both graph self-supervised learning and generative graph self-supervised learning, respectively.

2.1 Graph self-supervised learning

Graph self-supervised learning [7, 26, 31, 35] aims to employ supervisions extracted from graph-structured data without the need for annotated data. Existing methods can be mainly divided into four types: (1) generative models [12], whose objective is to reconstruct the input graph data. (2) auxiliary-property-based methods [36], which first obtain graph-related properties and then take them as supervisions, such as the pseudo labels of unlabeled nodes; (3) contrastive models [29], which construct positive and negative pairs for contrast. (4) hybrid approaches [38], which combine the objectives of the first three types in a multi-task learning fashion. For a comprehensive survey on graph self-supervised learning, see [15].

Recently, graph contrastive learning has been widely studied. According to whether negative samples are used in the learning process, existing methods include negative-sample-based and negative-sample-free ones. For the former, DGI [29] and InfoGraph [26] employ corruptions to construct negative pairs. GRACE [41], GCA [42] and GraphCL [35] take samples in a mini-batch as a dictionary whose size is constrained by the batch size and consider other samples in the same mini-batch as negatives of a sample, while GCC [21]

maintains a dynamic dictionary with larger size as in MoCo [6]. For the latter, BGRL [27] and CCA-SSG [37] are two representative models that are based on asymmetric encoding architectures. However, they require special training strategies to avoid the collapse of learned node embeddings to a constant, such as momentum update [6], exponential moving average [27] and stop gradient [27]. Further, existing GCL methods heavily rely on graph augmentation strategies to construct different graph views for contrast, including feature-oriented (e.g., masking [35] and shuffling [29]), proximity-oriented (e.g., perturbation [35]), and graph-sampling-based (e.g., random-walk [5]) augmentations.

2.2 Generative graph self-supervised learning

Generative graph self-supervised learning aims to take the input graph as self-supervision and recover the input data. It mainly consists of two families of models: graph autoregressive models and graph autoencoders (GAEs). Autoregressive models [33, 34] decompose joint probability distributions as a product of conditionals. The representative graph autoregressive model is GPT-GNN [9], which takes attributed graph generation as its objective. However, since autoregressive models require an explicit ordering to generate, they might not work well on graphs that do not exhibit inherent orders.

Different from graph autoregressive models, GAEs do not require any decoding ordering and they aim to reconstruct part of the input graph data. According to the reconstructed components, existing self-supervised GAE methods include those that reconstruct links only (e.g., ARVGA [19], GAE [12], VGAE [12]), features only (e.g., GraphMAE [7], GALA [20], MGAE [30], EP [3]), and a combination of both links and features (e.g., GATE [23], CAN [18], DGE [39]). However, most of these methods focus on the link prediction and node clustering tasks, and few of them compares favorably against the state-of-the-art GCL methods, especially in the classification task. While GraphMAE is recently proposed to bridge the gap, its performance on unsupervised learning tasks remains unexplored. Further, it is based on GAE and is essentially not a generative model. Different from GAE, variational graph auto-encoder (VGAE) is a generative model that recovers links only in the decoder. While there exist some self-supervised VGAE models that reconstruct features [23, 39], most of them only leverage node-level embeddings but ignore feature-level embeddings that contain fine-grained information for node features and can help boost feature reconstruction. In this paper, we reconsider generative graph self-supervised learning and show that self-supervised VGAE can outperform or perform comparably against other SOTA GCL models in a variety of tasks, such as link prediction, attribute inference and node classification.

3 PRELIMINARY

3.1 Notations

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph, where $\mathcal{V} = \{x_i\}_{i=1}^n$ is a set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges. Let A be the adjacency matrix of G, such that A_{ij} represents the weight of edge e_{ij} between objects x_i and x_j. For simplicity, we set $A_{ij} = 1$ if $e_{ij} \in \mathcal{E}$; 0, otherwise. Further, since nodes in a graph are usually associated with features, we denote $\mathcal{F} = \{f_r\}_{r=1}^l$ as a set of node features and $X \in \mathbb{R}^{n \times l}$ as the node feature matrix, where the i-th row X_i is the feature vector of node x_i. For the node representation matrix, let it be $Z^{\mathcal{V}} \in \mathbb{R}^{n \times d}$,

where d is the output embedding dimension satisfying $d \ll |\mathcal{V}|$. Note that the i-th row $Z_i^{\mathcal{V}}$ represents the embedding of node x_i. Similarly, $Z^{\mathcal{F}} \in \mathbb{R}^{l \times d}$ denotes the feature representation matrix, whose r-th row $Z_r^{\mathcal{F}}$ is the embedding of node feature f_r. In this paper, we learn both node and feature representations, and use node representations in various downstream tasks.

3.2 SIVI and SIG-VAE

Given observations Y and latent variable Z, the vanilla variational inference (VI) derives an evidence lower bound

$$\text{ELBO} = -\mathbb{E}_{Z \sim q(Z|\psi)} \left[\log q(Z|\psi) - \log p(Y, Z) \right], \quad (1)$$

where ψ is variational parameter, $q(Z|\psi)$ is variational distribution and $p(Y, Z)$ is joint distribution. However, VI restricts an exponential family assumption to the posterior. To address the problem, semi-implicit variational inference (SIVI) [32] considers variational parameters as random variables drawn from a mixing distribution. Specifically, the semi-implicit variational distribution for Z is defined in a hierarchical manner, which follows $Z \sim q(Z|\psi)$ and $\psi \sim q_\phi(\psi)$. Here, ϕ is the parameter of the mixing distribution $q_\phi(\psi)$. Further, ψ can be marginalized out to derive a distribution family $\mathcal{H}$ indexed by ϕ for Z:

$$\mathcal{H} = \left\{ h(Z) : h(Z) = \int_\psi q(Z|\psi) q_\phi(\psi) d\psi \right\}. \quad (2)$$

Note that $q(Z|\psi)$ is required to be explicit, but the mixing distribution $q_\phi(\psi)$ is allowed to be implicit. Moreover, the marginal distribution $h(Z) \in \mathcal{H}$ is often implicit unless $q_\phi(\psi)$ is conjugate to $q(Z|\psi)$. These are the reasons why the method is referred to as "semi-implicit" VI. To maintain simple optimization, $q(Z|\psi)$ is required to be either reparameterizable [11] or allow the ELBO under $q(Z|\psi)$ to be analytic. For $q_\phi(\psi)$, it needs to be reparameterizable. Generally, SIVI draws from $q_\phi(\psi)$ by injecting random noise ϵ into node features and transforming the features via neural networks.

Recently, Hasanzadeh et al. [4] apply SIVI to VGAE and propose the semi-implicit graph variational auto-encoder (SIG-VAE) model. Specifically, it sets $q(Z|\psi)$ to be Gaussian distribution and uses GNNs to characterize the mixing distribution $q_\phi(\psi)$. While SIG-VAE uses the hierarchical variational framework to capture complex non-Gaussian posteriors, it still has the problem of ignorance of feature reconstruction and structure/feature masking. Therefore, based on the framework of SIG-VAE, we next explore how to enhance self-supervised VGAE for unsupervised graph representation learning.

4 ALGORITHM

In this section, we present our model SeeGera. Different from SIG-VAE that uses node embeddings only, SeeGera further generates feature embeddings to capture the rich semantic information on node features, which can be used to enhance feature reconstruction. Specifically, we consider two cases in the encoder when generating node and feature embeddings: (1) they are independent; (2) they are correlated. After that, in the decoder part, we utilize GNNs to reconstruct node features based on both node and feature embeddings. Finally, we show how structure/feature masking can be integrated with the hierarchical variational framework and gives

the optimization techniques. The overall framework of SeeGera is summarized in Figure 1.

