Decoupled Self-supervised Learning for Graphs

Teng Xiao[1], **Zhengyu Chen**[2], **Zhimeng Guo**[1], **Zeyang Zhuang**[3], **Suhang Wang**[1]
[1]The Pennsylvania State University, [2]Zhejiang University, [3]Tongji University
{tengxiao,zhimeng,szw494}@psu.edu,chenzhengyu@zju.edu.cn
zeyangzhuang0315@gmail.com

Abstract

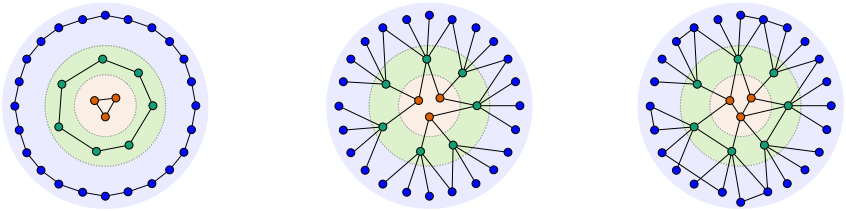
This paper studies the problem of conducting self-supervised learning for node representation learning on graphs. Most existing self-supervised learning methods assume the graph is homophilous, where linked nodes often belong to the same class or have similar features. However, such assumptions of homophily do not always hold in real-world graphs. We address this problem by developing a decoupled self-supervised learning (DSSL) framework for graph neural networks. DSSL imitates a generative process of nodes and links from latent variable modeling of the semantic structure, which decouples different underlying semantics between different neighborhoods into the self-supervised learning process. Our DSSL framework is agnostic to the encoders and does not need prefabricated augmentations, thus is flexible to different graphs. To effectively optimize the framework, we derive the evidence lower bound of the self-supervised objective and develop a scalable training algorithm with variational inference. We provide a theoretical analysis to justify that DSSL enjoys the better downstream performance. Extensive experiments on various types of graph benchmarks demonstrate that our proposed framework can achieve better performance compared with competitive baselines.

1 Introduction

Graph-structured data is ubiquitous in the real world, such as social networks, knowledge graphs, and molecular structures. In recent years, graph neural networks (GNNs) [14, 25, 50, 57, 58] have been proven to be powerful in node representation learning over graph-structured data. Typically, GNNs are trained with annotated labeled data in a supervised manner. However, collecting labeled data is expensive and impractical in many applications, especially for those requiring domain knowledge, such as medicine and chemistry [69, 18]. Moreover, supervised learning may suffer from problems of less-transferrable, over-fitting, and poor generalization when the task-specific labels are scarce [8, 62].

Recently, self-supervised learning (SSL) provides a promising learning paradigm that reduces the dependence on manual labels in the image domain [5, 10, 11, 17]. Compared to image data, there are unique challenges in designing self-supervised learning schemes for graph-structured data since nodes in the graph are correlated with each other rather than completely independent, and geometric structures are essential and heavily impact the performance in downstream tasks [25]. A number of recent works [51, 16, 62, 68, 46, 63, 43] have studied graph self-supervised learning and confirm that it can learn transferrable and generalizable node representations without any labels. Typically, there are two main self-supervised schemes to capture structure information in graphs [51, 62, 43]. The first scheme is reconstructing the vertex adjacency following traditional network-embedding methods [24, 12, 15, 14, 44], which learns an encoder that imposes the topological closeness of nodes in the graph structure on latent representations. The key assumption behind this scheme is that neighboring nodes have similar representations [51, 44]. However, this assumption over-emphasizes proximity [51, 62, 45] and does not always hold true for heterophilic and non-homophilous (mixing) graphs. In comparison, contrastive learning methods [51, 16, 62, 68, 46, 63] construct two graph

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) homophilous pattern. (b) heterophilic pattern. (c) mixing pattern.

Figure 1: An illustration examples of different types of graphs. Nodes with similar labels typically have similar neighborhood patterns in all types of graph. This assumption is general than the standard homophily assumption where nodes with similar semantic label typically be linked with each other.

views via the stochastic augmentation and then learns representations by contrasting views with information maximization principle. While these contrastive methods can capture structure information without directly emphasizing proximity, their performance relies on topology augmentation [67, 48]. Importantly, conducting augmentation for non-homophilous graphs is relatively difficult since linked nodes may be dissimilar, and nodes with high similarities might be farther away from each other. Hence, the above problems pose an important and challenging research question: *How to design an effective self-supervised scheme for node representation learning in non-homophilous graphs?*

We approach this question by taking advantage of neighborhood strategies of nodes for the self-supervised learning on non-homophilous graphs. Our key motivation is that nodes with similar neighborhood patterns should have similar representations. In other words, we expect that the neighborhood distributions can be exploited to distinguish node representations. Our assumption is more general than the standard homophily assumption as shown in Figure 1. For instance, while the gender prediction in common dating networks lacks homophily [1], neighborhood distribution is very informative to the node gender labels, i.e., nodes with similar neighborhoods are likely to be similar.

While the motivation is straightforward, we are faced with two main challenges. The first challenge is capturing the local neighborhood distribution in a self-supervised learning manner. The neighborhood distributions typically follow heterogeneous and diverse patterns. For instance, a node usually connects with others due to the complex interaction of different latent factors and therefore possesses distribution consisting of local mixing patterns wherein certain parts of the neighborhood are homophilous while others are non-homophilous. The lack of supervision obstructs us from modeling the distribution of neighborhoods. The second challenge is capturing the long-range semantic dependencies. As shown in Figure 2, in non-homophilous graphs, nodes with high semantic and local structural similarities might be farther away from each other. For this reason, global semantic information is the objective that we would incorporate for self-supervised learning.

This paper presents a new self-supervised framework, decoupled self-supervised learning (DSSL), to achieve a good balance between these two challenges. At the core of DSSL is the latent variables, which empower the model with the flexibility to decouple the heterogeneous and diverse patterns in local neighborhood distributions and capture the global semantic dependencies in a coherent and synergistic framework. Our contributions can be summarized as follows: (1) We propose a DSSL for performing self-supervised learning on non-homophilous graphs, which can leverage both useful local structure and global semantic information. (2) We develop an efficient training algorithm based on variational inference to simultaneously infer the latent variables and learn encoder parameters. (3) We analyze the properties of DSSL and theoretically show that the learned representations can achieve better downstream performance. (4) We conduct experiments on real-world non-homophilous graphs, and the results demonstrate the effectiveness of our self-supervised learning framework.

2 Related Work

Non-homophilous Graphs. Non-homophilous is known in many settings such as online transaction networks [34], dating networks [1] and molecular networks [66]. Recently, various GNNs [35, 66, 27, 68, 6, 42, 60] have been proposed to deal with non-homophilous graphs with different methods such as potential neighbor discovery [35, 21, 20], adaptive message propagation [6, 57], exploring high-frequency signals [2] and higher-order message passing [66, 5]. Despite their success, they typically consider the semi-supervised setting and are trained with task-specific labeled data, while in

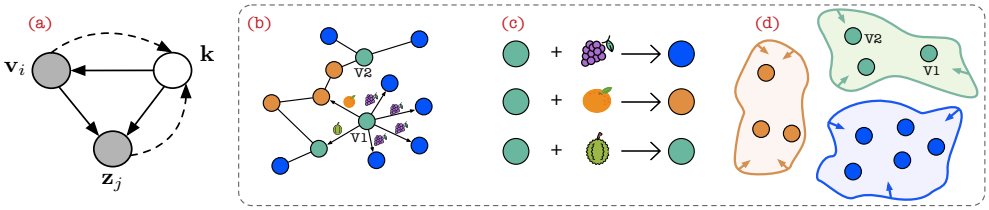


Figure 2: An illustration of (a) a graphical model for DSSL. The discrete latent variable k is used to instantiate a latent mixture-of-Gaussians $\mathbf{v}_i$, which is then decoded to $\mathbf{z}_j$ and (b) a toy subgraph example of non-homophilous graphs where color denotes the class of the node. (c) Our model encodes the semantic shift by decoupling structure since different each node has different latent factors to make connections to its different neighbor, and (d) captures global semantic structure via the semantic clusters, which seeks to push the representations of v_1 and v_2 to their closest prototypes.

practice, labels are often limited, expensive, and even inaccessible. In contrast, in this paper, we study the problem of self-supervised learning: learning node representations without relying on labels.

Self-supervised Learning on Graphs. Self-supervised learning holds great promise for improving representations when labeled data are scarce [5, 10, 11, 17]. Earlier combinations of GNNs and self-supervised learning involve GraphSAGE [14], VGAE [24], and Graphite [13], which typically follow the traditional network-embedding methods [36, 44, 12] and adopt the link reconstruction or random walk principle. Since these methods over-emphasize node proximity at the cost of structural information [51, 43, 62], various graph contrastive learning methods have been proposed [51, 16, 62, 68, 46, 33], which aim to learn representations by contrasting representation under differently augmented views and have achieved promising performance. However, they heavily rely on complex data- or task-specific augmentations, which are prohibitively difficult for non-homophilous graphs. Our work differs from the above methods and aims to answer the question of how to design an effective self-supervised learning scheme for non-homophilous graphs.

Disentangled graph learning. Our work is also related to but different from existing disentangled graph learning that aims to decouple latent factors in the graph. vGraph [41] considers a mixture process to define the conditional probability on each edge. vGraph is a classical shallow network embedding algorithm. However, we instead tackle the problem of self-supervised learning with GNNs. There are a couple of works that explore the disentangled factors in the node-level [31, 30], edge-level [65] and graph-level [61]. Whereas these methods require task-specific labels that can be extremely scarce. By contrast, we address the problem of learning node representations on non-homophilous graphs without labels. Disentangled contrastive learning [26] learns disentangled graph representations without labeled graphs. [59] proposes the self-supervised graph-level representation learning with disentangled local and global structure [59]. Nevertheless, their goal is to conduct graph-level classification tasks, but we tackle a node-level representation problem. [64, 7, 53] consider adding a clustering layer or a prototype clustering component to capture the global information. However, they are still based on contrastive learning [64, 7] or supervised learning [53]. By contrast, our framework is an unsupervised generative model and does not rely on graph augmentations and downstream labels. Moreover, we focus on non-homophilous graphs where connected nodes may not be similar to each other by decoupling the local diverse neighborhood context.

3 Decoupled Self-supervised Learning

In this section, we describe our problem setting with respect to self-supervised learning and demonstrate our approach. An illustration of generative and inference processes is depicted in Figure 2 (a).

3.1 Problem Formulation

We consider a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of $|\mathcal{V}| = N$ nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of $|\mathcal{E}|$ edges between nodes. $\mathbf{A} \in \{0,1\}^{N \times N}$ is the adjacency matrix of G. The (i,j)-th element $\mathbf{A}_{ij} = 1$ if there exists an edge (v_i, v_j) between node v_i and v_j, otherwise $\mathbf{A}_{ij} = 0$. The matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ describes node features. The i-th row of $\mathbf{X}$, i.e., $\mathbf{x}_i$, is the feature of node v_i. Given the graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$, the objective of self-supervised node representation learning is to learn an encoder

function $f_\theta(\mathbf{X}, \mathbf{A}) : \mathbb{R}^{N\times N} \times \mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{N\times D'}$ where θ denotes its parameters, such that the representation of node v_i: $f_\theta(\mathbf{X}, \mathbf{A})[i]$, can be used for downstream tasks such as node classification.

3.2 The Probabilistic Framework

In this section, we introduce our framework, decoupled self-supervised learning, which can learn meaningful node representations on non-homophilous graphs by capturing the intrinsic graph structure. The core idea of DSSL is to model the distributions of node neighbors via a mixture of generative processes in the representation space. Specifically, we model the generation of neighbors by assuming each node has latent heterogeneous factors which are utilized to make connections to its different neighbors. Intuitively, the factor denotes various reasons behind why two nodes are connected in non-homophilous graphs. For instance, two nodes in a school network will be connected depending on some factors such as colleagues, friends, or classmates; in protein networks, even if they do not have similar features, different amino acid types are likely to be connected due to various interactions.

Formally, let $f_\theta(\mathbf{X}, \mathbf{A})[i] = \mathbf{v}_i$ and $f_\theta(\mathbf{X}, \mathbf{A})[j] = \mathbf{z}_j$ be the representations of nodes v_i and v_j, respectively. Here we utilize different notations, i.e., $\mathbf{v}$ and $\mathbf{z}$ to distinguish between central and neighbor nodes since each node plays two roles: the node itself and a specific neighbor of other nodes. Our goal is to find the encoder parameter θ which maximizes the likelihood of distribution $p(\mathbf{z}_j|\mathbf{v}_i;\theta)$ on central node and its neighbor. To model the unobserved factors, we associate every node $\mathbf{v}_i$ with a discrete latent variable k to indicate to which factor $\mathbf{v}_i$ has. Assume that there are K factors in total, the log-likelihood of node neighbors $\mathbf{v}_i$ can be written by marginalizing out the latent variables:

$$\mathcal{L}_{DSSL}(\theta) = \frac{1}{|N(i)|}\sum_{j\in N(i)} \log[p_\theta(\mathbf{z}_j|\mathbf{v}_i)] = \frac{1}{|N(i)|}\sum_{j\in N(i)} \log\Big[\sum_{k=1}^{K} p_\theta(\mathbf{z}_j|\mathbf{v}_i,k)p_\theta(k|\mathbf{v}_i)\Big], \tag{1}$$

where $N(i)$ is the set of out-neighbors of v_i, the distribution $p_\theta(k|\mathbf{v}_i)$ indicates the assignment of latent semantics over central node representation $\mathbf{v}_i$, and $p_\theta(\mathbf{z}_j|\mathbf{v}_i,k)$ is the probability that node v_i and its neighbor v_j are connected under factor k. Unlike previous works [44, 12, 24], which directly encourage nearby nodes to have similar representations, we provide an alternative way to model node neighbors and seek to decouple their latent relationship without any prior on neighbor partitions. The similar high-level ideas of modeling latent variables between two nodes has also been explored in local augmentation in graphs (LA-GCN) [29] and vGraph [41]. Different from them, we consider modeling discrete relations in the learned representation space for self-supervised learning on non-homophilous graphs. In Eq. (1), probabilities $p_\theta(k|\mathbf{v}_i)$ and $p_\theta(\mathbf{z}_j|\mathbf{v}_i,k)$ are not specified, and involve discrete latent variables. To make it solvable, we introduce the following generative process.

Let $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ be the mean and variance of the latent mixture component k, and π_k be its corresponding mixture probability. The generation of a node and its neighbor shown in Figure 2 (a) typically involves three steps: (1) draw a latent variable k from a categorical distribution $p(k)$ on all mixture components, where $p(k)$ is usually defined as uniform distribution $p(k) = \frac{1}{K}$ for unknown graphs and better generalization, (2) draw the central node representation $\mathbf{v}_i$ from the Gaussian distribution $p_\theta(\mathbf{v}_i|k) = \mathcal{N}(\mathbf{v}_i;\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, and (3) draw the neighbor representation $\mathbf{z}_j$ from Gaussian distribution $p_\theta(\mathbf{z}_j|\mathbf{v}_i,k) = \mathcal{N}(\mathbf{z}_j;\boldsymbol{\mu}_{z_j}, \boldsymbol{\Sigma}_j)$ where the mean depends both on central representation and latent variable: $\boldsymbol{\mu}_{z_j} = \mathbf{v}_i + \beta g_\theta(k)$. Here the projector $g_\theta(\cdot)$ denotes another network that embeds latent variable k to the representation space, and β is the parameter that controls the strength of interpolation. In practice, to reduce complexity, we consider using isotropic Gaussian, i.e., $\forall k:\ \Sigma_k = \mathbf{I}\sigma_1^2$ and $\forall j:\ \Sigma_j = \mathbf{I}\sigma_2^2$ where $\mathbf{I}$ is the identity matrix, σ_1 and σ_2 are hyperparameters. In alignment with this generative process, the joint distribution of the edge and latent variable can be written as:

$$p_\theta(\mathbf{v}_i, \mathbf{z}_j, k) = p_\theta(\mathbf{z}_j|\mathbf{v}_i,k)p_\theta(\mathbf{v}_i|k)p(k). \tag{2}$$

Intuitively, $p_\theta(\mathbf{v}_i|k)$ can be viewed as the probability for node representation under the k-th mixture component. Regarding $p_\theta(\mathbf{z}_j|\mathbf{v}_i,k) = \mathcal{N}(\mathbf{z}_j;\boldsymbol{\mu}_{z_j}, \boldsymbol{\Sigma}_j)$, this formulation is inspired by knowledge graph embedding [3], where two entities should be closed to each other under a certain relation operation. However, unlike knowledge graph embedding, the relational context is a latent variable and not observed in our setting. Intuitively, instead of enforcing the exact representation alignment of two linked nodes [24, 12, 15, 14, 44], our design can relax this strong homophily assumption and account for the semantic shift of representations between the central node and its neighbors.

Now, the posterior probability $p_\theta(k|\mathbf{v}_i)$ in Eq. (1) can be derived by using Bayesian rules as follows:

$$p_\theta(k|\mathbf{v}_i) = \frac{p_\theta(\mathbf{v}_i|k)p(k)}{\sum_{k'=1}^{K} p_\theta(\mathbf{v}_i|k')p(k')} = \frac{\mathcal{N}(\mathbf{v}_i;\boldsymbol{\mu}_k, \mathbf{I}\sigma_1^2)p(k)}{\sum_{k'=1}^{K} \mathcal{N}(\mathbf{v}_i;\boldsymbol{\mu}_{k'}, \mathbf{I}\sigma_1^2)p(k')}, \tag{3}$$

where $\boldsymbol{\mu} = \{\boldsymbol{\mu}_k\}_{k=1}^K$ can be treated as a set of trainable prototype representations, which are the additional distribution parameters. This posterior probability represents the soft semantic assignment of the learned representation to the prototypes, which makes the node with similar semantic properties be close to its prototype and encode both the inter- and intra-cluster variation. Substituting $p_\theta(k|\mathbf{v}_i)$ and $p_\theta(\mathbf{z}_j|\mathbf{v}_i, k)$ in Eq. (1) with the specified probabilities, we get final objective $\mathcal{L}_{DSSL}(\theta, \boldsymbol{\mu})$.

3.3 Evidence Lower Bound

Given the framework above, we are interested in: (i) learning the model parameters θ and $\boldsymbol{\mu}$ by maximizing the log-likelihood $\mathcal{L}_{DSSL}(\theta, \boldsymbol{\mu})$, and (ii) inferring the posterior of latent variable k for each observed links. However, it is computationally intractable to directly solve these two problems due to the latent variables. To solve this, we resort to amortized variational inference methods [23], and maximize the evidence lower-bound (ELBO) of Eq. (1), i.e., $\mathcal{L}_{DSSL}(\theta, \boldsymbol{\mu}) \geq \mathcal{L}_{DSSL}(\theta, \phi, \boldsymbol{\mu})$:

$$\mathcal{L}_{DSSL}(\theta, \phi, \boldsymbol{\mu}) = \frac{1}{|N(i)|} \sum_{j \in N(i)} \mathbb{E}_{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)}[\log p_\theta(\mathbf{z}_j|\mathbf{v}_i, k) + \log p_\theta(k|\mathbf{v}_i)] + \mathcal{H}(q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)), \quad (4)$$

where $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$ is the introduced variational distribution parameterized by ϕ and $\mathcal{H}(\cdot)$ is the entropy operator. $\mathbf{z}_j$ is included in this variational posterior, so the inference is also conditioned on the neighborhood information. We derive this ELBO in Appendix A.1. Maximizing this ELBO w.r.t. $\{\theta, \phi, \boldsymbol{\mu}\}$ is equivalent to (i) maximizing $\mathcal{L}_{DSSL}(\theta, \boldsymbol{\mu})$ and to (ii) make variational $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$ be close to true posterior. Plugging the parameterized probabilities into this ELBO, we obtain the following loss to minimize. See Appendix A.2 for corresponding derivations.

$$\begin{aligned}\mathcal{L} = \frac{1}{|N(i)|} \sum_{j \in N(i)} & \mathbb{E}_{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)}\left[\|\mathbf{v}_i + \beta g_\theta(k) - \mathbf{z}_j\|_2^2\right] - \sigma_2^2 \mathbb{E}_{q_\phi(k|\mathbf{v}_i)}\left[\log \frac{\exp(\mathbf{v}_i^\top \cdot \boldsymbol{\mu}_k/\sigma_1^2)}{\sum_{k'=1}^K \exp(\mathbf{v}_i^\top \cdot \boldsymbol{\mu}_{k'}/\sigma_1^2)}\right] \\ & - \mathcal{H}(q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)), \end{aligned} \quad (5)$$

where $q_\phi(k|\mathbf{v}_i) = 1/|N(i)| \sum_{j \in N(i)} q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$ is the posterior probability of semantic assignment for central node v_i, by aggregating all its neighbors. Thus, the first term (denoted as $\mathcal{L}_{local}$) in the loss encourages the model to reconstruct the local neighbors while considering different semantic shifts captured by latent variable k (see Figure 2 (b)). The second term (denoted as $\mathcal{L}_{global}$) encourages the model to perform clustering with learned representation where possible, i.e., seeking to push the representation $\mathbf{v}_i$ to its closest prototype cluster (see Figure 2 (c)). The final entropy term makes the model choose to have high entropy over $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$ such that all of the K-channel losses must be low. Overall, this loss derived from ELBO can capture global semantic similarities over neighborhoods and learn to decouple different latent patterns in the local neighbors.

Regarding the variaitonal distribution $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$, we model it as categorical distribution since k is a discrete multinomial variable. Specifically, the representations $\mathbf{v}_i$ and $\mathbf{z}_j$ are encoded to a combined representation and then $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$ is determined by an output softmax inference head as follows:

$$q_\phi(k|\mathbf{v}_i, \mathbf{z}_j) = \frac{\exp(h_\phi([\mathbf{v}_i; \mathbf{z}_j])[k])}{\sum_{k'=1}^K \exp(h_\phi([\mathbf{v}_i; \mathbf{z}_j])[k'])}, \quad (6)$$

where h_ϕ denotes the inference network parameterized by ϕ and $[\cdot, \cdot]$ can be the element-wise product or concatenation operation. $h_\phi([\mathbf{v}_i; \mathbf{z}_j])[k]$ indicates the kth element, i.e., the logit corresponding the latent context k. Instead of introducing variational parameters individually, we consider the amortization inference [23, 56], which fits a shared network to calculate each local parameter.

For the expectation terms in Eq. (5), back-propagation through the discrete variable k is not directly feasible. We alleviate this by adopting the Straight-Through Gumbel-Softmax estimator [19], which provides a continuous differentiable approximation for drawing samples from a categorical distribution. Specifically, for each sample, a latent cluster vector $\mathbf{c} \in (0, 1)^K$ is drawn from:

$$\mathbf{c}[k] = \frac{\exp((h_\phi([\mathbf{v}_i; \mathbf{z}_j])[k] + \boldsymbol{\epsilon}[k])/\gamma)}{\sum_{k'=1}^K \exp((h_\phi([\mathbf{v}_i; \mathbf{z}_j])[k] + \boldsymbol{\epsilon}[k'])/\gamma)}, \quad (7)$$

where $\boldsymbol{\epsilon}[k]$ is i.i.d drawn from the $\mathrm{Gumbel}(0, 1)$ distribution and γ is a temperature. With this reparameterization trick, we can obtain the surrogate $\mathbb{E}_{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)}[\|\mathbf{v}_i + \beta g_\theta(k) - \mathbf{z}_j\|_2^2] \simeq \mathbb{E}_{\boldsymbol{\epsilon}}[\|\mathbf{v}_i + \beta g_\theta(\mathbf{c}) - \mathbf{z}_j\|_2^2]$ and the gradients are estimated with Monte Carlo. The expectation term over $q_\phi(k|\mathbf{v}_i)$ can be similarly estimated. Then, $\{\theta, \phi, \boldsymbol{\mu}\}$ in Eq. (5) can be efficiently solved by gradient descent.

3.4 Algorithm Optimization

The overall optimization involves simultaneously training (1) the encoder f_θ, (2) the projector g_θ, (3) the inference predictor h_ϕ and (4) the prototypes $\boldsymbol{\mu}$. The most canonical way to update the parameters is stochastic gradient descent. However, we observe that stochastically updating all parameters suffers from two problems: (1) The objective admits trivial solutions, e.g., outputting the same representation for all nodes in the optimization process. (2) Updating prototypes without any constraints will lead to a degenerate solution, i.e., all nodes are assigned to a single cluster.

To address the issue of trivial solutions, inspired by the recent works [17, 11], we consider an asymmetric encoder architecture that includes online and target encoders. Specifically, for each node pair (v_i, v_j), the online encoder f_θ produces the representation of the central node $\mathbf{v}_i$; while the target encoder f_ξ is used to produce the representation of its neighbor $\mathbf{z}_j$. Importantly, the gradient of loss is only used to update the online encoder f_θ while being blocked in the target encoder. The weights of the target encoder ξ are updated via the exponential moving average (EMA) of the online encoder θ:

$$[\theta, \phi, \boldsymbol{\mu}] \leftarrow [\theta, \phi, \boldsymbol{\mu}] - \boldsymbol{\Gamma}\left(\nabla_{\theta,\phi,\boldsymbol{\mu}}\mathcal{L}\right), \quad \xi \leftarrow \tau\xi + (1-\tau)\theta, \tag{8}$$

where $\boldsymbol{\Gamma}(\cdot)$ indicates a stochastic optimizer and $\tau \in [0, 1]$ is the target decay rate. This update introduces an asymmetry between two encoders that prevents collapse to trivial solutions [11, 46]. To alleviate the second issue, besides the stochastic update, we also apply a global update for the prototype vectors $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}_{i=1}^K$ at the end of each training epoch to avoid a degenerate solution:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^N \pi_i(k) \cdot \mathbf{v}_i}{\|\sum_{i=1}^N \pi_i(k) \cdot \mathbf{v}_i\|_2^2}, \text{ where } \pi_i(k) = 1/|N(i)| \sum\nolimits_{j \in N(i)} q_\phi(k|\mathbf{v}_i, \mathbf{z}_j). \tag{9}$$

The derivation is provided in Appendix A.3. Intuitively, $\pi_i(k)$ reflects the degree of relevance of node v_i to the kth prototype. Instead of only updating the prototypes in a mini-batch, we also aggregate all the representations as the prototype based on the soft assignment probability at the end of the epoch. After the training is finished, we only keep the online encoder f_θ for the downstream task. Our full algorithm and network are provided in Appendix B. The details about time complexity of DSSL is given in Appendix C.1, which scales linearly in the size of edges.

3.5 Theoretical Analysis

In this section, we provide an analysis of the proposed framework. We first present the connection between the proposed objective and the mutual information maximization, then show that the learned representations by our objective provably enjoy a good downstream performance. Due to the space limitation, all proofs of theorems and corollaries are provided in Appendix D.

We denote the random variable of the input graph as $\mathcal{G}$ and the downstream label as $\mathbf{y}$. For clarity, we omit subscript i in what follows. From an information-theoretic learning perspective, a desirable way is to maximize the mutual information $I(\mathbf{v}, \mathbf{y})$ between the representation $\mathbf{v}$ and downstream label $\mathbf{y}$. However, due to the lack of the downstream label, self-supervised learning resorts to maximizing $I(\mathbf{v}, \mathbf{s})$ where $\mathbf{s}$ is different designed self-supervised signal [49, 4, 54, 63, 9]. In our method, we have two self-supervised signals: the global semantic cluster information inferred by k and the local structural roles captured by the representations of the neighbors $\mathbf{z} = \{\mathbf{z}_i | v_i \in N(v)\}$ of node v.