4.1 Variational lower bound

In VI, given a graph $\mathcal{G}$ with an adjacency matrix A and a feature matrix X, we approximate the true posterior $p(Z^{\mathcal{V}}, Z^{\mathcal{F}}|A, X)$ with a variational distribution $q(Z^{\mathcal{V}}, Z^{\mathcal{F}}|\psi_1, \psi_2)$, where ψ_1 and ψ_2 are variational parameters. To capture more complex posteriors that go beyond the exponential family, we adopt the hierarchical variational framework in SIVI and assume

$$Z^{\mathcal{V}} \sim q_1(Z^{\mathcal{V}}|\psi_1), \ \psi_1 \sim q_{\phi_1}(\psi_1), \ Z^{\mathcal{F}} \sim q_2(Z^{\mathcal{F}}|\psi_2), \ \psi_2 \sim q_{\phi_2}(\psi_2), \quad (3)$$

where ϕ_1 and ϕ_2 are parameters of mixing distributions. We marginalize ψ_1 and ψ_2 out and derive

$$Z^{\mathcal{V}} \sim h_{\phi_1}(Z^{\mathcal{V}}) = \int_{\psi_1} q_1(Z^{\mathcal{V}}|\psi_1)q_{\phi_1}(\psi_1)\mathrm{d}\psi_1,$$
$$Z^{\mathcal{F}} \sim h_{\phi_2}(Z^{\mathcal{F}}) = \int_{\psi_2} q_2(Z^{\mathcal{F}}|\psi_2)q_{\phi_2}(\psi_2)\mathrm{d}\psi_2. \quad (4)$$

We maximize the log-likelihood of observations A and X, and use Jensen's inequality to get

$$\log p(A, X) \geq \mathbb{E}_{h_\phi(Z^{\mathcal{V}}, Z^{\mathcal{F}})}\left[\log \frac{p(A, X, Z^{\mathcal{V}}, Z^{\mathcal{F}})}{h_\phi(Z^{\mathcal{V}}, Z^{\mathcal{F}})}\right] = \mathcal{L}, \quad (5)$$

where $\mathcal{L}$ is ELBO and

$$h_\phi(Z^{\mathcal{V}}, Z^{\mathcal{F}}) = \int_{\psi_1}\int_{\psi_2} q(Z^{\mathcal{V}}, Z^{\mathcal{F}}|\psi_1, \psi_2)q_\phi(\psi_1, \psi_2)\mathrm{d}\psi_1\mathrm{d}\psi_2 \quad (6)$$

is the marginal distribution over $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$. Since h_ϕ is often intractable, the Monte Carlo estimation of ELBO could be prohibited. To address the problem, we first take the mean-field assumption:

$$q(Z^{\mathcal{V}}, Z^{\mathcal{F}}|\psi_1, \psi_2) = q_1(Z^{\mathcal{V}}|\psi_1)q_2(Z^{\mathcal{F}}|\psi_2),$$
$$q_\phi(\psi_1, \psi_2) = q_{\phi_1}(\psi_1)q_{\phi_2}(\psi_2), \quad (7)$$

and substitute Eq. 7 into Eq. 6 to get:

$$h_\phi(Z^{\mathcal{V}}, Z^{\mathcal{F}}) = h_{\phi_1}(Z^{\mathcal{V}})h_{\phi_2}(Z^{\mathcal{F}}). \quad (8)$$

From Eq. 8, we see that $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ are independent. Then we derive a lower bound for the ELBO based on Eq. 8:

$$\mathcal{L} = \mathbb{E}_{h_{\phi_1}(Z^{\mathcal{V}})}\mathbb{E}_{h_{\phi_2}(Z^{\mathcal{F}})}\left[\log \frac{p(A, X, Z^{\mathcal{V}}, Z^{\mathcal{F}})}{h_{\phi_1}(Z^{\mathcal{V}})h_{\phi_2}(Z^{\mathcal{F}})}\right]$$
$$\geq \mathbb{E}_{\psi_1 \sim q_{\phi_1}(\psi)}\mathbb{E}_{Z^{\mathcal{V}} \sim q_1(Z^{\mathcal{V}}|\psi_1)}\mathbb{E}_{\psi_2 \sim q_{\phi_2}(\psi)}\mathbb{E}_{Z^{\mathcal{F}} \sim q_2(Z^{\mathcal{F}}|\psi_2)} \quad (9)$$
$$\left[\log \frac{p(A, X, Z^{\mathcal{V}}, Z^{\mathcal{F}})}{q_1(Z^{\mathcal{V}}|\psi_1)q_2(Z^{\mathcal{F}}|\psi_2)}\right] = \underline{\mathcal{L}}_1$$

Details on the derivation of Equation 9 are deferred to Appendix E. In $\underline{\mathcal{L}}_1$, q_1 and q_2 are required to be explicit and have analytic density function, while q_{ϕ_1} and q_{ϕ_2} could be implicit but have to be convenient to be sampled from. Directly optimizing $\underline{\mathcal{L}}_1$ by Monte Carlo Estimation is much easier.

However, in practice, nodes and their features are highly correlated. On the one hand, node embeddings are generated based on features. On the other hand, the semantic information of features are directly reflected by nodes. Therefore, the independence

between $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ in Equation 8 is inappropriate. To tackle the issue, we modify Eq. 7 into:

$$q(Z^{\mathcal{V}}, Z^{\mathcal{F}}|\psi_1, \psi_2) = q_1(Z^{\mathcal{V}}|\psi_1)q_2(Z^{\mathcal{F}}|\psi_2),$$
$$q_\phi(\psi_1, \psi_2) = q_{\phi_2}(\psi_2|\psi_1)q_{\phi_1}(\psi_1), \quad (10)$$

which explicitly characterizes the dependence between variational parameters ψ_1 and ψ_2. In this way, $h_\phi(Z^{\mathcal{V}}, Z^{\mathcal{F}}) \neq h_{\phi_1}(Z^{\mathcal{V}})h_{\phi_2}(Z^{\mathcal{F}})$, which shows that $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ are correlated. Then we can derive another lower bound for the ELBO in Equation 5:

$$\mathcal{L} \geq \mathbb{E}_{\psi_1 \sim q_{\phi_1}(\psi_1)}\mathbb{E}_{\psi_2 \sim q_{\phi_2}(\psi_2|\psi_1)}\mathbb{E}_{(Z^{\mathcal{V}}, Z^{\mathcal{F}}) \sim q(Z^{\mathcal{V}}, Z^{\mathcal{F}}|\psi_1, \psi_2)}$$
$$\left[\log \frac{p(A, X, Z^{\mathcal{V}}, Z^{\mathcal{F}})}{q(Z^{\mathcal{V}}, Z^{\mathcal{F}}|\psi_1, \psi_2)}\right] = \underline{\mathcal{L}}_2 \quad (11)$$

4.2 Encoder

In the encoder, we generate $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ from observations A and X. We next show how to generate $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ according to whether they are independent or not.

[$Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ are independent]. To generate $Z^{\mathcal{V}}$, we assume that $q_1(Z^{\mathcal{V}}|\psi_1) = \prod_{i=1}^n q_1(Z_i^{\mathcal{V}}|\mu_i^{\mathcal{V}}, \Sigma_i^{\mathcal{V}})$, where $q_1(Z_i^{\mathcal{V}}|\mu_i^{\mathcal{V}}, \Sigma_i^{\mathcal{V}}) = \mathcal{N}(\mu_i^{\mathcal{V}}, \Sigma_i^{\mathcal{V}})$ and $\mathcal{N}$ is multivariate Gaussian distribution with mean $\mu_i^{\mathcal{V}}$ and diagonal co-variance matrix $\Sigma_i^{\mathcal{V}}$. Since $\mu_i^{\mathcal{V}}$ and $\Sigma_i^{\mathcal{V}}$ are random variables, we draw them by injecting noise $\tilde{\epsilon}$ into a GNN model:

$$\tilde{X} = \text{CONCAT}(X, \tilde{\epsilon}), \ \tilde{\epsilon} \sim \tilde{q}(\epsilon), \ [\mu_i^{\mathcal{V}}, \Sigma_i^{\mathcal{V}}] = \text{GNN}_1(A, \tilde{X}), \quad (12)$$

where $\text{CONCAT}(\cdot)$ is the concatenation function and $\text{GNN}_1(\cdot)$ is a GNN model. Note that $\tilde{\epsilon}$ is random noise sampled from distribution $\tilde{q}(\epsilon)$, whose row size should be the same as X. The injected noise $\tilde{\epsilon}$ enables the uncertainty propagation between neighboring nodes in the GNN layer, which drives the outputs of the GNN to be random variables rather than deterministic values. Similarly, for $Z^{\mathcal{F}}$, we assume $q_2(Z^{\mathcal{F}}|\psi_2) = \prod_{r=1}^l q_2(Z_r^{\mathcal{F}}|\mu_r^{\mathcal{F}}, \Sigma_r^{\mathcal{F}})$ with $q_2(Z_r^{\mathcal{F}}|\mu_r^{\mathcal{F}}, \Sigma_r^{\mathcal{F}}) = \mathcal{N}(\mu_r^{\mathcal{F}}, \Sigma_r^{\mathcal{F}})$. To infer $\mu_r^{\mathcal{F}}$ and $\Sigma_r^{\mathcal{F}}$, we use a MLP model:

$$\hat{X} = \text{CONCAT}(X^T, \hat{\epsilon}), \ \hat{\epsilon} \sim \hat{q}(\epsilon), \ [\mu_r^{\mathcal{F}}, \Sigma_r^{\mathcal{F}}] = \text{MLP}_1(\hat{X}). \quad (13)$$

Note that $X^T \in \mathbb{R}^{l \times n}$ and the r-th row $X_r^T \in \mathbb{R}^n$ can be considered as the feature vector of feature f_r. The random noise $\hat{\epsilon}$ drawn from $\hat{q}(\epsilon)$ injects uncertainty to the matrix X^T, which models $\mu_r^{\mathcal{F}}$ and $\Sigma_r^{\mathcal{F}}$ as random variables.