Then, we can interpret our objective in Eq. (5) from the information maximization perspective:

Theorem 1. *Optimizing local and global terms in Eq.* (5) *is equivalent to maximizing the mutual information between the representation* $\mathbf{v}$ *and global signal* k *and maximizing the conditional mutual information between* $\mathbf{v}$ *and the local signal* $\mathbf{z}$*, conditioned on global signal* k*. Formally, we have:*

$$\max_{\theta,\phi,\boldsymbol{\mu}} \mathcal{L} \Rightarrow \max_{\mathbf{v}} I(\mathbf{v}; k) + I(\mathbf{v}; \mathbf{z}|k) = I(\mathbf{v}; k, \mathbf{z}). \tag{10}$$

This theorem suggests that we essentially combine both local structure and global semantic information as the self-supervised signal and maximize the mutual information between the representation $\mathbf{v}$ and their joint distribution $(k, \mathbf{z})$. Next, we discuss how the learned representation affects the downstream task $\mathbf{y}$ based on the information bottleneck principle [49, 9, 54, 4]. The rationality of self-supervised learning is that the task-relevant information lies mostly in the shared information between the input and the self-supervised signals [49, 9, 4]. Specifically, we formulate our lightweight and reasonable assumption below, which serves as a foundation for our analysis.

Assumption 1. *Nodes with similar labels should have similar "local structural roles" and "global semantic clusters." In this work, we equate "local structure" with the 1-hop neighborhood and "global*

semantic" with the clustering membership of a node. Formally, we have the task-relevant information $\mathbf{y}$ *left in* $\mathcal{G}$ *except that in joint self-supervised signal* $(\mathbf{z}, k)$ *is relatively small:* $I(\mathcal{G}; \mathbf{y}|\mathbf{z}, k) \leq \epsilon$.

Intuitively, this assumption indicates that most of the task-relevant information in the graph is contained in the self-supervised signal (local neighborhood and global semantic patterns). Based on this assumption, we have the following theorem, which reveals why the downstream tasks can benefit from the learned representations learned by our objective function.

Theorem 2. *Let* $\mathbf{v}_{\text{joint}} = \arg\max_{\mathbf{v}} I(\mathbf{v}; \mathbf{z}, k), \mathbf{v}_{\text{local}} = \arg\max_{\mathbf{v}} I(\mathbf{v}; k)$*, and* $\mathbf{v}_{\text{global}} = \arg\max_{\mathbf{v}} I(\mathbf{v}; \mathbf{z})$*. Formally, we have the following inequalities about the task-relevant information:*

$$I(\mathcal{G}; \mathbf{y}) = \max_{\mathbf{v}} I(\mathbf{v}; \mathbf{y}) \geq I(\mathbf{v}_{\text{joint}}; \mathbf{y}) \geq \max(I(\mathbf{v}_{\text{local}}; \mathbf{y}), I(\mathbf{v}_{\text{global}}; \mathbf{y})) \geq I(\mathcal{G}; \mathbf{y}) - \epsilon. \quad (11)$$

Theorem 2 shows that the gap of task-relevant information between supervised representation $\mathbf{v}_{sup} = \arg\max_{\mathbf{v}} I(\mathbf{v}, \mathbf{y})$ and self-supervised representation $\mathbf{v}_{\text{joint}}$ is ϵ. Thus, we can guarantee a good downstream performance as long as the Assumption 1 is satisfied. It is noteworthy that jointly utilizing local structure and global semantics as the self-supervised signal is expected to contain more task information. As further enlightenment, we can relate Eq. (36) with the Bayes error rate [49]:

Corollary 1. *Suppose that downstream label* $\mathbf{y}$ *is a M-categorical random variable. Then we have the upper bound for the downstream Bayes errors* $P^e_{\mathbf{v}} = \mathbb{E}_{\mathbf{v}}\left[1 - \max_{y \in \mathbf{y}} P(\hat{\mathbf{y}} = y|\mathbf{v})\right]$ *on learned representation* $\mathbf{v}$*, where* $\hat{\mathbf{y}}$ *is the estimation for label from our downstream classifier:*

$$\text{Th}(P^e_{\mathbf{v}_{joint}}) \leq \log 2 + P^e_{\mathbf{v}_{\text{sup}}} \cdot \log M + I(\mathcal{G}; \mathbf{y}|\mathbf{z}, k) \triangleq \text{RHS}_{\mathbf{v}_{joint}}, \quad (12)$$

where $\text{Th}(x) = \min\{\max\{x, 0\}, 1 - 1/|M|\}$ *is a thresholded operation [49]. Similarly, we can obtain the error upper bound of other representations* $\mathbf{v}_{\text{local}}$ *and* $\mathbf{v}_{\text{global}}$: $\text{RHS}_{\mathbf{v}_{local}}$*and* $\text{RHS}_{\mathbf{v}_{global}}$*. Then, we have inequalities on error upper bounds* : $\text{RHS}_{\mathbf{v}_{joint}} \leq \min(\text{RHS}_{\mathbf{v}_{local}}, \text{RHS}_{\mathbf{v}_{global}})$.

This corollary says that our self-supervised signal has a tighter upper bound on the downstream Bayes error. Thus, we can expect that the representation learned by our objective function, which utilizes both local structure and global semantic information, has superior performance on downstream tasks.

4 Experiments

In this section, we empirically evaluate the proposed self-supervised learning method on several real-world graph datasets and analyze its behavior on graphs to gain further insights.

4.1 Experimental Setup

Datasets. We perform experiments on widely-used homophilic graph datasets: Cora, Citeseer, and Pubmed [40], as well as non-homophilic datasets: Texas, Cornell, Wisconsin [35], Penn94 and Twitch. Penn94 and Twitch are two relatively large non-homophilous graph datasets proposed by [39, 27]. We provide the detailed descriptions, statistics, and homophily measures of datasets in Appendix E.1.

Baselines. To evaluate the effectiveness of DSSL, we consider the following representative unsupervised and self-supervised learning methods for the node representation task, including Deepwalk [36], LINE [44], Struc2vec [38], GAE [24], VGAE [24], DGI [51], GraphCL [62], MVGRL [16] and BGRL [46]. The detailed description of baselines and implementations are given in Appendix E.2.

Evaluation Protocol. We consider two types of downstream tasks: node classification and node clustering. For node classification, we follow the standard linear-evaluation protocol on graphs [51, 46], where a linear classifier is trained on top of the frozen representation, and test accuracy (ACC) is used as a proxy for representation quality. For all datasets, we adopt the similar random split with a train/validation/test split ratio of $60\%/20\%/20\%$ for the training of downstream linear classifier following [35, 27]. For the node clustering task, we perform K-means clustering on the obtained representations and set the number of clusters to the number of classes. We utilize the normalized mutual information (NMI) [52] as the evaluation metric for clustering.

Setup. For all self-supervised methods on graph neural networks, we consider a two-layer GCN [25] as the encoder unless otherwise stated, and randomly initialize parameters. We run experiments with 10 random splits and report the average performance. We select the best configuration of hyper-parameters based on accuracy on the validation. The detailed settings are given in Appendix E.2.

Table 1: Experimental results (%) with standard deviations on the node classification task. The best and second best performance under each dataset are marked with boldface and underline, respectively.

Method	Cora	Citeseer	Pubmed	Texas	Cornell	Squirrel	Penn94	Twitch
Deepwalk	77.14±0.82	67.85±0.79	79.38±1.22	42.31±2.21	41.55±3.12	37.54±2.19	56.13±0.46	66.37±0.11
LINE	78.93±0.55	68.79±0.41	80.56±0.92	48.69±1.39	43.68±2.17	38.92±1.58	57.59±0.17	67.23±0.27
Struc2vec	30.26±1.52	53.38±0.62	40.83±1.85	49.31±3.22	30.22±5.87	36.49±1.15	50.29±0.31	63.52±0.35
GAE	78.33±0.27	66.39±0.24	78.28±0.77	53.98±3.22	44.18±3.56	30.53±1.33	58.11±0.18	67.98±0.27
VGAE	80.59±0.35	69.90±0.57	81.33±0.69	50.27±2.21	48.73±4.19	29.13±1.16	58.29±0.21	65.09±0.08
DGI	84.17±1.35	71.80±1.33	81.65±0.71	58.53±2.98	45.33±6.11	26.44±1.12	53.68±0.19	66.97±0.25
GraphCL	84.28±0.91	72.46±1.79	81.96±0.73	48.67±4.37	47.22±4.50	22.53±0.98	58.43±0.31	68.37±0.16
MVGRL	**85.21±1.18**	72.13±1.04	82.33±0.88	51.26±0.38	51.16±1.67	38.43±0.87	57.22±0.17	66.03±0.26
BGRL	83.29±0.72	71.56±0.87	81.34±0.50	52.77±1.98	50.33±2.29	36.22±1.97	58.98±0.13	67.43±0.22
DSSL	83.06±0.53	**73.51±0.64**	**82.98±0.49**	**62.11±1.53**	**53.15±1.28**	**40.51±0.38**	**60.38±0.32**	**69.81±0.17**

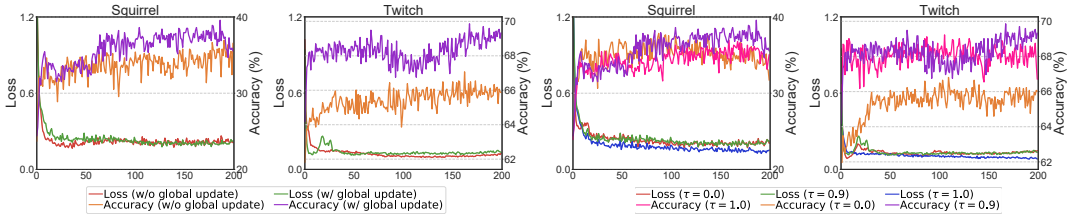


Figure 3: (Left) Performance w/ and w/o global update. (Right) Performance with varying τ.

4.2 Overall Performance Comparison

In this section, we conduct experiments on real-world graphs compared to state-of-the-art methods. Table 1 reports the average classification accuracy with the standard deviation on node classification after ten runs. Since node clustering results have a similar tendency, we provide them in Appendix F.1. From Table 1, we have the following observations: (1) Generally, our DSSL achieves the best performance on both node clustering and classification tasks over the best baseline, demonstrating the effectiveness of DSSL on both homophilous and non-homophilous graphs and the robustness to different downstream tasks and graphs. Especially, DSSL achieves a relative improvement of over 6% and 4% on Texas and Squirrel compared to the best baselines (2) Our DSSL cannot achieve the best performance on Cora. This is reasonable since Cora is highly homophilous (see Appendix E.1), and the core design of augmentation in contrastive learning methods such as MVGRL and GraphCL enables them to be effective on homophilous graphs but failed on low-homophily settings. This supports our motivation in the introduction that it is relatively difficult to design effective graph augmentation on non-homophilous graphs due to their heterogeneous and diverse patterns. (3) When compared with GAE, VGAE, and DGI, DSSL consistently and significantly outperforms them. We deem that this improvement is mainly from the joint local semantic shift and global semantic clustering in DSSL, which is not included in existing methods.

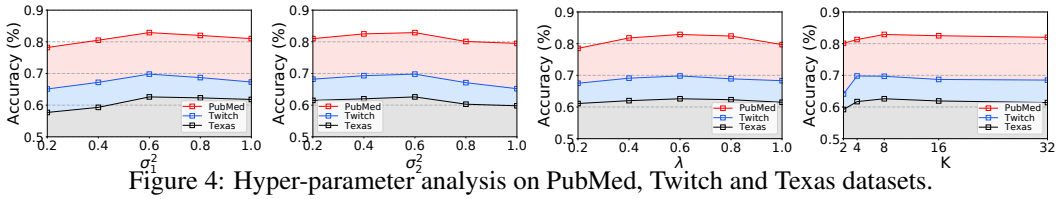


Figure 4: Hyper-parameter analysis on PubMed, Twitch and Texas datasets.

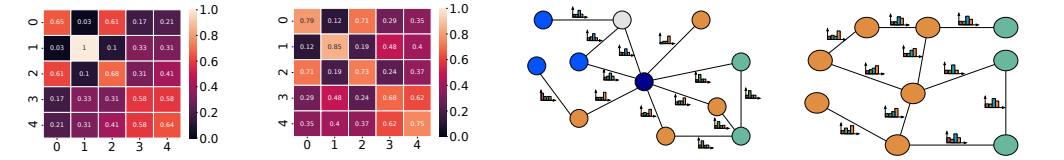
4.3 Ablation Study and Parameter Analysis

In this section, we conduct an ablation study and investigate the sensitivity of hyper-parameters.

Ablation Study. We consider the following ablations: **(A1)** We remove the key component of DSSL: the local reconstruction loss (`w/o` $\mathcal{L}_{local}$). **(A2)** We remove global clustering loss to see if $\mathcal{L}_{local}$ alone is still effective (`w/o` $\mathcal{L}_{global}$). **(A3)** We remove the entropy term in Eq. (5) (`w/o entropy`). **(A4)** We remove the semantic shift term $g_\theta(k)$ in $\mathcal{L}_{local}$ (`w/o semantic shift`). **(A5)** We set posterior $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j) = 1/K$ as uniform distribution for each link (`w/ uniform posterior`). We show the ablation study results in Table 2. $\mathcal{L}_{global}$ alone does not provide much discriminative information, and it does not perform very well. $\mathcal{L}_{local}$ as a key component of DSSL alone produces better results

Table 3: Experimental results (%) on the node classification task with GAT. The best and second best performances under each dataset are marked with boldface and underline, respectively.

Method	Cora	Citeseer	Pubmed	Texas	Cornell	Squirrel	Penn94	Twitch
MVGRL	**84.52±0.95**	71.52±0.41	81.05±0.68	53.42±0.29	50.29±0.96	37.58±0.95	57.21±0.30	67.11±0.23
BGRL	83.91±0.25	72.15±0.42	80.12±0.52	51.02±1.10	48.97±1.03	35.33±1.12	56.52±0.25	66.21±0.33
DSSL	82.54±0.46	**73.67±0.68**	**81.69±0.37**	**59.73±1.28**	**53.63±1.16**	**39.42±1.34**	**58.97±0.26**	**69.55±0.48**



(a) Neighborhood similarity on Texas (b) Latent factor similarity on Texas (c) Visualization of subgraph (Texas id 58) (d) Visualization of subgraph (Twitch id 9)

Figure 5: The visualization and case study results (best viewed on a computer screen and note that the latent distribution of each link need to be zoomed in to be better visible).

than the DGI baseline. We can also observe that considering semantic shift and personalized posterior can further improve the performance a lot which demonstrates our key motivations. The full model (last row) achieves the best performance, which illustrates that different components in the proposed DSSL are complementary to each other.

Effect of Global Update on Prototypes. We conduct ablation studies to gain insights into the global updating of prototypes. As shown in Figure 3, we can observe that without Eq. (9), the performance is not very good, and we are stuck in bad local optima. As discussed above, a possible reason is that we will experience strong degeneracy if we only update prototype vectors with mini-batch training. This figure also demonstrates that DSSL can converge within a few hundred steps, which is efficient.

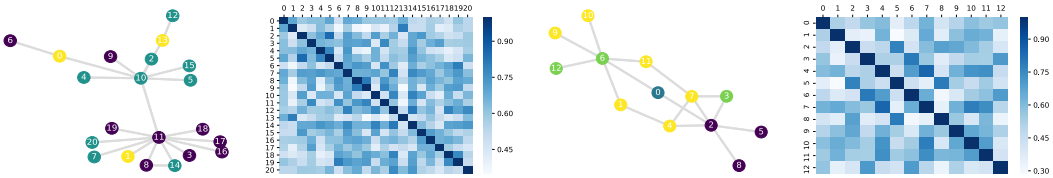
Table 2: Node classification accuracy (%) on the Texas dataset.

Ablation	Accuracy
A1 w/o local loss $\mathcal{L}_{local}$	25.18±1.31
A2 w/o global loss $\mathcal{L}_{global}$	59.34±2.76
A3 w/o entropy loss	60.57±1.27
A4 w/o semantic shift	56.19±1.25
A5 w/ uniform posterior	50.27±1.04
A2+A3	57.61±1.72
A2+A4	50.52±2.01
A2+A5	48.59±1.87
DGI	58.53±2.98
DSSL	**62.11±1.53**

Effect of Asymmetric Architecture. We then explore the effect of target decay rate τ on the performance. Figure 3 shows the learning curves of DSSL on Squirrel and Twitch. We can observe that the best result is achieved at $\tau = 0.9$. When $\tau = 1.0$, i.e., the target network is never updated, DSSL obtains a competitive result but is lower than $\tau = 0.9$. This confirms that slowly updating the target network is crucial in obtaining superior performance. At the other extreme value $\tau = 0$, the target network is the same as the online network, and DSSL demonstrates a degenerated performance.

Hyper-parameters Analysis. We investigate the hyper-parameters most essential to our framework design, i.e., the standard deviation σ_1 and σ_2, the temperature of the Gumbel Softmax γ, and the total number of factors K. The corresponding results are shown in Figure 4. σ_1^2 resembles the temperature scaling in Eq. (5). We observe σ_1^2 is better to be selected from 0.6 to 1.0, and a too small (e.g., 0.2) value may degenerate the performance in all three datasets. σ_2^2 balances the local and global loss in Eq. (5). We can find that having large values of σ_2^2 does not improve the performance, as the local loss plays an essential role in the proposed model. Further, we observe that DSSL is not very sensitive to the Gumbel softmax temperature λ, while a moderate hardness of the Softmax gives the best results. We also find that as K increases from 2 to 8, the performance of DGCL improves, which suggests the importance of decoupling latent factors. However, training with large K will lead to a performance slightly drop. We provide results on other datasets in Appendix F.3.

Encoders Analysis. To further evaluate the effectiveness of the proposed DSSL, we consider the case where the self-supervised learning methods are implemented using another GNN encoder. Table 3 shows the results of the selected baseline with GAT as the encoder. As shown in Table 3, our DSSL performs better than all the compared methods in most cases, which once again proves the effectiveness of DSSL, and also shows that DSSL is broadly applicable to various GNN encoders.



(a) Example graph from Pubmed (b) Corresponding correlation map (c) Example graph from Cornell (d) Corresponding correlation map

Figure 6: Example graphs and correlation maps. The graphs are randomly sampled from Pubmed and Cornell. The correlation maps are obtained based on the pair-wise similarities of learned posteriors.

4.4 Visualization and Case Study

Local neighborhood Patterns. We provide the visualization and case study results to understand how DSSL uncovers the latent patterns in local neighborhoods. More results can be found in Appendix F.4. In Figure 5 (a), we calculate the cross-class neighborhood similarity (see Appendix E.3 for definition), which serves as the ground truth. If nodes with the same label share the same neighborhood distributions, the intra-class similarity should be high. In Figure 5 (b), we calculate average posterior distribution $q_\phi(k|\mathbf{v}_i)$ over all nodes on each class and provide cosine similarity of them. We can observe that our learned distribution shares a similar pattern with the cross-class neighborhood similarity, which shows that learned latent factors can capture the latent semantic information related to neighborhood patterns. In Figures 5 (c) and (d), we plot the subgraph of a randomly selected node, and the distribution $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$. We use different colors to indicate different labels. We find that similar neighbors over class generally have a similar distribution, while different types of links exhibit different latent distributions. This observation matches our motivation that DSSL can decouple the diverse semantics in the local neighborhoods.

Global Semantic Patterns. We investigate how our learned semantic clusters perform on the long-range nodes. To show this, we study the learned cluster distribution $q_\phi(k|\mathbf{v}_i)$ for each node. To facilitate visualization, we randomly sample a sub-graph that contains high-hop neighborhoods from each dataset and use different colors to indicate different node labels. We then calculate the pair-wise similarity of the posteriors $q_\phi(k|\mathbf{v}_i)$ between nodes as shown in Figure 6. We can observe that our learned semantic clusters exhibit similar patterns for the nodes in the same class. Some nodes exhibit similar semantic clusters regardless of the distance between them. For instance, node ID 6 exhibits a very similar posterior distribution to the same class node ID 19 in Pubmed, and node ID 3 exhibits similar posterior distribution to node ID 12 in Cornell, despite these nodes being ID 4 hops away. Since our DSSL seeks to pull the node representation to the inferred semantic clusters, we can learn the global node-node relationships in representations instead of favoring nearby nodes.

5 Conclusions

In this paper, we study the problem of conducting self-supervised learning on non-homophilous graphs data. We present a novel decoupled self-supervised learning (DSSL) framework to decouple the diverse neighborhood context of a node in an unsupervised manner. Specifically, DSSL imitates the generative process of neighbors and explicitly models unobserved factors by latent variables. We show that DSSL can simultaneously capture the global clustering information and the local structure roles with the semantic shift. We theoretically show that DSSL enjoys a good downstream performance. Extensive experiments on several graph datasets validated the superiority of DSSL and showed that DSSL could learn meaningful class-discriminative representations.

Acknowledgments

This work was partially supported by the National Science Foundation (NSF) under grant number IIS-1909702, Army Research Office (ARO) under grant number W911NF-21-1-0198 and Department of Homeland Security under grant number E205949D. The findings and conclusions in this paper do not necessarily reflect the view of the funding agency.

References

[1] Kristen M Altenburger and Johan Ugander. Monophily in social networks introduces similarity among friends-of-friends. *Nature human behaviour*, pages 284–290, 2018.

[2] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3950–3957, 2021.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 2013.

[4] Manh-Ha Bui, Toan Tran, Anh Tran, and Dinh Phung. Exploiting domain-specific features to enhance domain generalization. *Advances in Neural Information Processing Systems*, 2021.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607, 2020.

[6] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2020.

[7] Kaize Ding, Yancheng Wang, Yingzhen Yang, and Huan Liu. Structural and semantic contrastive learning for self-supervised node representation learning. *arXiv preprint arXiv:2202.08480*, 2022.

[8] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

[9] Marco Federici, Anjan Dutta, Patrick Forré, Nate Kushman, and Zeynep Akata. Learning robust representations via multi-view information bottleneck. In *International Conference on Learning Representations*, 2019.

[10] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, pages 21271–21284, 2020.

[12] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[13] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning*, pages 2434–2444, 2019.

[14] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[15] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

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

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

[18] W Hu, B Liu, J Gomes, M Zitnik, P Liang, V Pande, and J Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[19] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[20] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. Universal graph convolutional networks. *Advances in Neural Information Processing Systems*, 2021.

[21] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 148–156, 2021.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

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

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

[26] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 2021.

[27] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34, 2021.

[28] Derek Lim, Xiuyu Li, Felix Hohne, and Ser-Nam Lim. New benchmarks for learning on non-homophilous graphs. *arXiv preprint arXiv:2104.01404*, 2021.

[29] Songtao Liu, Rex Ying, Hanze Dong, Lanqing Li, Tingyang Xu, Yu Rong, Peilin Zhao, Junzhou Huang, and Dinghao Wu. Local augmentation for graph neural networks. In *International Conference on Machine Learning*, pages 14054–14072, 2022.

[30] Yanbei Liu, Xiao Wang, Shu Wu, and Zhitao Xiao. Independence promoted graph disentangled networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4916–4923, 2020.

[31] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *International conference on machine learning*, pages 4212–4221, 2019.

[32] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2021.

[33] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. Simple unsupervised graph representation learning. In *AAAI*, 2022.

[34] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 201–210, 2007.

[35] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2019.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[37] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180, 2019.

[38] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 385–394, 2017.

[39] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, page cnab014, 2021.

[40] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, pages 93–93, 2008.

[41] Fan-Yun Sun, Meng Qu, Jordan Hoffmann, Chin-Wei Huang, and Jian Tang. vgraph: A generative model for joint community detection and node representation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[42] Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In *KDD*, 2021.

[43] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 2021.

[44] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[45] Mingyue Tang, Pan Li, and Carl Yang. Graph auto-encoder via neighborhood wasserstein reconstruction. In *International Conference on Learning Representations*, 2021.

[46] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2021.

[47] MTCAJ Thomas and A Thomas Joy. *Elements of information theory*. Wiley-Interscience, 2006.

[48] Puja Trivedi, Ekdeep Singh Lubana, Yujun Yan, Yaoqing Yang, and Danai Koutra. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. *arXiv preprint arXiv:2111.03220*, 2021.

[49] Yao-Hung Hubert Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Demystifying self-supervised learning: An information-theoretical framework. *arXiv preprint arXiv:2006.05576*, 2020.

[50] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[51] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2018.

[52] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, pages 2837–2854, 2010.

[53] Yanling Wang, Jing Zhang, Haoyang Li, Yuxiao Dong, Hongzhi Yin, Cuiping Li, and Hong Chen. Clusterscl: Cluster-aware supervised contrastive learning on graphs. In *Proceedings of the ACM Web Conference 2022*, pages 1611–1621, 2022.

[54] Yifei Wang, Zhengyang Geng, Feng Jiang, Chuming Li, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Residual relaxation for multi-view representation learning. *Advances in Neural Information Processing Systems*, 34:12104–12115, 2021.

[55] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[56] Teng Xiao and Donglin Wang. A general offline reinforcement learning framework for interactive recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[57] Teng Xiao, Zhengyu Chen, Donglin Wang, and Suhang Wang. Learning how to propagate messages in graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[58] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.

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

[60] Liang Yang, Mengzhe Li, Liyang Liu, Chuan Wang, Xiaochun Cao, Yuanfang Guo, et al. Diverse message passing for attribute with heterophily. *Advances in Neural Information Processing Systems*, 2021.

[61] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. *Advances in Neural Information Processing Systems*, pages 20286–20296, 2020.

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

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

[64] Han Zhao, Xu Yang, Zhenru Wang, Erkun Yang, and Cheng Deng. Graph debiased contrastive learning with joint representation clustering. In *IJCAI*, pages 3434–3440, 2021.

[65] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Exploring edge disentanglement for node classification. *arXiv preprint arXiv:2202.11245*, 2022.

[66] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, pages 7793–7804, 2020.

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

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

[69] Marinka Zitnik, Rok Sosič, and Jure Leskovec. Prioritizing network communities. *Nature communications*, pages 1–9, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A The Omitted Derivations

A.1 The Evidence Lower-Bound

In this section, we provide the details of the lower-bound in Eq. (4). By introducing the approximated posterior $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$, the likelihood $\mathcal{L}_{DSSL}(\theta, \boldsymbol{\mu})$ in Eq. (1) becomes:

$$\begin{aligned}
\mathcal{L}_{DSSL}(\theta, \boldsymbol{\mu}) &= \frac{1}{|N(i)|} \sum_{j \in N(i)} \log \Big[\sum_{k=1}^{K} p_\theta(\mathbf{z}_j|\mathbf{v}_i, k) p_\theta(k|\mathbf{v}_i) \Big] \\
&= \frac{1}{|N(i)|} \sum_{j \in N(i)} \log \Big[\sum_{k=1}^{K} p_\theta(\mathbf{z}_j|\mathbf{v}_i, k) p_\theta(k|\mathbf{v}_i) \frac{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)}{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)} \Big] \\
&\geq \frac{1}{|N(i)|} \sum_{j \in N(i)} \sum_{k=1}^{K} q_\phi(k|\mathbf{v}_i, \mathbf{z}_j) [\log p_\theta(\mathbf{z}_j|\mathbf{v}_i, k) + \log p_\theta(k|\mathbf{v}_i) - \log q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)] \\
&= \frac{1}{|N(i)|} \sum_{j \in N(i)} \mathbb{E}_{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)} [\log p_\theta(\mathbf{z}_j|\mathbf{v}_i, k) + \log p_\theta(k|\mathbf{v}_i) - \log q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)],
\end{aligned} \tag{13}$$

where the third step uses Jensen's inequality.