[$Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ are correlated]. We also assume q_1 and q_2 follow Gaussian distribution and use the same method as in Equation 12 to generate $[\mu_i^{\mathcal{V}}, \Sigma_i^{\mathcal{V}}]$. However, to capture the dependence between ψ_1 and ψ_2 [2] in Equation 10, we compute $[\mu_r^{\mathcal{F}}, \Sigma_r^{\mathcal{F}}]$ for feature f_r based on node embeddings. Specifically, since the rich semantic information of each feature is directly reflected by values of nodes in the feature, we take the feature vector $X_r^T \in \mathbb{R}^n$ as the weight vector over all the nodes, and compute:

$$[\mu_r^{\mathcal{F}}, \Sigma_r^{\mathcal{F}}] = \text{MLP}_2\left(\frac{\sum_{i=1}^n X_{ri}^T[\mu_i^{\mathcal{V}}, \Sigma_i^{\mathcal{V}}]}{\sum_{i=1}^n X_{ri}^T}\right). \quad (14)$$

---

[2]Here, we denote $\psi_1 = [\mu^{\mathcal{V}}, \Sigma^{\mathcal{V}}]$ and $\psi_2 = [\mu^{\mathcal{F}}, \Sigma^{\mathcal{F}}]$, respectively.
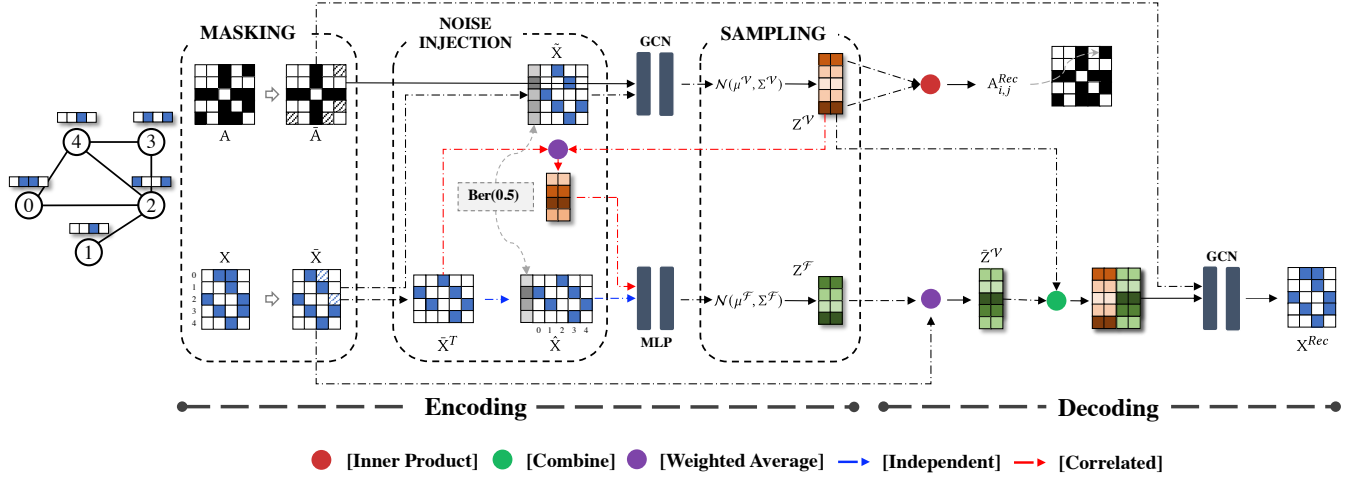
Figure 1: The overall framework of SeeGera.

In this way, $[\mu_r^{\mathcal{F}}, \Sigma_r^{\mathcal{F}}]$ is derived from node embeddings. Since $\mu_i^{\mathcal{V}}$ and $\Sigma_i^{\mathcal{V}}$ are random variables, $\mu_r^{\mathcal{F}}$ and $\Sigma_r^{\mathcal{F}}$ will also be random variables.

4.3 Decoder

In the decoder, we aim to reconstruct both edges and features in the given graph. The generative process is described as follows.

First, for each node x_i and each feature f_r, we draw $(Z_i^{\mathcal{V}}, Z_r^{\mathcal{F}}) \sim h_\phi(Z_i^{\mathcal{V}}, Z_r^{\mathcal{F}})^3$. Second, for each edge A_{ij} in the adjacency matrix A, draw $A_{ij} \sim \text{Ber}(p_{ij}^{A})$. Here, $\text{Ber}(\cdot)$ denotes Bernoulli distribution and p_{ij}^{A} is the probability for the existence of edge A_{ij}. We implement p_{ij}^{A} simply by inner product as: $p_{ij}^{A} = \sigma((Z_i^{\mathcal{V}})^T Z_j^{\mathcal{V}})$, where σ is the sigmoid function. Third, for each attribute X_{ir} in the attribute matrix X, draw $X_{ir} \sim \mathcal{N}(\mu_{ir}^{X}, \Sigma_{ir}^{X} | Z_i^{\mathcal{V}}, Z_r^{\mathcal{F}})$. Here, $\mu_{ir}^{X}, \Sigma_{ir}^{X}$ are functions of $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$.

We next introduce how to compute μ_{ir}^{X} and Σ_{ir}^{X}. Since $Z_r^{\mathcal{F}}$ contains rich semantic information on feature f_r, the affinity between x_i and f_r can provide fine-grained knowledge for feature reconstruction. Given a node x_i, to capture its affinities with all the features, the attention mechanism [28] can be applied on node and feature embeddings. However, this will increase the time complexity of the model. For simplicity, we directly use the feature vector $X_i \in \mathbb{R}^l$ of x_i as the weight vector and calculate the weighted average over all the feature embeddings:

$$\bar{Z}_i^{\mathcal{V}} = \frac{\sum_{r=1}^l X_{ir} Z_r^{\mathcal{F}}}{\sum_{r=1}^l X_{ir}}. \tag{15}$$

Compared with $Z_i^{\mathcal{V}}$, $\bar{Z}_i^{\mathcal{V}}$ contains more details on how each feature can be reconstructed. After that, we combine $Z_i^{\mathcal{V}}$ and $\bar{Z}_i^{\mathcal{V}}$ to get:

$$Z_i^{\mathcal{V}} = \text{COMBINE}(Z_i^{\mathcal{V}}, \bar{Z}_i^{\mathcal{V}}). \tag{16}$$

---

^{3}When $Z_i^{\mathcal{V}}$ and $Z_r^{\mathcal{F}}$ are independent, we draw $Z_i^{\mathcal{V}} \sim h_{\phi_1}(Z_i^{\mathcal{V}})$ and $Z_r^{\mathcal{F}} \sim h_{\phi_2}(Z_r^{\mathcal{F}})$.

In our experiments, we set the COMBINE function to be CONCAT. Finally, the updated $Z_i^{\mathcal{V}}$ is taken as input and fed into a GNN model to learn parameters w.r.t. node x_i:

$$[\mu_i^{X}, \Sigma_i^{X}] = \text{GNN}_2(A, Z_i^{\mathcal{V}}), \tag{17}$$

where $\text{GNN}_2(\cdot)$ is a GNN model.

4.4 Masking

To further improve the model generalizability, we introduce the masking mechanism in SeeGera by adding an additional layer to the hierarchical variational framework. Specifically, we transform Equation 6 into:

$$h_\phi(Z^{\mathcal{V}}, Z^{\mathcal{F}}) = \int_{\psi_1} \int_{\psi_2} \int_{\tilde{G}} q(Z^{\mathcal{V}}, Z^{\mathcal{F}} | \psi_1, \psi_2) \cdot \\ q_\phi(\psi_1, \psi_2 | \tilde{G}) p(\tilde{G} | A, X) d\psi_1 d\psi_2 d\tilde{G} \tag{18}$$

From the above equation, we see that in addition to ψ_1 and ψ_2, the integration is performed over a new variable $\tilde{G}$ and a probability function $p(\tilde{G}|A, X)$. Here, $\tilde{G}$ denotes a new graph and p is the graph augmentation probability function. The equation can lead to a new variational lower bound for the ELBO, but it is more difficult to optimize compared with $\underline{\mathcal{L}}_1$ and $\underline{\mathcal{L}}_2$. To tackle the issue, we can first perform graph augmentation and generate a perturbed graph $\tilde{G}$. After that, based on $\tilde{G}$, node and feature embeddings are learned based on $\underline{\mathcal{L}}_1$ or $\underline{\mathcal{L}}_2$. We repeat the above process until convergence. Although graph augmentation can include more operations than masking, we mainly focus on structure/feature masking in this paper, because masking is beneficial for node classification [7].