A.2 The Optimization Loss

In this section, we derive the optimization loss for learning parameters in Eq. (5). Introducing the posterior in Eq. (3) and probabilities specification $p_\theta(\mathbf{z}_j|\mathbf{v}_i, k) = \mathcal{N}(\mathbf{z}_j; \boldsymbol{\mu}_{z_j}, \boldsymbol{\Sigma}_j)$ to the negative of the evidence lower-bounded in Eq. (4), we have:

$$\begin{aligned}
\mathcal{L} = -\frac{1}{|N(i)|} \sum_{j \in N(i)} \mathbb{E}_{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)} [\log \mathcal{N}(\mathbf{z}_j; \boldsymbol{\mu}_{z_j}, \boldsymbol{\Sigma}_j) - \log \frac{\mathcal{N}(\mathbf{v}_i; \boldsymbol{\mu}_k, \mathbf{I}\sigma_1^2) p(k)}{\sum_{k'=1}^{K} \mathcal{N}(\mathbf{v}_i; \boldsymbol{\mu}_{k'}, \mathbf{I}\sigma_1^2) p(k')} \\
+ \log q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)].
\end{aligned} \tag{14}$$

Since we consider isotropic Gaussian with $\boldsymbol{\Sigma}_j = \mathbf{I}\sigma_2^2$, we have:

$$\begin{aligned}
\mathcal{N}(\mathbf{z}_j; \boldsymbol{\mu}_{z_j}, \boldsymbol{\Sigma}_j) &= \frac{1}{(2\pi)^{D'/2} \sigma_2^{D'}} \exp(-\frac{1}{2\sigma_2^2} (\mathbf{z}_j - \mathbf{v}_i - \beta g_\theta(k))^T \cdot (\mathbf{z}_j - \mathbf{v}_i - \beta g_\theta(k))) \\
&= \frac{1}{(2\pi)^{D'/2} \sigma_2^{D'}} \exp(-\frac{\|\mathbf{v}_i + \beta g_\theta(k) - \mathbf{z}_j\|_2^2}{2\sigma_2^2})
\end{aligned} \tag{15}$$

Similarly, for $\mathcal{N}\left(\mathbf{v}_i; \boldsymbol{\mu}_k, \mathbf{I}\sigma_1^2\right)$, we have:

$$\begin{aligned}
\mathcal{N}(\mathbf{v}_i; \boldsymbol{\mu}_k, \mathbf{I}\sigma_1^2) &= \frac{1}{(2\pi)^{D'/2} \sigma_1^{D'}} \exp(-\frac{1}{2\sigma_1^2} (\mathbf{v}_i - \boldsymbol{\mu}_k)^T \cdot (\mathbf{v}_i - \boldsymbol{\mu}_k)) \\
&= \frac{1}{(2\pi)^{D'/2} \sigma_1^{D'}} \exp(-\frac{1}{2\sigma_1^2} (\mathbf{v}_i - \boldsymbol{\mu}_k)^T \cdot (\mathbf{v}_i - \boldsymbol{\mu}_k)) = \frac{1}{(2\pi)^{D'/2} \sigma_1^{D'}} \exp(\frac{(\mathbf{v}_i^\top \cdot \boldsymbol{\mu}_k - 1)}{\sigma_1^2}),
\end{aligned} \tag{16}$$

where the last equivalence holds because we apply L2 normalization to $\mathbf{v}_i$ and $\boldsymbol{\mu}_k$, respectively. Substituting the corresponding terms in Eq. (14) with Eq. (15) and Eq. (17), and removing the constant terms that are irrelevant to model parameters $\{\theta, \phi, \boldsymbol{\mu}\}$, we have:

$$\begin{aligned}
\mathcal{L} = \frac{1}{|N(i)|} \sum_{j \in N(i)} \frac{1}{2\sigma_2^2} \mathbb{E}_{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)} \left[\|\mathbf{v}_i + \beta g_\theta(k) - \mathbf{z}_j\|_2^2 \right] - \mathcal{H}(q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)) \\
- \frac{1}{|N(i)|} \sum_{j \in N(i)} \mathbb{E}_{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)} \Big[\log \frac{\exp\left(\boldsymbol{v}_i^\top \cdot \boldsymbol{\mu}_k / \sigma_1^2\right)}{\sum_{k'=1}^{K} \exp\left(\boldsymbol{v}_i^\top \cdot \boldsymbol{\mu}_{k'} / \sigma_1^2\right)} \Big] = \frac{1}{|N(i)|} \sum_{j \in N(i)} \frac{1}{2\sigma_2^2} \\
\mathbb{E}_{q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)} \left[\|\mathbf{v}_i + \beta g_\theta(k) - \mathbf{z}_j\|_2^2 \right] - \mathbb{E}_{q_\phi(k|\mathbf{v}_i)} [\log \frac{\exp(\boldsymbol{v}_i^\top \cdot \boldsymbol{\mu}_k / \sigma_1^2)}{\sum_{k'=1}^{K} \exp(\boldsymbol{v}_i^\top \cdot \boldsymbol{\mu}_{k'} / \sigma_1^2)}] - \mathcal{H}(q_\phi(k|\mathbf{v}_i, \mathbf{z}_j))
\end{aligned} \tag{17}$$

where $q_\phi(k|\mathbf{v}_i) = 1/|N(i)| \sum_{j \in N(i)} q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$. Multiplying the constant $2\sigma_2^2$ for all terms and absorbing $2\sigma_2^2$ to the hyper-parameter σ_2^2, we obtain the loss function in Eq. (5).

A.3 Derivation of Global Update

In this section, we derive the global analytical update of prototypes. Specifically, we have:

$$\begin{aligned}\boldsymbol{\mu}_k \leftarrow \arg\min_{\boldsymbol{\mu}_k} \sum_{i=1}^{N} \mathcal{L} &= \arg\max_{\boldsymbol{\mu}_k} \sum_{i=1}^{N} \mathbb{E}_{q_\phi(k|\mathbf{v}_i)} \Big[\log \frac{\exp\left(\boldsymbol{v}_i^\top \cdot \boldsymbol{\mu}_k/\sigma_1^2\right)}{\sum_{k'=1}^{K} \exp\left(\boldsymbol{v}_i^\top \cdot \boldsymbol{\mu}_{k'}/\sigma_1^2\right)} \Big] \\ &= \arg\max_{\boldsymbol{\mu}_k} \sum_{i=1}^{N} \mathbb{E}_{q_\phi(k|\mathbf{v}_i)} \Big[\log \exp(\boldsymbol{v}_i^\top \cdot \boldsymbol{\mu}_k/\sigma_1^2) \Big] \\ &= \arg\max_{\boldsymbol{\mu}_k} \sum_{i=1}^{N} \mathbb{E}_{q_\phi(k|\mathbf{v}_i)} \Big[(\boldsymbol{v}_i^\top \cdot \boldsymbol{\mu}_k/\sigma_1^2) \Big]. \end{aligned} \tag{18}$$

In the second line, following [55], we treat the normalizing term as the constant. If we apply L2 normalization to $\boldsymbol{\mu}_k$: $\|\boldsymbol{\mu}_k\| = 1$, we need to solve the Lagrangian of the objective function:

$$\mathcal{L}(\boldsymbol{\mu}_k, \beta) = \max_{\boldsymbol{\mu_k}, \beta} \sum_{i=1}^{N} \mathbb{E}_{q_\phi(k|\mathbf{v}_i)} \Big[(\mathbf{v}_i^\top \cdot \boldsymbol{\mu}_k/\sigma_1^2) \Big] + \beta(1 - \boldsymbol{\mu}_k^\top \boldsymbol{\mu}_k). \tag{19}$$

Take the gradient over $\boldsymbol{\mu}_k$ with respect to $\mathcal{L}(\boldsymbol{\mu}_k, \beta)$ and set it to zero, we have:

$$\begin{aligned} &\sum_{i=1}^{N} q_\phi(k|\mathbf{v}_i)\mathbf{v}_i/\sigma_1^2 - 2\beta \cdot \boldsymbol{\mu}_k = 0 \\ &\Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} q_\phi(k|\mathbf{v}_i) \cdot \mathbf{v}_i}{2\beta\sigma_1^2} = \frac{\sum_{i=1}^{N} \pi_i(k) \cdot \mathbf{v}_i}{2\beta\sigma_1^2}, \end{aligned} \tag{20}$$

where $\pi_i(k) = q_\phi(k|\mathbf{v}_i)$. By taking the gradient with respect to multiplier β and setting it to zero, we have $\boldsymbol{\mu}_k^\top \boldsymbol{\mu}_k = 1$. Combining $\boldsymbol{\mu}_k^\top \boldsymbol{\mu}_k = 1$ and Eq. (20), we have:

$$\beta = \frac{\| \sum_{i=1}^{N} \pi_i(k) \cdot \mathbf{v}_i \|}{2\sigma_1^2}. \tag{21}$$

Further combining the above equation with Eq. (20), we obtain the analytical global update:

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \pi_i(k) \cdot \mathbf{v}_i}{\| \sum_{i=1}^{N} \pi_i(k) \cdot \mathbf{v}_i \|_2^2}, \quad \text{where} \quad \pi_i(k) = 1/|N(i)| \sum\nolimits_{j \in N(i)} q_\phi(k|\mathbf{v}_i, \mathbf{z}_j). \tag{22}$$

Algorithm 1 Training Algorithm of DSSL

Input: $G = (\mathcal{V}, \mathcal{E})$
Output: Online encoder network parameters θ.
Initialize target encoder parameter $\xi = \theta$
repeat
  Randomly select a mini-batch nodes from $\mathcal{V}$
  for *each* v_i *in the batch* **do**
    Randomly sample its neighbors $\mathcal{N}(i)$
    Sample $\mathbf{c}$ with Gumbel softmax
  end
  $\mathcal{L} \leftarrow$ Eq. (5)
  $[\theta, \phi, \boldsymbol{\mu}] \leftarrow [\theta, \phi, \boldsymbol{\mu}] - \boldsymbol{\Gamma}\left(\nabla_{\theta,\phi,\boldsymbol{\mu}} \mathcal{L}\right)$
  Update ξ with momentum moving average: $\xi \leftarrow \tau\xi + (1-\tau)\theta$
  Update $\boldsymbol{\mu}$ at the end of each training epoch with Eq. (9)
until *convergence or reaching max iteration*;

B The Training Algorithm and Network Architecture

B.1 The Training Algorithm of DSSL

The overall training algorithm is shown in Algorithm 1. Concretely, for each iteration of DSSL, we randomly sample the nodes and their neighbors. Then, we calculate the loss function with Gumbel

softmax approximation. All online trainable parameters are updated with stochastic gradient descent (SGD), while those marked with ξ are the target network counterparts to be updated with exponential moving average. We apply a global update for the prototypes $\boldsymbol{\mu}$ at the end of each training epoch.

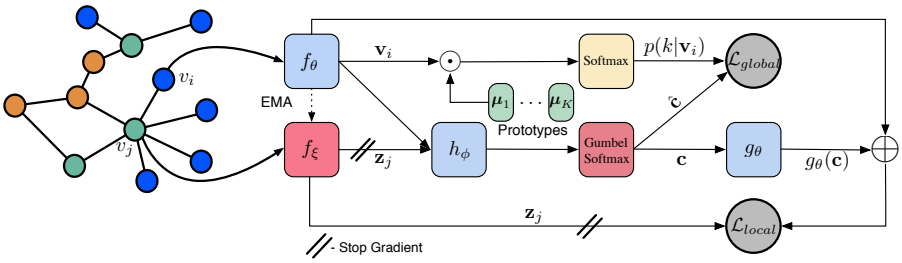


Figure 7: An illustration of the overall DSSL framework. For simplicity, we only illustrate the learning process on one neighbor v_j of central node v_i while it can be applied to all neighbors.

B.2 The Network Architecture of DSSL

The DSSL framework is illustrated in Figure 7. DSSL proposes a self-supervised scheme that learns to encode the node using an online encoder along with a set of prototypes, inference projector g_θ, and predictor h_ϕ. A target encoder is used to produce presentations of neighbors and updated by exponential moving average. We stop gradients on the target encoder coming from the local loss to introduce asymmetry between the online and target encoders.

C Efficiency Analysis

C.1 Time Complexity Analysis

We give the detailed time complexity per iteration of DSSL and compare it with representative methods on node representation learning such as GCA [68] and BGRL [46]. For self-supervised learning, the time complexity mainly depends on two parts: the encoder and loss calculation. Here, we assume the backward pass to be approximately as costly as a forward pass following [46]. Since our DSSL is agnostic to GNN encoders, we consider L-layer GCN as an example. A L-layer GCN [25] with d hidden dimensions has $\mathcal{O}\left(dL|\mathcal{E}| + Nd^2L\right)$ complexity where $\mathcal{E}$ is the number of edges, since it costs $\mathcal{O}(d|\mathcal{E}|)$ to propagate message in each layer, and $\mathcal{O}\left(Nd^2\right)$ to multiply by the weight matrix in each layer. Our DSSL additionally has the inference predictor and projector components in the forward pass, which costs $\mathcal{O}(C_{\text{predictor}}N + C_{\text{projector}}N)$ where $C.$ are constants depending on architecture of the different components. The complexity of DSSL on loss calculation is $\mathcal{O}(KNd) + \mathcal{O}(|\mathcal{E}|d)$ where $\mathcal{O}(KNd)$ and $\mathcal{O}(|\mathcal{E}|d)$ are for global and local losses, respectively. Thus the total time complexity per iteration for DSSL with GCN is $\mathcal{O}\left(dL|\mathcal{E}| + Nd^2L + C_{\text{predictor}}N + C_{\text{projector}}N + KNd + |\mathcal{E}|d\right)$. With the similar notations, we can express the time complexity of GCA as $\mathcal{O}\left(dL|\mathcal{E}| + Nd^2L + C_{\text{projector}}N + N^2d\right)$ and that of BGRL as $\mathcal{O}\left(dL|\mathcal{E}| + Nd^2L + C_{\text{predictor}}N + Nd\right)$. Typically, the architectures for predictor and projector are one-layer MLP for all methods. Thus the $C_{\text{predictor}}$ and $C_{\text{projector}}$ in different methods should be similar. Based on the analysis above, we can find that the time complexity of GCA scales quadratically in the size of the nodes, and both BGRL and our DSSL linearly increase with the number of nodes N when $\mathcal{E}$ is proportional to N. Thus the overall time complexity of our DSSL is at the same level as BGRL [46], which is a large-scale self-supervised learning method proposed recently.

D Theoretical Analysis

D.1 Proof of Theorem 1

To prove Theorem 1, we first present three Lemmas:

Lemma D.1. *For any random variables X, Y, Z and V, we have the following relations:*

$$I(X, Y; Z|V) \geq I(X; Z|V) \tag{23}$$

Proof.

$$
\begin{aligned}
&I(X,Y;Z|V) - I(X;Z|V) \\
&= \iiiint_{XYZV} p(V)p(X,Y,Z|V) \log \frac{p(X,Y,Z|V)}{p(X,Y|V)p(Z|V)} dXdYdZdV \\
&- \iiint_{XZV} p(V)p(X,Z|V) \log \frac{p(X,Z|V)}{p(X|V)p(Z|V)} dXdZdV = \iiiint_{XYZV} P(V)p(X,Y,Z|V) \\
&\log \frac{p(X,Y,Z|V)}{p(X,Y|V)p(Z|V)} - p(V)p(X,Y,Z|V) \log \frac{p(X,Z|V)}{p(X|V)p(Z|V)} dXdYdZdV \\
&= \iiint_{XYZV} p(V)p(X,Y,Z|V) \log \frac{p(X,Y,Z|V)}{p(Y|X,V)p(X,Z|V)} dXdYdZdV = \qquad (24) \\
&\iiint_{XYZV} p(V)p(Y,Z|X,V)p(X|V) \log \frac{p(Y,Z|X,V)}{p(Y|X,V)p(Z|X,V)} dXdYdZdV = I(Y;Z|X,V) \geq 0,
\end{aligned}
$$

which completes the proof of this Lemma. □

Lemma D.2. *For any random variables X and Z, the mutual information $I(X;Z)$ has the following variational lower bound:*

$$
\begin{aligned}
I(X;Z) = H(Z) - H(Z|X) &= \mathbb{E}_{q(Z)}[\log q(Z)] + \mathbb{E}_{q(Z)q(X|Z)}[\log q(Z|X)] \qquad (25) \\
&\geq \max_{\theta} \mathbb{E}_{q(Z)}[\log q(Z)] + \mathbb{E}_{q(Z)q(X|Z)}[\log p_\theta(Z|X)],
\end{aligned}
$$

where $p_\theta(Z|X)$ is an introduced variational discriminator with θ representing the parameters.

Proof.

$$
\begin{aligned}
\mathbb{E}_{q(Z)q(Z|X)}[\log q(Z|X)] &= \max_{\theta} \mathbb{E}_{q(Z)q(X|Z)}[\log p_\theta(Z|X)] + KL(q(Z|X)||p_\theta(Z|X)) \\
&\geq \max_{\theta} \mathbb{E}_{q(Z)q(X|Z)}[\log p_\theta(Z|X)]. \qquad (26)
\end{aligned}
$$

Plugging the inequality above into Eq. (25) completes the proof. □

Lemma D.3. *For any random variables X and Z, the mutual information $I(X,Z)$ has the following contrastive lower bound:*

$$
I(X;Z) \geq \mathbb{E}_{p(X,Z)}[f_\theta(x,z) - \mathbb{E}_{q(\tilde{\mathcal{B}})}[\log \sum_{\tilde{z} \in \tilde{\mathcal{B}}} \exp f_{\boldsymbol{\theta}}(x,\tilde{z})]] + \log |\tilde{\mathcal{B}}| \qquad (27)
$$

where x and z are the sample instances, $f_\theta \in \mathbb{R}$ is a function parameterized by θ (e.g., a dot product between encoded representations of x and z), and $\tilde{\mathcal{B}}$ is a set of samples drawn from a proposal distribution $q(\tilde{\mathcal{B}})$. The set $\tilde{\mathcal{B}}$ contains the one positive sample and $|\tilde{\mathcal{B}}| - 1$ negative samples.

Proof. The proof of this contrastive estimation bound can be founded in [37]. □

Now, we are ready to prove our Theorem 1. We first restate Theorem 1:

Theorem 1. *Optimizing local and global terms in Eq.* (5) *is equivalent to maximizing the mutual information between the representation* $\mathbf{v}$ *and global signal* k *and maximizing the conditional mutual information between* $\mathbf{v}$ *and the local signal* $\mathbf{z}$*, conditioned on global signal* k*. Formally, we have:*

$$
\max_{\theta,\phi,\boldsymbol{\mu}} \mathcal{L} \Rightarrow \max_{\mathbf{v}} I(\mathbf{v};k) + I(\mathbf{v};\mathbf{z}|k) = I(\mathbf{v};k,\mathbf{z}). \qquad (28)
$$

Proof. According to Lemma D.1, for any $v_j \in N(v)$ we have:

$$
I(\mathbf{v};\mathbf{z}|k) = I(\mathbf{v};\mathbf{z}_1,\cdots,\mathbf{z}_{N(v)}|k) \geq I(\mathbf{v};\mathbf{z}_j|k). \qquad (29)
$$

Given the above, we further have the following:

$$
I(\mathbf{v};\mathbf{z}|k) = \frac{1}{|\mathcal{N}(v)|} \sum_{j \in N(v)} I(\mathbf{v};\mathbf{z}|k) \geq \frac{1}{|\mathcal{N}(v)|} \sum_{j \in N(v)} I(\mathbf{v};\mathbf{z}_j|k). \qquad (30)
$$

For $I(\mathbf{v};\mathbf{z}_j|k)$, we can relate it to our local loss $\mathcal{L}_{local}$ in the main body of the paper:

$$\begin{aligned}
\max_{\mathbf{v}} I(\mathbf{v};\mathbf{z}_j|k) &\Leftrightarrow \max_{\mathbf{v}} H(\mathbf{z}_j|k) - H(\mathbf{z}_j|k,\mathbf{v}) \\
&\Leftarrow \min_{\mathbf{v},\theta} -\mathbb{E}_{q(\mathbf{v})q(\mathbf{z}_j)q(k|\mathbf{v},\mathbf{z}_j)}[\log p_\theta(\mathbf{z}_j|k,\mathbf{v})] \\
&\Leftrightarrow \min_{\theta,\phi} -\mathbb{E}_{q_\theta(\mathbf{v})q_\xi(\mathbf{z}_j)q_\phi(k|\mathbf{v},\mathbf{z}_j)}[\log p_\theta(\mathbf{z}_j|k,\mathbf{v})] \\
&\Leftrightarrow \min_{\theta,\phi} -\mathbb{E}_{\mathbf{v}=f_\theta(\mathcal{G})[v],\mathbf{z}_j=f_\xi(\mathcal{G})[j],q_\phi(k|\mathbf{v},\mathbf{z}_j)}[\log p_\theta(\mathbf{z}_j|k,\mathbf{v})] \\
&\Leftrightarrow \min_{\theta,\phi} -\mathbb{E}_{\mathbf{v}=f_\theta(\mathcal{G})[v],\mathbf{z}_j=f_\xi(\mathcal{G})[j],q_\phi(k|\mathbf{v},\mathbf{z}_j)}[\log p_\theta(\mathbf{z}_j|k,\mathbf{v})] \\
&\Leftrightarrow \min_{\theta,\phi} \frac{1}{2\sigma_2^2}\mathbb{E}_{q_\phi(k|\mathbf{v},\mathbf{z}_j)}[\left\|\mathbf{v}+\beta g_\theta(k)-\mathbf{z}_j\right\|_2^2],
\end{aligned} \tag{31}$$

where we use Lemma D.2 in the second line, and $\mathbf{v}$ and $\mathbf{z}_j$ are the representations of node v and node z_j through deterministic encoder f with input $\mathcal{G}=\{\mathbf{X},\hat{\mathbf{A}}\}$. Note that the target encoder ξ is updated via the exponential moving average of the online encoder θ. Combining Eqs. (31) and (30), we have:

$$\min_{\theta,\phi} \frac{1}{\mathcal{N}(v)} \sum_{j\in\mathcal{N}(v)} \frac{1}{2\sigma_2^2}\mathbb{E}_{q_\phi(k|\mathbf{v},\mathbf{z}_j)}[\left\|\mathbf{v}+\beta g_\theta(k)-\mathbf{z}_j\right\|_2^2] \Rightarrow \max_{\mathbf{v}} I(\mathbf{v};\mathbf{z}|k). \tag{32}$$

Our global loss is further related to contrastive estimation. Using Lemma D.4 on $I(\mathbf{v};k)$, we have:

$$I(\mathbf{v};k) \geq \mathbb{E}_{p(\mathbf{v},k)}[f_\theta(\mathbf{v},k) - \mathbb{E}_{q(\tilde{\mathcal{B}})}[\log \sum_{\tilde{k}\in\tilde{\mathcal{B}}} \exp f_{\boldsymbol{\theta}}(\mathbf{v},\tilde{k})] + \log|\tilde{\mathcal{B}}|. \tag{33}$$

When $\tilde{\mathcal{B}}$ always includes all possible values of the latent variable k (i.e., $|\tilde{\mathcal{B}}|=K$) and they are uniformly distributed, and setting $f_\theta(\mathbf{v},k)=\mathbf{v}^\top\cdot\boldsymbol{\mu}_k/\sigma_1^2$, we have:

$$I(\mathbf{v};k) \geq \mathbb{E}_{\mathbf{v}=f_\theta(\mathcal{G})[v],q_\phi(k|\mathbf{v})}[\log \frac{\exp\left(\mathbf{v}^\top\cdot\boldsymbol{\mu}_k/\sigma_1^2\right)}{\sum_{k'=1}^K \exp(\mathbf{v}^\top\cdot\boldsymbol{\mu}_{k'}/\sigma_1^2)}]. \tag{34}$$

Given the above, we have:

$$\max_{\theta,\phi,\boldsymbol{\mu}} \mathbb{E}_{\mathbf{v}=f_\theta(\mathcal{G})[v],q_\phi(k|\mathbf{v})}[\log \frac{\exp\left(\mathbf{v}^\top\cdot\boldsymbol{\mu}_k/\sigma_1^2\right)}{\sum_{k'=1}^K \exp\left(\mathbf{v}^\top\cdot\boldsymbol{\mu}_{k'}/\sigma_1^2\right)}] \Rightarrow \max_{\mathbf{v}} I(\mathbf{v};k). \tag{35}$$

Combining Eqs. (31) and (35), and omitting the entropy term in Eq. (5) completes the proof. □

D.2 Proof of Theorem 2

To prove Theorem 2, we first present the following Lemmas.

Lemma D.4. *For a representation* $\mathbf{v}$ *that is obtained with a deterministic GNN encoder* f_θ *of input graph* $\mathcal{G}$ *with enough capacity, we have the data processing Markov chain:* $(k,\mathbf{z}) \leftrightarrow \mathbf{y} \leftrightarrow \mathcal{G} \rightarrow \mathbf{v}$.

Proof. Since $\mathbf{v}=f_\theta(\mathcal{G})$ is a deterministic function of input graph $\mathcal{G}$, we have the following conditional independence: $(k,\mathbf{z}) \perp \mathbf{v}|\mathcal{G}$ and $\mathbf{y} \perp \mathbf{v}|\mathcal{G}$ [9], which leads to the data processing Markov chain $(k,\mathbf{z}) \leftrightarrow \mathbf{y} \leftrightarrow \mathcal{G} \rightarrow \mathbf{v}$. Thus, the proof is completed. □

Lemma D.5. *By optimizing local and global terms in Eq.* (5)*,* $\mathbf{v}_{joint}$ *is both minimal and sufficient:* $\mathbf{v}_{joint}$ *is sufficient* $\mathbf{v}_{joint}=\arg\max_{\mathbf{v}} I(\mathbf{v};k,\mathbf{z})$*;* $\mathbf{v}_{joint}$ *is minimal* $\mathbf{v}_{joint}=\arg\min_{\mathbf{v}} H(\mathbf{v}|k,\mathbf{z})$.

Proof. We have $I(\mathbf{v};\mathbf{z}|k)=H(\mathbf{v}|k)-H(\mathbf{v}|k,\mathbf{z})$, in which the entropy $H(\mathbf{v}|k)$ is a constant since $p(\mathbf{v}|k)$ is a Gaussian distribution. Thus, maximizing $I(\mathbf{v};\mathbf{z}|k)$ is equivalent to minimize $H(\mathbf{v}|k,\mathbf{z})$. Combining this with Theorem 1, one can directly conclude that $\mathbf{v}_{joint}$ is sufficient and minimal. □

Next, we restate Theorem 2 and provide a complete proof with the help of the Lemmas above.