4.5 Optimization

In Section 4.1, we have derived two lower bounds $\underline{\mathcal{L}}_1$ and $\underline{\mathcal{L}}_2$ for the ELBO, according to whether $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ are independent. For notation brevity, we use $\underline{\mathcal{L}}$ to overload both $\underline{\mathcal{L}}_1$ and $\underline{\mathcal{L}}_2$. However, directly optimizing $\underline{\mathcal{L}}$ could lead to the degeneracy problem [32] that $q_{\phi_1}(\psi_1)$, $q_{\phi_2}(\psi_2)$ and $q_\phi(\psi_1, \psi_2)$ might converge to a point

mass density, which degenerates SIVI to the vanilla VI. To address the problem, we can regularize $\underline{\mathcal{L}}$ by B_K:

$$B_K = \mathbb{E}_{(\psi_1,\psi_2),\{(\tilde{\psi}_1^k,\tilde{\psi}_2^k)\}_{k=1}^K \sim q_\phi(\psi_1,\psi_2)} D_{\mathrm{KL}} \left(q(Z^{\mathcal{V}},Z^{\mathcal{F}}|\psi_1,\psi_2) || \tilde{h}_K(Z^{\mathcal{V}},Z^{\mathcal{F}}) \right)$$

$$\tilde{h}_K(Z^{\mathcal{V}},Z^{\mathcal{F}}) = \frac{q(Z^{\mathcal{V}},Z^{\mathcal{F}}|\psi_1,\psi_2) + \sum_{k=1}^K q(Z^{\mathcal{V}},Z^{\mathcal{F}}|\tilde{\psi}_1^k,\tilde{\psi}_2^k)}{K+1}.$$

Note that B_K satisfies (1) $B_K \geq 0$; (2) $B_K = 0$ if and only if $K = 0$ or q_ϕ degenerates to a point mass density. According to [32], $\underline{\mathcal{L}}_K = \underline{\mathcal{L}} + B_K$ is an asymptotically exact surrogate ELBO that satisfies $\underline{\mathcal{L}}_0 = \underline{\mathcal{L}}$ and $\lim_{K \to \infty} \underline{\mathcal{L}}_K = \mathcal{L}$. Maximizing $\underline{\mathcal{L}}_K$ with $K \geq 1$ derives positive B_K and could drive q_ϕ away from degeneracy. Moreover, importance reweighting [1] can be further introduced to tighten $\underline{\mathcal{L}}_K$ by drawing J samples $\{(Z^{\mathcal{V}})_j, (Z^{\mathcal{F}})_j, \psi_1^j, \psi_2^j\}_{j=1}^J$ from $q(Z^{\mathcal{V}}, Z^{\mathcal{F}}, \psi_1, \psi_2)$. The objective can be formulated as

$$\underline{\mathcal{L}}_K^J = \mathbb{E}_{\{(Z^{\mathcal{V}})_j,(Z^{\mathcal{F}})_j,\psi_1^j,\psi_2^j\}_{j=1}^J \sim q(Z^{\mathcal{V}},Z^{\mathcal{F}}|\psi_1,\psi_2)q_\phi(\psi_1,\psi_2)}$$

$$\mathbb{E}_{\{\tilde{\psi}_1^k,\tilde{\psi}_2^k\}_{k=1}^K \sim q_\phi(\psi_1,\psi_2)} \log \frac{1}{J} \sum_{j=1}^J \frac{p(A,X,(Z^{\mathcal{V}})_j,(Z^{\mathcal{F}})_j)}{\Omega_j}, \quad (19)$$

where

$$\Omega_j = \frac{1}{K+1} \left[q((Z^{\mathcal{V}})_j,(Z^{\mathcal{F}})_j|\psi_1^j,\psi_2^j) + \sum_{k=1}^K q((Z^{\mathcal{V}})_j,(Z^{\mathcal{F}})_j|\tilde{\psi}_1^k,\tilde{\psi}_2^k) \right].$$

Then we take $\underline{\mathcal{L}}_K^J$ as the surrogate ELBO and use stochastic gradient ascent to optimize it. Note that $\underline{\mathcal{L}}_1$ and $\underline{\mathcal{L}}_2$ treats differently for $q_\phi(\psi_1,\psi_2)$. Finally, we summarize the pseudocodes of SeeGera in Algorithm 1 (see Appendix B).

[**Complexity analysis**] The major time complexity in the encoder comes from GNN and MLP. Suppose we use GCN as the GNN model. Since the adjacency matrix is generally sparse, let d_A be the average number of non-zero entries in each row of the adjacency matrix. Let l be the number of features and d be the embedding dimension. Further, we denote $\tilde{d}$ and $\hat{d}$ as the dimensions of injected noise to the GCN and MLP, respectively. Then, the time complexities for GCN and MLP are $O(nd_A(l+\tilde{d})+n(l+\tilde{d})d)$ and $O(l(n+\hat{d})d)$, respectively. In the decoder, suppose we still adopt GCN as the GNN model. Then the time complexities for reconstructing links and features are $O(n^2 d)$ and $O(nd_A d + ndl)$, respectively. As suggested by [12], we can down-sample the number of nonexistent edges in the graph to reduce the time complexity for recovering links.

5 EXPERIMENT

In this section we comprehensively evaluate the quality of node embeddings learned by SeeGera. We mainly study four research questions:

(RQ1) How does SeeGera perform in the link prediction task?

(RQ2) Can SeeGera effectively predict node attributes?

(RQ3) While SeeGera is an unsupervised learning method, can it perform well when generalized to the node classification task?

(RQ4) How does structure/feature masking influence the performance of SeeGera?

5.1 Datasets and Baselines

To answer the above four questions, we conduct extensive experiments on seven public datasets: *Cora*, *Citeseer*, *Pubmed*, *Coauthor CS*, *Coauthor Physics*, *Amazon Computer* and *Amazon Photo*. Detailed descriptions and statistics on these datasets are provided in Appendix A. We also compare SeeGera with 9 other SOTA baselines, which can be categorized into two groups:

•**[Generative graph SSL methods]**. This group of methods are based on GAE/VGAE and aim to reconstruct links and/or features, including SIG-VAE [4], CAN [18], GATE [23] and GraphMAE [7]. Note that GraphMAE is the SOTA generative graph SSL model.

•**[Graph contrastive learning methods]**. Models in this type construct positive (and negative) pairs for contrast to learn node representations, including DGI [29], MVGRL [5], GRACE [41], GCA [42] and CCA-SSG [37].

Further, for more implementation details, see Appendix D.

5.2 Link Prediction (RQ1)

Link prediction is a typical unsupervised learning task for graph analysis, which aims to predict whether an edge exists between two nodes or not. We compare SeeGera with 8 other SOTA baselines, including GCL models: DGI [29], MVGRL [5], GRACE [41], GCA [42], CCA-SSG [37], and the generative graph SSL methods: CAN [18], SIG-VAE [4], GraphMAE [7]. For our proposed method SeeGera, we put forward three versions. Specifically, SeeGera-v1 assumes the independence between $Z^{\mathcal{V}}$ and $Z^{\mathcal{F}}$ and optimizes $\underline{\mathcal{L}}_1$, while SeeGera-v2 captures the correlations between them and optimizes $\underline{\mathcal{L}}_2$. Further, SeeGera-v3 upgrades SeeGera-v2 by adding the masking mechanism.

To evaluate the model performance, we construct the validation/test set by randomly selecting 20%/10% edges in the original graph as positive samples and an equal number of nonexistent edges as negative samples. After the removal of these selected edges, we train all the models on the resulting graph with the remaining 70% edges. We use two commonly used metrics, the area under the ROC curve (AUC) and the average precision (AP), to report the model performance. For both metrics, a larger value indicates a better performance. We use the validation set for hyper-parameter tuning and early stopping with a patience of 100, i.e., we stop training if both metric scores on the validation set do not increase for 100 consecutive epochs. Similar as in [12], the predicted probability of an edge between nodes x_i and x_j is calculated by $A_{ij} \sim \mathrm{Ber}(p_{ij}^A)$, where $p_{ij}^A = \sigma((Z_i^{\mathcal{V}})^T Z_j^{\mathcal{V}})$ and σ is the sigmoid function. For each method, we run experiments 10 times and report the average results on the test set. Table 1 summarizes the results across all the datasets. From the table, we have the following observations:

(1) The generative graph SSL methods except GraphMAE generally perform better than GCL methods. This is because these methods learn to reconstruct links in the objective. For GraphMAE, it only reconstructs features, which explains its poor performance.

(2) While SeeGera is based on SIG-VAE, it achieves better performance. This demonstrates the importance of feature reconstruction and structure/feature masking.

(3) Although CAN co-embeds both nodes and features, it still performs not very well, due to the independence assumption between node and feature embeddings. Further, it uses linear decoder for feature reconstruction, which restricts the model's effectiveness.

(4) SeeGera significantly outperforms other competitors across all the datasets, which indicates the superiority of generative VGAE

Table 1: Link prediction results. The error bar (±) denotes the standard deviation score of results over 10 trials. We highlight the best score on each dataset in bold. For CAN, the released codes by the authors do not implement reconstruction for numerical features, so we cannot run it on datasets with numerical features. OOM denotes the out-of-the-memory error.