Theorem 2. *Let* $\mathbf{v}_{\text{joint}}=\arg\max_{\mathbf{v}} I(\mathbf{v};\mathbf{z},k)$, $\mathbf{v}_{\text{local}}=\arg\max_{\mathbf{v}} I(\mathbf{v};k)$*, and* $\mathbf{v}_{\text{global}}=\arg\max_{\mathbf{v}} I(\mathbf{v};\mathbf{z})$*. Formally, we have the following inequalities about the task-relevant information:*

$$I(\mathcal{G};\mathbf{y}) = \max_{\mathbf{v}} I(\mathbf{v};\mathbf{y}) \geq I(\mathbf{v}_{\text{joint}};\mathbf{y}) \geq \max(I(\mathbf{v}_{\text{local}};\mathbf{y}), I(\mathbf{v}_{\text{global}};\mathbf{y})) \geq I(\mathcal{G};\mathbf{y}) - \epsilon. \tag{36}$$

Proof. Based on Lemma D.4, we have the following data processing inequality [47]:

$$I(k, \mathbf{z}; \mathcal{G}) \geq I(k, \mathbf{z}; \mathbf{v}), I(k, \mathbf{z}; \mathcal{G}; \mathbf{y}) \geq I(k, \mathbf{z}; \mathbf{v}; \mathbf{y}), I(\mathcal{G}, \mathbf{y}) \geq I(\mathbf{v}, \mathbf{y}). \tag{37}$$

Since $\mathbf{v}_{\text{joint}} = \arg\max_{\mathbf{v}} I(\mathbf{z}, k; \mathbf{v})$, we have: $I(k, \mathbf{z}; \mathbf{v}_{\text{joint}}) = I(k, \mathbf{z}; \mathcal{G})$ and $I(k, \mathbf{z}; \mathbf{v}_{\text{joint}}; \mathbf{y}) = I(k, \mathbf{z}; \mathcal{G}; \mathbf{y})$. In addition, since $\mathbf{v}_{joint}$ is minimal, we also have, $I(\mathbf{v}_{joint}; \mathbf{y}|k, \mathbf{z}) \leq H(\mathbf{v}_{joint}|k, \mathbf{z}) = 0$. Give the above, we have the following equality:

$$\begin{aligned}
I(\mathbf{v}_{joint}; \mathbf{y}) &= I(\mathbf{v}_{joint}; \mathbf{y}; k, \mathbf{z}) + I(\mathbf{v}_{joint}; \mathbf{y}|k, \mathbf{z}) \\
&= I(\mathcal{G}; \mathbf{y}; k, \mathbf{z}) + I(\mathbf{v}_{joint}; \mathbf{y}|k, \mathbf{z}) \\
&= I(\mathcal{G}; \mathbf{y}; k, \mathbf{z}) + 0 \\
&= I(\mathcal{G}; \mathbf{y}) - I(\mathcal{G}; \mathbf{y}|k, \mathbf{z}) \\
&= \max_{\mathbf{v}} I(\mathbf{v}; \mathbf{y}) - I(\mathcal{G}; \mathbf{y}|k, \mathbf{z}) = I(\mathbf{v}_{\text{sup}}; \mathbf{y}) - I(\mathcal{G}; \mathbf{y}|k, \mathbf{z}).
\end{aligned} \tag{38}$$

Thus, the mutual information gap between self-supervised representation $\mathbf{v}_{joint}$ and supervised representation $\mathbf{v}_{sup}$ is $I(\mathcal{G}; \mathbf{y}|k, \mathbf{z})$. Based on the property of mutual information, we further have:

$$I(\mathcal{G}; \mathbf{y}|k) = I(\mathcal{G}; \mathbf{y}; \mathbf{z}|k) + I(\mathcal{G}; \mathbf{y}|k, \mathbf{z}) \geq I(\mathcal{G}; \mathbf{y}|k, \mathbf{z}). \tag{39}$$

Similarly, we have $I(\mathcal{G}; \mathbf{y}|\mathbf{z}) \geq I(\mathcal{G}; \mathbf{y}|k, \mathbf{z})$. Combining Eqs. (38) and (39), we have the following inequalities about the task-relevant information based on our Assumption 1:

$$I(\mathcal{G}; \mathbf{y}) = \max_{\mathbf{v}} I(\mathbf{v}; \mathbf{y}) \geq I(\mathbf{v}_{\text{joint}}; \mathbf{y}) \geq \max(I(\mathbf{v}_{\text{local}}; \mathbf{y}), I(\mathbf{v}_{\text{global}}; \mathbf{y})) \geq I(\mathcal{G}; \mathbf{y}) - \epsilon, \tag{40}$$

which completes the proof □

D.3 Proof of Corollary 1

Restate Corollary 1:

Corollary 1. *Suppose that downstream label* $\mathbf{y}$ *is a M-categorical random variable. Then we have the upper bound for the downstream Bayes errors* $P^e_{\mathbf{v}} = \mathbb{E}_{\mathbf{v}}\left[1 - \max_{y \in \mathbf{y}} P(\hat{\mathbf{y}} = y|\mathbf{v})\right]$ *on learned representation* $\mathbf{v}$*, where* $\hat{\mathbf{y}}$ *is the estimation for label from our downstream classifier:*

$$\text{Th}(P^e_{\mathbf{v}_{joint}}) \leq \log 2 + P^e_{\mathbf{v}_{\text{sup}}} \cdot \log M + I(\mathcal{G}; \mathbf{y}|\mathbf{z}, k) \triangleq \text{RHS}_{\mathbf{v}_{joint}} \tag{41}$$

where $\text{Th}(x) = \min\{\max\{x, 0\}, 1 - 1/|M|\}$ *is a thresholded operation [49]. Similarly, we can obtain the error upper bound of other representations* $\mathbf{v}_{\text{local}}$ *and* $\mathbf{v}_{\text{global}}$: $\text{RHS}_{\mathbf{v}_{local}}$ *and* $\text{RHS}_{\mathbf{v}_{global}}$*. Then, we have inequalities on error upper bounds* : $\text{RHS}_{\mathbf{v}_{joint}} \leq \min(\text{RHS}_{\mathbf{v}_{local}}, \text{RHS}_{\mathbf{v}_{global}})$.

Proof. To prove this Corollary, we use the following inequalities [47, 49]:

$$\text{Th}(P^e_{\mathbf{v}_{joint}}) \leq -\log(1 - P^e_{\mathbf{v}_{joint}}) \leq H(\mathbf{y}|\mathbf{v}_{\text{joint}}) \tag{42}$$

$$H(\mathbf{y}|\mathbf{v}_{\text{sup}}) \leq \log 2 + P^e_{\mathbf{v}_{\text{sup}}} \log M. \tag{43}$$

For $H(\mathbf{y}|\mathbf{v}_{\text{joint}})$ and $H(\mathbf{y}|\mathbf{v}_{\text{sup}})$, we have the following relations:

$$\begin{aligned}
H(\mathbf{y}|\mathbf{v}_{\text{joint}}) &= H(\mathbf{y}) - I(\mathbf{v}_{\text{joint}}; \mathbf{y}) = H(\mathbf{y}) - I(\mathbf{v}_{\text{sup}}; \mathbf{y}) + I(\mathcal{G}; \mathbf{y}|\mathbf{z}, k) \\
&= H(\mathbf{y}|\mathbf{v}_{\text{sup}}) + I(\mathcal{G}; \mathbf{y}|\mathbf{z}, k),
\end{aligned} \tag{44}$$

where we use Eq. (38) in the second equality. Combining Eqs. (42), (43) and (44), we have:

$$\text{Th}(P^e_{\mathbf{v}_{joint}}) \leq \log 2 + P^e_{\mathbf{v}_{\text{sup}}} \cdot \log M + I(\mathcal{G}; \mathbf{y}|\mathbf{z}, k) \triangleq \text{RHS}_{\mathbf{v}_{joint}}. \tag{45}$$

Further using that $I(\mathcal{G}; \mathbf{y} \mid \mathbf{z}, k) \leq I(\mathcal{G}; \mathbf{y}|\mathbf{z})$ and $I(\mathcal{G}; \mathbf{y}|\mathbf{z}, k) \leq I(\mathcal{G}; \mathbf{y}|k)$ in Eq. (39), we have

$$\text{RHS}_{\mathbf{v}_{joint}} \leq \min(\text{RHS}_{\mathbf{v}_{local}}, \text{RHS}_{\mathbf{v}_{global}}), \tag{46}$$

which completes the proof. □

Table 4: Statistics of used graph datasets in this paper.

Dataset	# Nodes	# Edges	# Classes	# Features	Class-average homophily	Edge homophily
Cora	2,708	5,278	7	1,433	0.766	0.81
Citeseer	3,327	4,552	6	3,703	0.627	0.74
Pubmed	19,717	44,324	3	500	0.664	0.80
Texas	183	309	5	1,793	0.001	0.11
Cornell	183	295	5	1,703	0.047	0.30
Squirrel	5,201	216,933	5	2,089	0.025	0.22
Penn94	41,554	1,362,229	2	5	0.046	0.47
Twitch	9,498	76,569	2	2,545	0.142	0.63

E Experimental Details

E.1 Datasets Description and Statistics

In our experiments, we use the following real-world datasets.

Cora, Citeseerm, and Pubmed are citation and high-homophily graphs, which are among the most widely used benchmarks for semi-supervised node classification [25, 14, 50]. In these graphs, nodes are documents, and edges are citations. Each node is assigned a class label based on the research field. The features of each node are represented by a bag of words of its abstracts.

Texas and Cornell are low-homophily graphs representing links between web pages of the corresponding universities and originally collected by the CMU WebKB projects, where nodes and edges represent web pages and hyperlinks, respectively. We use the pre-processed datasets by [35].

Squirrel is the subgraph of web pages in Wikipedia discussing the related topics, collected by [39]. The Squirrel graph is rather complex, with both homophily and heterophily combined. We also use the pre-processed datasets by [35]

Penn94 is a friendship network from the Facebook 100 networks of university students from 2005 [27], where nodes represent students and are labeled with the reported gender of students. The node features are major, second major/minor, dorm/house, year, and high school.

Twitch is a graph of relationships between accounts on the streaming platform Twitch. Node features are games liked, location, and streaming habits. Nodes are labeled with the explicit language used by a streamer. We utilize the pre-processed sub-graph Twitch-DE [28]. In Twitch-DE, streamers that do not use explicit content (class 0) also often connect to streamers of class 1, which results in an overall non-homophilous structure.

We utilize the edge-homophily [66] and class-average homophily metrics proposed recently by [27] to measure the homophily level of graphs. Specifically, the edge-homophily is the proportion of edges that connect two nodes of the same class and the class-average homophily is defined as:

$$\hat{h} = \frac{1}{C-1} \sum_{k=0}^{C-1} \left[h_k - \frac{|C_k|}{N} \right]_+, \tag{47}$$

where $[a]_+ = \max(a, 0)$ and C is the total number of classes, and h_k is the class-wise homophily:

$$h_k = \frac{\sum_{u \in C_k} d_u^{(k_u)}}{\sum_{u \in C_k} d_u}, \tag{48}$$

in which d_u is the number of neighbors of node u, and $d_u^{(k_u)}$ is the number of neighbors of u that have the same class label. Compared to the edge homophily metirc [66], this metric is less sensitive to the number of classes and the size of each class. The statistics of datasets are given in Table 4. We can find that the widely used citation graphs Cora, Citeseer, and Pubmed are highly homophilous, and other graphs show low homophily degrees and thus are non-homophilous.

E.2 Baselines and Setup

Deepwalk [36] is a network embedding method that generates random walks in the graph and then learns latent representations of nodes by treating walks as the equivalent of sentences with SkipGram.

LINE [44] is an embedding model that preserves both the first-order and second-order proximities.

Struc2vec [38] is the method for node representations by capturing the node structural identity.

GAE [24] is a GCN encoder trained by reconstructing adjacency matrix.

VGAE [24] is a probabilistic version of GAE and is trained by reconstructing the adjacency matrix.

DGI [51] is an unsupervised node representation method that maximizes the mutual information between node representations and graph summary.

GraphCL [62] learns node representations by contrasting different augmentations of graphs.

MVGRL [16] is a learning method that contrasts encoders from neighbors and a graph diffusion.

BGRL [46] is a graph representation learning method that learns node representations by predicting alternative augmentations of the graph.

We used the official implementations released by the authors for all baselines. We ran our experiments on GeForce RTX 2080 Ti. In all our experiments, we use the Adam optimizer [22]. The projector g_θ and predictor h_ϕ are both the Multilayer Perceptron (MLP) with a single hidden layer. We also used techniques like batch normalization and negative sampling on Penn94 and Twitch datasets. We use the same dataset splits and training procedure for all methods. We tune hyper-parameters for all models individually based on accuracy on the validation set and randomly initialize the model parameters. For all methods, the hyper-parameter search spaces are as follows: learning rate $\{0.001, 0.005, 0.01\}$, representation dimension $\{16, 32, 64\}$, L2 weight-decay $\{5e{-}4, 1e{-}4, 5e{-}6, 1e{-}6\}$. For our DSSL, we tune the following hyper-parameters: $\sigma_1^2 \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, $\sigma_2^2 \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, $\lambda \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ and $K \in \{2, 4, 6, 8, 16, 32\}$.

E.3 Cross-Class Neighborhood Similarity

To further demonstrate how DSSL uncovers the latent patterns in neighborhood distributions, we utilize the cross-class neighborhood similarity [32] to serves as the ground-truth for our case study experiments in section 4.4.