Metrics	Method	Cora	Citeseer	Pubmed	Photo	Computer	CS	Physics
AUC	DGI	93.88 ± 1.00	95.98 ± 0.72	96.30 ± 0.20	80.95 ± 0.39	81.27 ± 0.51	93.81 ± 0.20	93.51 ± 0.22
	MVGRL	93.33 ± 0.68	88.66 ± 5.27	95.89 ± 0.22	69.58 ± 2.04	92.37 ± 0.78	91.45 ± 0.67	OOM
	GRACE	82.67 ± 0.27	87.74 ± 0.96	94.09 ± 0.92	81.72 ± 0.31	82.94 ± 0.20	85.26 ± 2.07	83.48 ± 0.96
	GCA	81.46 ± 4.86	84.81 ± 1.25	94.20 ± 0.59	70.02 ± 9.66	89.92 ± 0.91	84.35 ± 1.13	85.24 ± 5.41
	CCA-SSG	93.88 ± 0.95	94.69 ± 0.95	96.63 ± 0.15	73.98 ± 1.31	75.91 ± 1.50	96.80 ± 0.16	96.74 ± 0.05
	CAN	93.67 ± 0.62	94.56 ± 0.68	−	97.00 ± 0.28	96.03 ± 0.37	−	−
	SIG-VAE	94.10 ± 0.68	92.88 ± 0.74	85.89 ± 0.54	94.98 ± 0.86	91.14 ± 1.10	95.26 ± 0.36	98.76 ± 0.23
	GraphMAE	90.70 ± 0.01	70.55 ± 0.05	69.12 ± 0.01	77.42 ± 0.02	75.14 ± 0.02	91.47 ± 0.01	87.61 ± 0.02
	SeeGera-v1	94.95 ± 0.72	96.75 ± 0.54	97.07 ± 2.20	98.40 ± 0.08	96.87 ± 0.29	97.82 ± 0.11	98.95 ± 0.06
	SeeGera-v2	95.37 ± 0.60	96.81 ± 0.51	97.79 ± 0.22	98.47 ± 0.05	97.28 ± 0.00	97.83 ± 0.11	98.97 ± 0.04
	SeeGera-v3	**95.50 ± 0.71**	**97.04 ± 0.47**	**97.87 ± 0.20**	**98.64 ± 0.05**	**97.70 ± 0.19**	**98.42 ± 0.13**	**99.03 ± 0.05**
AP	DGI	93.60 ± 1.14	96.18 ± 0.68	95.65 ± 0.26	81.01 ± 0.47	82.05 ± 0.50	92.79 ± 0.31	92.10 ± 0.29
	MVGRL	92.95 ± 0.82	89.37 ± 4.55	95.53 ± 0.30	63.43 ± 2.02	91.73 ± 0.40	89.14 ± 0.93	OOM
	GRACE	82.36 ± 0.24	86.92 ± 1.11	93.26 ± 1.20	81.18 ± 0.37	83.12 ± 0.23	83.90 ± 2.20	82.20 ± 1.06
	GCA	80.87 ± 4.11	81.93 ± 1.76	93.31 ± 0.75	65.17 ± 10.11	89.50 ± 0.64	83.24 ± 1.16	82.80 ± 4.46
	CCA-SSG	93.74 ± 1.15	95.06 ± 0.91	95.97 ± 0.23	67.99 ± 1.60	69.47 ± 1.94	96.40 ± 0.30	96.26 ± 0.10
	CAN	94.49 ± 0.60	95.49 ± 0.61	−	96.68 ± 0.30	95.96 ± 0.38	−	−
	SIG-VAE	94.79 ± 0.71	94.21 ± 0.53	85.02 ± 0.49	94.53 ± 0.93	91.23 ± 1.04	94.93 ± 0.37	98.85 ± 0.12
	GraphMAE	89.52 ± 0.01	74.50 ± 0.04	87.92 ± 0.01	77.18 ± 0.02	75.80 ± 0.01	83.58 ± 0.01	86.44 ± 0.03
	SeeGera-v1	95.53 ± 0.54	97.10 ± 0.49	97.25 ± 2.07	98.32 ± 0.09	96.73 ± 0.31	98.30 ± 0.11	99.10 ± 0.09
	SeeGera-v2	95.90 ± 0.49	97.17 ± 0.46	**97.89 ± 0.21**	98.37 ± 0.09	97.15 ± 0.00	98.33 ± 0.10	99.13 ± 0.06
	SeeGera-v3	**95.92 ± 0.68**	**97.33 ± 0.46**	97.87 ± 0.20	**98.48 ± 0.06**	**97.50 ± 0.15**	**98.53 ± 0.18**	**99.18 ± 0.04**

Table 2: Attribute inference performance w.r.t the MSE metric. The best result in each dataset is highlighted in bold.

Method	Cora	Citeseer	Pubmed	Photo	Computer	CS	Physics
CAN	-	-	-	0.22 ± 0.00	0.23 ± 0.01	-	-
GATE	$1.80 \times 10^{-3} \pm 2.15 \times 10^{-4}$	$\mathbf{4.58 \times 10^{-4} \pm 8.07 \times 10^{-5}}$	$3.90 \times 10^{-4} \pm 1.99 \times 10^{-5}$	0.24 ± 0.01	0.25 ± 0.01	2.03 ± 0.25	OOM
GraphMAE	$\mathbf{1.57 \times 10^{-3} \pm 7.42 \times 10^{-5}}$	$8.68 \times 10^{-4} \pm 1.30 \times 10^{-4}$	$7.29 \times 10^{-4} \pm 2.66 \times 10^{-5}$	0.48 ± 0.00	0.48 ± 0.00	2.70 ± 0.06	2.97 ± 0.05
SeeGera-v1	$1.90 \times 10^{-3} \pm 8.18 \times 10^{-5}$	$4.87 \times 10^{-4} \pm 2.62 \times 10^{-6}$	$4.21 \times 10^{-4} \pm 4.70 \times 10^{-5}$	0.22 ± 0.00	0.23 ± 0.14	2.12 ± 0.06	2.15 ± 0.05
SeeGera-v2	$1.89 \times 10^{-3} \pm 8.13 \times 10^{-5}$	$4.86 \times 10^{-4} \pm 2.24 \times 10^{-6}$	$3.68 \times 10^{-4} \pm 7.11 \times 10^{-6}$	**0.21 ± 0.01**	0.23 ± 0.01	2.08 ± 0.07	**2.14 ± 0.03**
SeeGera-v3	$1.89 \times 10^{-3} \pm 8.13 \times 10^{-5}$	$4.84 \times 10^{-4} \pm 2.96 \times 10^{-6}$	$\mathbf{3.66 \times 10^{-4} \pm 7.34 \times 10^{-6}}$	**0.21 ± 0.01**	**0.22 ± 0.00**	**1.93 ± 0.07**	**2.14 ± 0.03**

Table 3: Node classification performance w.r.t. the classification accuracy. We highlight the best results in bold.
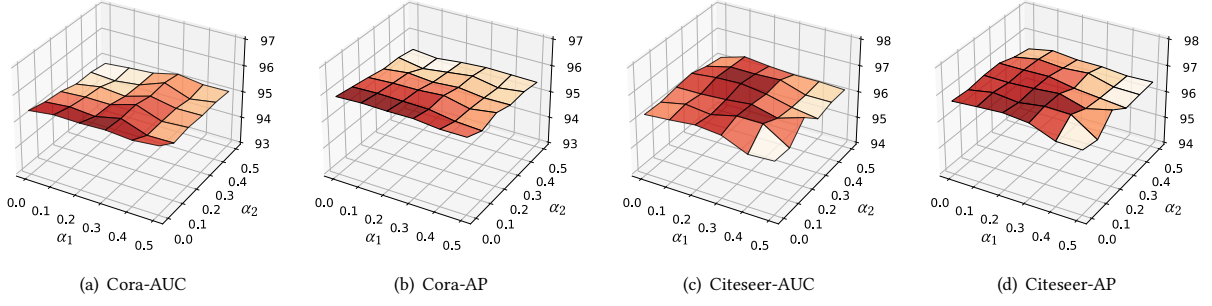
Method	Cora	Citeseer	Pubmed	Photo	Computer	CS	Physics
DGI	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.6	91.61 ± 0.22	83.95 ± 0.47	92.15 ± 0.63	94.51 ± 0.52
MVGRL	83.5 ± 0.4	73.3 ± 0.5	80.1 ± 0.7	91.74 ± 0.07	87.52 ± 0.11	92.11 ± 0.12	95.33 ± 0.03
GRACE	81.9 ± 0.4	71.2 ± 0.5	80.6 ± 0.4	92.15 ± 0.24	86.25 ± 0.25	92.93 ± 0.01	95.26 ± 0.02
CCA-SSG	84.0 ± 0.4	73.1 ± 0.3	81.0 ± 0.4	**93.14 ± 0.14**	**88.74 ± 0.28**	93.31 ± 0.22	95.38 ± 0.06
GraphMAE	84.2 ± 0.4	**73.4 ± 0.4**	**81.1 ± 0.4**	92.98 ± 0.35	88.34 ± 0.27	93.08 ± 0.17	95.30 ± 0.12
SeeGera-v1	82.9 ± 0.4	71.7 ± 0.6	78.9 ± 0.9	92.53 ± 0.41	88.44 ± 0.24	93.72 ± 0.29	**95.40 ± 0.10**
SeeGera-v2	84.0 ± 0.4	73.0 ± 0.8	80.4 ± 0.4	92.70 ± 0.42	88.39 ± 0.26	93.83 ± 0.22	95.39 ± 0.08
SeeGera-v3	**84.3 ± 0.4**	73.0 ± 0.8	80.4 ± 0.4	92.81 ± 0.45	88.39 ± 0.26	**93.84 ± 0.11**	95.39 ± 0.08

model in graph representation learning. In particular, the consistent outperformance of SeeGera-v2 over SeeGera-v1 verifies the importance of capturing the correlations between node and feature embeddings. Further, the improvement of SeeGera-v3 over SeeGera-v2 shows the necessity of structure/feature masking.

5.3 Attribute Inference (RQ2)

Attribute inference is a task that predicts values of missing node attributes. Similar as in link prediction, we hide a certain percentage of node features and train on the rest. To construct the training/validation/test set, we randomly select 70%/10%/20% node features. The validation set is used for hyper-parameter tuning and

(a) Cora-AUC (b) Cora-AP (c) Citeseer-AUC (d) Citeseer-AP

Figure 2: Hyper-parameter sensitivity analysis on the masking rates α_1 and α_2 in terms of link prediction. The darker the color, the larger the value.

early stopping with a patience of 100 epoches. We take the Mean Squared Error (MSE) as the evaluation metric. The smaller the value, the better the performance. In this task, we compare SeeGera with generative graph SSL methods that reconstruct features in their decoders, including CAN [20], GATE [23] and GraphMAE [7]. For GCL models and other generative graph SSL methods that recover links only, they cannot be easily adapted to the task, so we do not take them as baselines. For each method, we run experiments 10 times and report the average results in Table 2. For Cora and Citeseer, we normalize node features for fair comparison, so CAN cannot be applied. From the table, while CAN, GATE and Graph-MAE can perform well on some datasets, they cannot consistently provide excellent performance. For example, GraphMAE achieves the best result on Cora, but it performs very poorly on Citeseer. Further, SeeGera-v3 outperforms other competitors on 5 out of 7 datasets. This shows the effectiveness of our proposed feature reconstruction method and also the masking mechanism. We also notice that, in all cases, SeeGera-v2 achieves better performance than SeeGera-v1, which again verifies the necessity of capturing correlations between node and feature embeddings.

5.4 Node Classification (RQ3)

To further study SeeGera, we generalize learned embeddings to the node classification task. After node embeddings are trained on the entire graph, we train an additional classifier. Here, we employ Logistic Regression as the classifier. For Cora, Citeseer and Pubmed, we use the public split for evaluation, where each class has fixed 20 nodes for training, another fixed 500 nodes and 1000 nodes for validation and testing, respectively. For other datasets, we randomly split the nodes into 10%/10%/80% training/validation/test sets. We use classification accuracy as the metric to evaluate the model performance. Since GCL methods have been shown to perform well in classification tasks, we compare SeeGera with 4 state-of-the-arts, including DGI, MVGRL, GRACE and CCA-SSA. We also take the recently proposed generative model GraphMAE as baseline, because it bridges the gap between generative graph SSL models and GCL methods in terms of classification tasks. Table 3 summarizes the classification results on all the datasets. From the table, we see that CCA-SSG, GraphMAE and SeeGera lead other competitors

and they almost tie. This shows that SeeGera achieves comparable performance with the state-of-the-art methods in the node classification task. Further, with the significant advantage in link prediction and attribute inference tasks, we conclude that SeeGera, a VGAE-based graph SSL method, can generate versatile node representations that can be widely used in various downstream tasks.

5.5 Parameter Analysis (RQ4)

We end this section with a sensitivity analysis on the key hyper-parameters in SeeGera, i.e., the structure masking rate α_1 and the feature masking rate α_2. Specifically, we explore the stability of SeeGera w.r.t. the perturbation of α_1 and α_2. We conduct experiments on the link prediction task by varying these parameters from 0 to 0.5, and keeping others fixed. Figure 2 illustrates the AUC and AP scores of SeeGera-v3 under different α_1 and α_2 values on Cora and Citeseer. From the figure, we see that SeeGera-v3 can give very stable performance over a wide range of α_1 and α_2 values, as shown by the plateau in the figure. This demonstrates the insensitivity of SeeGera w.r.t. these two hyper-parameters.

6 CONCLUSIONS

We studied generative graph SSL in this paper and proposed SeeGera, which enhances the family of VGAE on graph representation learning. Specifically, SeeGera adopts the hierarchical variational framework in SIG-VAE and mainly focuses on feature reconstruction and structure/feature masking. On the one hand, SeeGera co-embeds both nodes and features in the encoder and computes their embeddings by assuming they are independent and correlated, respectively. After that, feature embeddings that contain rich semantic information on features are combined with node embeddings to provide more fine-grained information for feature reconstruction in the decoder. On the other hand, we injected the masking mechanism into SeeGera by adding an additional layer to the hierarchical variational framework. We conducted extensive experiments to evaluate the performance of SeeGera. The results show that SeeGera significantly outperforms other competitors in link prediction and attribute inference, and achieves comparable results with them in node classification. This further verifies the power of generative graph SSL methods in graph representation learning.

REFERENCES

[1] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2015. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519* (2015).

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[3] Alberto Garcia Duran and Mathias Niepert. 2017. Learning graph representations with embedding propagation. *Advances in neural information processing systems* 30 (2017).

[4] Arman Hasanzadeh, Ehsan Hajiramezanali, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Semi-implicit graph variational auto-encoders. In *NeurIPS*. 10711–10722.

[5] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*. PMLR, 4116–4126.

[6] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[7] Zhenyu Hou, Xiao Liu, Yuxiao Dong, Chunjie Wang, Jie Tang, et al. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. *arXiv preprint arXiv:2205.10803* (2022).

[8] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265* (2019).

[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1857–1867.

[10] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. 2020. Sub-graph contrast for scalable self-supervised graph representation learning. In *ICDM*. IEEE, 222–231.

[11] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

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

[13] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[14] Alexander C Li, Alexei A Efros, and Deepak Pathak. 2022. Understanding Collapse in Non-Contrastive Learning. *arXiv preprint arXiv:2209.15007* (2022).

[15] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, and Philip Yu. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[16] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. 2021. Anomaly detection on attributed networks via contrastive self-supervised learning. *TNNLS* 33, 6 (2021), 2378–2392.

[17] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.

[18] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding attributed networks. In *WSDM*. 393–401.

[19] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*. 2609–2615.

[20] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6519–6528.

[21] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1150–1160.

[22] Tom Rainforth, Adam R Kosiorek, Tuan Anh Le, Chris J Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. 2018. Tighter variational bounds are not necessarily better. In *ICML*. 4274–4282.

[23] Amin Salehi and Hasan Davulcu. 2019. Graph attention auto-encoders. *arXiv preprint arXiv:1905.10715* (2019).

[24] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*. 243–246.

[25] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. Ladder variational autoencoders. In *NeurIPS*. 3738–3746.

[26] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000* (2019).

[27] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. 2021. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514* (2021).

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need, Vol. 30.

[29] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. *ICLR (Poster)* 2, 3 (2019), 4.

[30] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 889–898.

[31] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. 2021. Self-supervised graph-level representation learning with local and global structure. In *International Conference on Machine Learning*. PMLR, 11548–11558.

[32] Mingzhang Yin and Mingyuan Zhou. 2018. Semi-Implicit Variational Inference. In *ICML*. 5660–5669.

[33] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. 2018. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems* 31 (2018).

[34] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*. PMLR, 5708–5717.

[35] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020), 5812–5823.

[36] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. 2020. When does self-supervision help graph convolutional networks?. In *international conference on machine learning*. PMLR, 10871–10880.

[37] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. 2021. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 76–89.

[38] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140* (2020).

[39] Sheng Zhou, Xin Wang, Jiajun Bu, Martin Ester, Pinggang Yu, Jiawei Chen, Qihao Shi, and Can Wang. 2020. DGE: Deep Generative Network Embedding Based on Commonality and Individuality. In *AAAI*.

[40] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. 2021. An empirical study of graph contrastive learning. *arXiv preprint arXiv:2109.01116* (2021).

[41] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).

[42] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*. 2069–2080.

A DATASETS

We use 7 public datasets which do not have license. We next briefly introduce them as follows.

Cora, *Citeseer* and *Pubmed* [13] are three citation networks, where nodes represent publications and edges are citation links. Features for each node are the keywords it contains. Each dimension in the feature vector indicates the presence of a keyword in the publication. Nodes in these datasets are associated with labels that describe research topics of publications.