Definition 1. *(Cross-Class Neighborhood Similarity). Given a graph G and node labels, the cross-class neighborhood similarity between two classes c, c' is given by $s\left(c, c'\right) = \frac{1}{|\mathcal{V}_c||\mathcal{V}_{c'}|}\sum_{i\in\mathcal{V}_c, j\in\mathcal{V}_{c'}} \cos(d(i), d(j))$ where $\mathcal{V}_c$ indicates the set of nodes in class c and $d(i)$ denotes the empirical histogram (over $|\mathcal{C}|$ classes) of node i's neighbors' labels, and the function $\cos(\cdot,\cdot)$ measures the cosine similarity.*

This cross-class neighborhood similarity measures the neighborhood distributions between difference classes. If neighborhood patterns of different label nodes are different, the inter-class similarity will be low. In contrast, if nodes with the same label share the same neighborhood distributions, the intra-class similarity should be high.

F Additional Experimental Results

F.1 Node Clustering Results

We provide the node clustering results in Table 5. As shown in Table 5, we can find that our DSSL can consistently improve the node clustering performance compared to the baselines on all datasets except the Cora. This observation once again verifies the effectiveness of DSSL in learning generalizable node representations. In addition, along with the node classification performance, these results show that DSSL can provide transferable and robust node representations for various downstream tasks.

F.2 Additional Results on Synthetic Dataset

In this section, we generate the synthetic graphs with various homophily ratios h to evaluate DSSL. Specifically, we consider the synthetic dataset: syn-products generated by [66]. We refer the interested reader to [66] for the detailed generation process. The statistics for synthetic dataset syn-products are provided in Table 6. Roughly speaking, the synthetic graphs can be generated by controlling various edge-homophily ratios h where $h = 1$ indicts strong edge-homophily, and

Table 5: Experimental results (%) with standard deviations on the node clustering task. The best and second best performance under each dataset are marked with boldface and underline, respectively.

Method	Cora	Citeseer	Pubmed	Texas	Cornell	Squirrel	Penn94	Twitch
GAE	37.43±0.05	32.45±0.05	34.38±0.06	26.97±0.11	15.39±0.13	11.73±0.08	16.70±0.15	27.45±0.17
VGAE	38.92±0.08	36.49±0.03	41.09±0.04	27.75±0.16	17.87±0.13	10.83±0.09	17.34±0.08	25.89±0.08
DGI	45.17±0.08	42.03±0.08	45.33±0.02	34.17±0.07	15.92±0.15	8.49±0.13	12.14±0.19	25.84±0.18
GraphCL	46.29±0.03	46.38±0.12	46.78±0.03	30.25±0.13	16.86±0.17	7.97±0.13	16.35±0.13	27.55±0.16
MVGRL	**48.65±0.07**	45.33±0.02	48.81±0.21	32.72±0.18	18.02±0.11	17.65±0.08	15.80±0.17	24.53±0.16
BGRL	47.35±0.03	44.78±0.04	47.21±0.07	33.59±0.15	19.74±0.14	15.13±0.09	17.49±0.13	25.69±0.09
DSSL	46.77±0.04	**47.89±0.04**	**49.39±0.02**	**38.22±0.15**	**20.36±0.08**	**19.85±0.13**	**18.66±0.15**	**29.63±0.08**

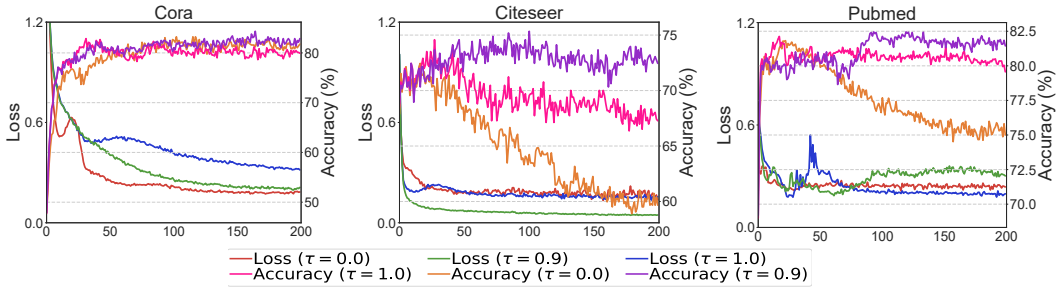


Figure 8: Performance on Cora, Citeseer and Pubmed with varying τ.

$h = 0$ indicts perfect heterophily of generated graph. Here, we consider six edge-homophily ratios, $h = [0.0, 0.25, 0.50, 0.75, 1.0]$. We randomly split 25% of nodes into the training set, 25% into validation, and 50% into the test set for the downstream linear classifier. Table 7 gives the results on syn-products dataset. From this table, we have the following observations: (1) GAE achieves the best accuracy under strong homophily ($h = 1$), but it is significantly less accurate than other methods in low-homophily settings. This is reasonable since GAE over-emphasizes proximity and the homophily assumption. (2) Our DSSL outperforms recent contrastive learning methods such as DGI and MVGRL, which shows the effectiveness of DSSL on non-homophily scenarios, and suggests that designing effective graph augmentation for those contrastive baselines on the non-homophilous graphs is difficult due to the heterogeneous and diverse neighbor pattern. (3) Our DSSL can generally achieve the best performance and effectively adapt to all levels of homophily.

Table 6: Statistics of the synthetic dataset: syn-products.

Dataset	# Nodes	# Edges	# Classes	# Features	Edge homophily
syn-products	10000	59,640 to 59,648	10	100	[0.0, 0.25, 0.50, 0.75, 1.0]

F.3 Additional Results on Ablation Study and Parameter Analysis

Ablation Study. We further conduct experiments to study the effects of the global update on prototypes and asymmetric architecture on homophily datasets. Figures 8 and 9 show the ablation results on Cora, Citeseer and Pubmed. These results also echo the collapse issue as discussed in the main text: without the global update and asymmetric architecture, there is a significant performance drop on these three homophily datasets.

Parameter Analysis. We provide more results on the parameter analysis. Figure 10 shows the results on Citeseer, Cornell, and Squirrel datasets. We can observe that having large values of σ_2^2 also does not improve the overall performance, and DSSL is not very sensitive to the Gumbel softmax temperature λ for these three datasets. We also find that as K increases from 2 to 8 (2 to 16 for Citeseer), the accuracy of DSSL improves.

Table 7: Experimental results (%) on synthetic datasets, syn-products, measured in terms of node classification accuracy along with standard deviations. The best and second best performances under each dataset are marked with boldfac and underline, respectively.

Homophily ratio h	**0.00**	**0.25**	**0.50**	**0.75**	**1.00**
GAE	18.28±0.52	25.49±0.37	50.62±0.28	82.05±0.38	**95.27±0.11**
DGI	40.21±0.43	42.38±0.26	57.31±0.36	85.21±0.43	92.12±0.43
MVGRL	42.69±0.32	44.35±0.40	59.43±0.27	86.65±0.39	93.11±0.35
BGRL	45.31±0.47	47.52±0.38	61.37±0.26	86.21±0.24	93.28±0.37
DSSL	**49.19±0.62**	**51.35±0.51**	**64.78±0.25**	**87.59±0.46**	94.83±0.55

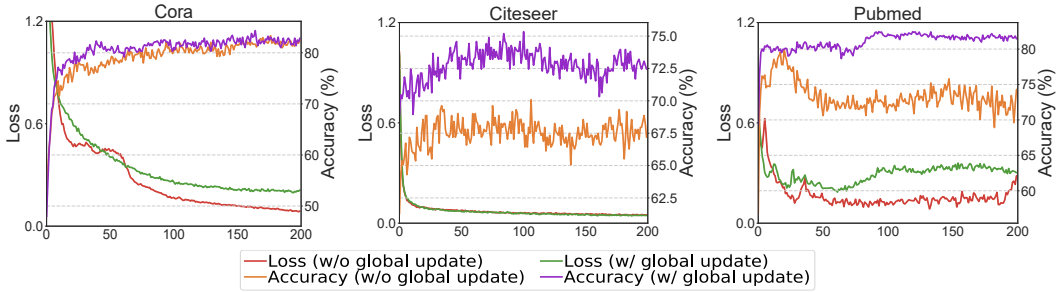


Figure 9: Performance on Cora, Citeseer and Pubmed w/ and w/o global update.

F.4 Additional Results on Visualization and Case Study

To further examine the learned latent factors, in Figure 11, we provide the additional case study and visualization results on other datasets. We can find that, in most cases, the latent factors share a similar distribution for the same type of link, which illustrates its effectiveness in decoupling the underlying latent semantic meaning of different neighbors. This also interprets the reason why capturing the semantic shift can improve performance. Figure 12 shows the results of long-range nodes on Cora and Texas datasets. We can also find that some nodes exhibit similar semantic clusters, regardless of the distance between them. These additional qualitative visualization results, along with our other experiment results, suggests that considering the semantic shift between local neighbors and capturing global semantics may be necessary for non-homophilous graphs.

G Discussion and Broader impact

In this paper, we proposed a new framework called decoupled self-supervised learning (DSSL) for unsupervised node representation learning on non-homophilous graphs. The theoretical analysis and extensive experiments show the effectiveness of our proposed DSSL.

Limitation of the work. Despite the theoretical grounds and the promising experimental justifications, there is one limitation of the current work, which we hope to improve in future work: our theoretic analysis requires some assumptions on the relationship between the downstream labels and the neighborhood distribution. While, not surprisingly, we have to make some assumptions to expect good generalization for the unsupervised node representation learning, it is an interesting future direction to explore more relaxed assumptions than the ones used in this work.

Potential negative societal impacts. In this work, we propose a self-supervised learning framework for node representation learning which does not rely on annotated labels, which might reduce the need for label annotation and thus makes a few individuals who focus on labeling or annotating data

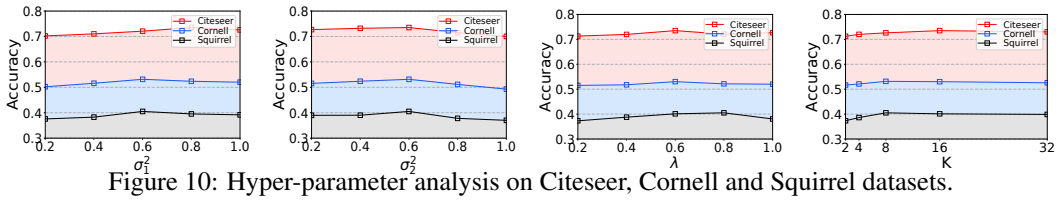
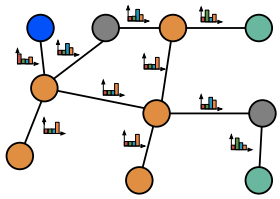
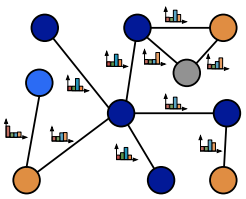


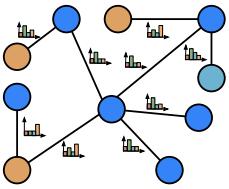
Figure 10: Hyper-parameter analysis on Citeseer, Cornell and Squirrel datasets.



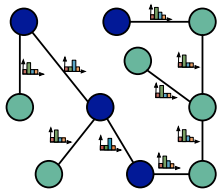
(a) Visualization of subgraph (Cora id 6)



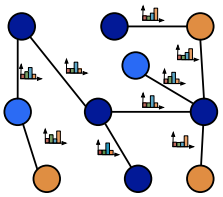
(b) Visualization of subgraph (CiteSeer id 1)



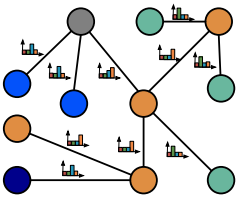
(c) Visualization of subgraph (Pubmed id 0)



(d) Visualization of subgraph (Penn94 id 19)

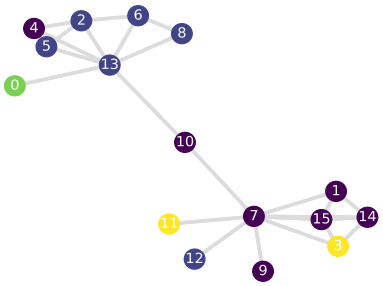


(e) Visualization of subgraph (Squirrel id 11)

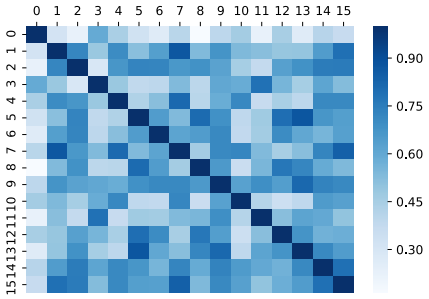


(f) Visualization of subgraph (Cornell id 8)

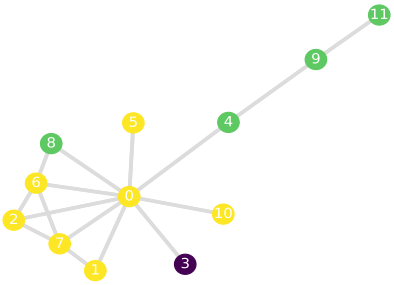
Figure 11: The visualization and case study results on other datasets. (best viewed on a computer screen and note that the latent distribution of each link need to be zoomed in to be better visible).



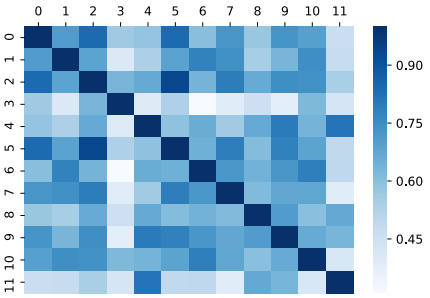
(a) Example graph from Cora



(b) Corresponding correlation map from Cora



(c) Example graph from Texas



(d) Corresponding correlation map from Texas

Figure 12: Example graphs and correlation maps. The graphs are randomly sampled from Cora and Texas. The correlation maps are obtained based on the pair-wise similarity of learned posteriors.

unemployed. In addition, our learned representation may suffer from malicious adversaries who seek to extract node-level information about sensitive attributes. Thus, improving the robustness of DSSL is an interesting future work direction.