Coauther CS and *Coauther Physics* are co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge [24]. In these datasets, nodes are authors and edges capture the co-authorship. Further, node features represent keywords in each author's papers, and class labels indicate the study fields for authors.

Amazon Computer and *Amazon Photo* are extracted from the Amazon co-purchase graph [17], where nodes represent goods and edges indicate that two goods are frequently bought together. Node features are bag-of-words encoded product reviews and class labels are the product categories. The statistics of these datasets are summarized in Table 4.

B PSEUDOCODES

This section summarizes the pseudocodes of SeeGera-v3 in Alg. 1.

Algorithm 1 SeeGera-v3

Input: A, X, $p(\tilde{G}|A, X)$, $\tilde{q}(\epsilon)$, $\hat{q}(\epsilon)$, ρ, neural networks T_{ϕ_1} and T_{ϕ_2}
Output: ϕ_1 and ϕ_2

1: Initialize ϕ_1, ϕ_2, set $\underline{\mathcal{L}}_K^J = 0$
2: **while** not converged **do**
3:      Sample $\tilde{G} \sim p(\tilde{G}|A, X)$
4:      **for** $k = 1$ to K **do**
5:          Sample $\tilde{\psi}_1^k = T_{\phi_1}(\tilde{G}, \tilde{\epsilon}_1^k)$, where $\tilde{\epsilon}_1^k \sim \tilde{q}(\epsilon)$ ▷ Eq. 12
6:          Sample $\tilde{\psi}_2^k = T_{\phi_2}(\tilde{G}, \tilde{\psi}_1^k, \hat{\epsilon}_2^k)$, where $\hat{\epsilon}_2^k \sim \hat{q}(\epsilon)$ ▷ Eq. 14
7:      **end for**
8:      **for** $j = 1$ to J **do**
9:          Sample $\epsilon_1^j \sim \tilde{q}(\epsilon)$, $\epsilon_2^j \sim \hat{q}(\epsilon)$
10:         Sample $\psi_1^j = [(\mu^{\mathcal{V}})_j, (\Sigma^{\mathcal{V}})_j] = T_{\phi_1}(\tilde{G}, \epsilon_1^j)$ ▷ Eq. 12
11:         Sample $\psi_2^j = [(\mu^{\mathcal{F}})_j, (\Sigma^{\mathcal{F}})_j] = T_{\phi_2}(\tilde{G}, \psi_1^j, \epsilon_2^j)$ ▷ Eq. 14
12:         Sample $\epsilon_j^{\mathcal{V}} \sim \mathcal{N}(0, I)$, $\epsilon_j^{\mathcal{A}} \sim \mathcal{N}(0, I)$
13:         Sample $(Z^{\mathcal{V}})_j = (\mu^{\mathcal{V}})_j + (\Sigma^{\mathcal{V}})_j \odot \epsilon_j^{\mathcal{V}}$
14:         Sample $(Z^{\mathcal{F}})_j = (\mu^{\mathcal{F}})_j + (\Sigma^{\mathcal{F}})_j \odot \epsilon_j^{\mathcal{A}}$
15:         Set $tmp_1 = -\log \Omega_j$ ▷ Eq. 19
16:         Set $tmp_2 = \log p(\tilde{G}|(Z^{\mathcal{V}})_j, (Z^{\mathcal{F}})_j)$
17:         Set $tmp_3 = \log p((Z^{\mathcal{V}})_j, (Z^{\mathcal{F}})_j)$
18:         Update $\underline{\mathcal{L}}_K^J = \underline{\mathcal{L}}_K^J + e^{tmp_1 + tmp_2 + tmp_3}$
19:      **end for**
20:      Update $\underline{\mathcal{L}}_K^J = \log \underline{\mathcal{L}}_K^J - \log J$
21:      Update $\phi_1 = \phi_1 + \rho \nabla_{\phi_1} \underline{\mathcal{L}}_K^J$
22:      Update $\phi_2 = \phi_2 + \rho \nabla_{\phi_2} \underline{\mathcal{L}}_K^J$
23: **end while**
24: **return** ϕ_1 and ϕ_2

Table 4: Statistics of datasets used in experiments

Datasets	#Nodes	#Edges	#Features	#Classes
Cora	2,708	5,278	1,433	7
Citeseer	3,327	4,676	3,703	6
Pubmed	19,717	88,651	500	3
Coauthor CS	18,333	327,576	6,805	15
Coauthor Physics	34,493	991,848	8,451	5
Amazon Computer	13,752	574,418	767	10
Coauthor Physics	7,650	287,326	745	8

C ABLATION STUDY

We conduct an ablation study to investigate the main components in SeeGera. In particular, we have extensively compared SeeGera-v1, SeeGera-v2 and SeeGera-v3 in our experiments. The advantage of SeeGera-v2 over SeeGera-v1 shows the importance of capturing the correlations between node and feature embeddings. Also, the outperformance of SeeGera-v3 over SeeGera-v2 verifies the importance of the masking mechanism. Further, to show the effectiveness of our proposed feature reconstruction method, we remove feature embeddings in the encoder and feed only node embeddings into GCN in the decoder to reconstruct features. We call this variant SeeGera_nf (**no f**eature embedding). Table 5 shows the results on attribute inference. We exclude SeeGera-v3 in the table, because it further uses the masking mechanism while others not. From the table, we see that both SeeGera-v1 and SeeGera-v2 outperform SeeGera_nf. This shows the importance of using both node and feature embeddings for feature reconstruction.

Table 5: The comparison between SeeGera and SeeGera_nf in the attribute inference task.

Method	Cora	Citeseer	CS
SeeGera_nf	$1.91 \times 10^{-3} \pm 3.78 \times 10^{-5}$	$5.15 \times 10^{-4} \pm 1.02 \times 10^{-6}$	2.14 ± 0.07
SeeGera-v1	$1.90 \times 10^{-3} \pm 8.18 \times 10^{-5}$	$4.87 \times 10^{-4} \pm 2.62 \times 10^{-6}$	2.12 ± 0.06
SeeGera-v2	$1.89 \times 10^{-3} \pm 8.13 \times 10^{-5}$	$4.86 \times 10^{-4} \pm 2.24 \times 10^{-6}$	2.08 ± 0.07

D IMPLEMENTATION DETAILS

We implemented SeeGera by PyTorch. The model is initialized by Glorot initialization and trained by Adam. We run the model for 3500 epochs on all the datasets. In particular, for the task of node classification, we further run the logistic regression classifier for 500 epochs. We adopt GCNs in both encoder and decoder for all the datasets except CS. For CS, we use MLPs to replace GNNs instead. We choose Bernoulli noise Ber(0.5) for both $\tilde{\epsilon} \sim \tilde{q}(\epsilon)$ and $\hat{\epsilon} \sim \hat{q}(\epsilon)$, and set the noise dimension to 5 in all tasks. As suggested by [25], we use warm-up during the first 300 epochs to gradually impose the prior regularization terms $D_{KL}(q_1(Z^{\mathcal{V}}|\psi_1)||p(Z^{\mathcal{V}}))$ and $D_{KL}(q_2(Z^{\mathcal{F}}|\psi_2)||p(Z^{\mathcal{F}}))$. For both priors $p(Z^{\mathcal{V}})$ and $p(Z^{\mathcal{F}})$, we assume that they follow standard multivariate normal distributions. In the node classification task, most results of baselines are publicly available and we directly report these results from their original papers. For the results that are missing, we run the released codes by their authors and fine-tune the model hyper-parameters. In the link prediction and attribute inference tasks, since most baselines are not studied in these tasks, we run their released codes with fine-tuning. For fairness, we run all the experiments on a server

with a single NVIDIA A100 GPU with 80G memory. For simplicity, we set $J = K = 1$ in Eq. 19 for both SeeGera and SIG-VAE. One may further refer to [22] for better value selection. All the datasets and codes are provided at https://github.com/SeeGera/SeeGera. We provide the detailed hyper-parameter settings of SeeGera-v3 on different datasets in Tables 6- 8. All hyper-parameters are selected through small grid search, and the search space is provided as:

- Number of layers in the encoder L_1: {1, 2, 3}
- Number of layers in the decoder L_2: {1, 2, 3}
- Learning rate of SeeGera: {1e-3, 5e-3, 1e-2}
- Dropout of SeeGera: {0, 0.1, 0.3, 0.5, 0.7, 0.9}
- Weight decay of SeeGera: {5e-5, 1e-4, 5e-4, 1e-3}
- Structure masking rate α_1: {0, 0.1, 0.2, 0.3, 0.4, 0.5}
- Feature masking rate α_2: {0, 0.1, 0.2, 0.3, 0.4, 0.5}
- Learning rate of logistic regression: {1e-3, 5e-3, 1e-2}
- Dropout of logistic regression: {0, 0.1, 0.3, 0.5, 0.7, 0.9}
- Weight decay of logistic regression: {5e-5, 1e-4, 5e-4, 1e-3}

Table 6: Hyper-parameter setting details of SeeGera-v3 in link prediction.

Dataset	L_1	L_2	lr	dropout	wd	α_1	α_2
Cora	2	3	1e-3	0.3	5e-5	0.3	0.0
Citeseer	1	2	1e-3	0.0	1e-4	0.6	0.0
Pubmed	2	1	5e-3	0.5	0.0	0.3	0.1
Photo	1	1	5e-3	0.0	0.0	0.5	0.0
Computer	2	2	1e-3	0.0	0.0	0.4	0.0
CS	1	2	1e-3	0.0	0.0	0.5	0.5
Physics	2	2	1e-3	0.0	0.0	0.2	0.0

Table 7: Hyper-parameter setting details of SeeGera-v3 in attribute inference.

Dataset	L_1	L_2	lr	dropout	wd	α_1	α_2
Cora	1	1	1e-3	0.1	5e-5	0.4	0.0
Citeseer	1	1	1e-3	0.0	0.0	0.3	0.0
Pubmed	2	2	1e-3	0.0	0.0	0.0	0.1
Photo	2	3	1e-3	0.0	0.0	0.3	0.0
Computer	2	3	1e-3	0.0	0.0	0.2	0.0
CS	1	3	1e-3	0.0	0.0	0.1	0.5
Physics	2	2	1e-3	0.0	0.0	0.0	0.0

Table 8: Hyper-parameter setting details of SeeGera-v3 in node classification.

Dataset	SeeGera							Logistic Regression		
	L_1	L_2	lr	dropout	wd	α_1	α_2	lr	dropout	wd
Cora	2	2	1e-3	0.3	1e-3	0.2	0.1	1e-3	0.9	1e-3
Citeseer	2	2	5e-3	0.3	5e-4	0.0	0.0	1e-3	0.9	0.0
Pubmed	2	1	5e-3	0.7	0.0	0.0	0.0	1e-3	0.3	0.0
Photo	1	2	1e-3	0.0	5e-4	0.5	0.5	5e-3	0.7	5e-5
Computer	1	3	5e-3	0.0	5e-4	0.0	0.0	5e-3	0.3	1e-4
CS	2	3	1e-2	0.1	1e-3	0.4	0.0	1e-2	0.0	1e-3
Physics	1	3	1e-2	0.0	0.0	0.0	0.0	1e-2	0.5	5e-5

E VARIATIONAL LOWER BOUND

In this section, we show the derivation on the variational lower bounds in detail.

$$
\begin{aligned}
\mathcal{L} &= \mathbb{E}_{h_{\phi_1}(Z^{\mathcal{V}})} \mathbb{E}_{h_{\phi_2}(Z^{\mathcal{F}})} \left[\log \frac{p(Z^{\mathcal{V}}|A,X)p(Z^{\mathcal{F}}|X^T)p(A,X)}{h_{\phi_1}(Z^{\mathcal{V}})h_{\phi_2}(Z^{\mathcal{F}})} \right] \\
&= -D_{KL}(h_{\phi_1}(Z^{\mathcal{V}})||p(Z^{\mathcal{V}}|A,X)) - D_{KL}(h_{\phi_2}(Z^{\mathcal{F}})||p(Z^{\mathcal{F}}|X^T)) + \log p(A,X) \\
&\geq -\mathbb{E}_{\psi_1 \sim q_{\phi_1}(\psi_1)} D_{KL}(q_1(Z^{\mathcal{V}}|\psi_1)||p(Z^{\mathcal{V}}|A,X)) \\
&\quad - \mathbb{E}_{\psi_2 \sim q_{\phi_2}(\psi_2)} D_{KL}(q_2(Z^{\mathcal{F}}|\psi_2)||p(Z^{\mathcal{F}}|X^T)) + \log p(A,X) \\
&= \mathbb{E}_{\psi_1 \sim q_{\phi_1}(\psi)} \mathbb{E}_{Z^{\mathcal{V}} \sim q_1(Z^{\mathcal{V}}|\psi_1)} \mathbb{E}_{\psi_2 \sim q_{\phi_2}(\psi)} \mathbb{E}_{Z^{\mathcal{F}} \sim q_2(Z^{\mathcal{F}}|\psi_2)} \\
&\quad \left[\log \frac{p(A,X,Z^{\mathcal{V}},Z^{\mathcal{F}})}{q_1(Z^{\mathcal{V}}|\psi_1)q_2(Z^{\mathcal{F}}|\psi_2)} \right] \\
&= \underline{\mathcal{L}}_1,
\end{aligned}
$$

where D_{KL} is the KL divergence and we employ $D_{KL}(\mathbb{E}_{\psi} q(Z|\psi)||p(Z)) \leq \mathbb{E}_{\psi} D_{KL}(q(Z|\psi)||p(Z))$ according to [32]. To better understand $\underline{\mathcal{L}}_1$, we decompose the joint distribution $p(A,X,Z^{\mathcal{V}},Z^{\mathcal{F}})$ as

$$
p(A,X,Z^{\mathcal{V}},Z^{\mathcal{F}}) = p(Z^{\mathcal{V}})p(Z^{\mathcal{F}}) \prod_{i,j\in\mathcal{V}} p(A_{ij}|Z_i^{\mathcal{V}},Z_j^{\mathcal{V}}) \prod_{i\in\mathcal{V},r\in\mathcal{F}} p(X_{ir}|Z_i^{\mathcal{V}},Z_r^{\mathcal{F}})
$$

and expand $\underline{\mathcal{L}}_1$ to derive:

$$
\begin{aligned}
\underline{\mathcal{L}}_1 &= \mathbb{E}_{\psi_1 \sim q_{\phi_1}(\psi_1)} \mathbb{E}_{Z^{\mathcal{V}} \sim q_1(Z^{\mathcal{V}}|\psi_1)} \left[\sum_{i,j\in\mathcal{V}} \log p(A_{ij}|Z_i^{\mathcal{V}},Z_j^{\mathcal{V}}) \right] \\
&\quad + \mathbb{E}_{\psi_1 \sim q_{\phi_1}(\psi_1)} \mathbb{E}_{Z^{\mathcal{V}} \sim q_1(Z^{\mathcal{V}}|\psi_1)} \mathbb{E}_{\psi_2 \sim q_{\phi_2}(\psi_2)} \mathbb{E}_{Z^{\mathcal{F}} \sim q_2(Z^{\mathcal{F}}|\psi_2)} \\
&\quad \left[\sum_{i\in\mathcal{V},r\in\mathcal{F}} \log p(X_{ir}|Z_i^{\mathcal{V}},Z_r^{\mathcal{F}}) \right] \\
&\quad - \mathbb{E}_{\psi_1 \sim q_{\phi_1}(\psi_1)} D_{KL}(q_1(Z^{\mathcal{V}}|\psi_1)||p(Z^{\mathcal{V}})) \\
&\quad - \mathbb{E}_{\psi_2 \sim q_{\phi_2}(\psi_2)} D_{KL}(q_2(Z^{\mathcal{F}}|\psi_2)||p(Z^{\mathcal{F}})).
\end{aligned}
$$

Here, $q_1(Z^{\mathcal{V}}|\psi_1)$ and $q_2(Z^{\mathcal{F}}|\psi_2)$ are encoders that generate embeddings of nodes and features, respectively; $p(A_{ij}|Z_i^{\mathcal{V}},Z_j^{\mathcal{V}})$ and $p(X_{ir}|Z_i^{\mathcal{V}},Z_r^{\mathcal{F}})$ are decoders that reconstruct links and features from learned embeddings. The first two terms in the equation correspond to the negative reconstruction loss for links and features, while the last two terms are regularizers that promote the closeness between variational distributions and prior distributions.

Similarly, we can expand $\underline{\mathcal{L}}_2$ as:

$$
\begin{aligned}
\underline{\mathcal{L}}_2 &= \mathbb{E}_{\psi_1 \sim q_{\phi_1}(\psi_1)} \mathbb{E}_{\psi_2 \sim q_{\phi_2}(\psi_2|\psi_1)} \mathbb{E}_{(Z^{\mathcal{V}},Z^{\mathcal{F}}) \sim q(Z^{\mathcal{V}},Z^{\mathcal{F}}|\psi_1,\psi_2)} \\
&\quad \left[\sum_{i,j\in\mathcal{V}} \log p(A_{ij}|Z_i^{\mathcal{V}},Z_j^{\mathcal{V}}) + \sum_{i\in\mathcal{V},r\in\mathcal{F}} \log p(X_{ir}|Z_i^{\mathcal{V}},Z_r^{\mathcal{F}}) \right] \\
&\quad - \mathbb{E}_{(\psi_1,\psi_2) \sim q_\phi(\psi_1,\psi_2)} D_{KL}(q(Z^{\mathcal{V}},Z^{\mathcal{F}}|\psi_1,\psi_2)||p(Z^{\mathcal{V}},Z^{\mathcal{F}})).
\end{aligned}
$$