# Decoupled Self-supervised Learning for Non-Homophilous Graphs

**Anonymous authors**

## Abstract

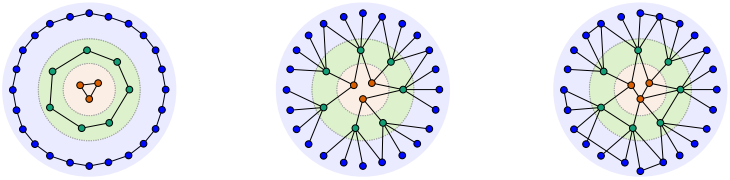
In this paper, we study the problem of conducting self-supervised learning for node representation learning on non-homophilous graphs. Existing self-supervised learning methods typically assume the graph is homophilous where linked nodes often belong to the same class or have similar features. However, such assumptions of homophily do not always hold true 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 node 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 with latent variables, 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 better downstream performance. Extensive experiments on various types of graph benchmarks demonstrate that our proposed framework can significantly achieve better performance compared with competitive self-supervised learning baselines.

## 1 Introduction

Graph-structured data is ubiquitous in real world such as social networks, knowledge graphs, and molecular structures. In recent years, graph neural networks (GNNs) [13, 23, 41, 45] have been proven to be powerful in node representation learning over graph-structured data and have achieved state-of-the-art performance across various tasks such as node classification. 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 for annotation, such as medicine and chemistry [55, 17]. Moreover, supervised learning with labeled data may suffer from problems of less-transferrable, over-fitting and poor generalization particularly when the task-specific labels are scarce [7, 48].

Recently, self-supervised learning (SSL) provides a promising learning paradigm that reduces the dependence on manual labels in the image domain [5, 9, 10, 16]. Compared to image data, there are unique challenges of 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 [23]. A number of recent works [42, 15, 48, 54, 38, 50] 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 [42, 48, 36]. The first scheme is reconstructing the vertex adjacency following traditional network-embedding methods [24, 11, 14, 13, 37], 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 [42, 37]. However, this assumption over-emphasizes proximity [42, 48] and does not hold true for heterophlic and non-homophilous (mixing) graphs where linked nodes may have dissimilar features and different class labels. In comparison, contrastive learning methods [42, 15, 48, 54, 38] constructs two graph 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

Preprint. Under review.



(a) Graph with homophilous pattern. (b) Graph with heterophlic pattern. (c) Graph with mixing pattern.

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

heavily relies on the topology augmentation and negative sample mining [53, 39]. Importantly, conducting augmentation and negative sample mining for non-homophilous or even extreme heterophlic graphs is difficult since linked nodes may be dissimilar and nodes with high similarities might be farther away from each other. As shown in our experiments in Table 1, most of contrastive learning methods fail to generalize to non-homophilous graphs. Although there are many efforts [30, 52, 26, 54, 6, 46] trying to deal with the non-homophilous graph by designing new GNN models, they are still supervised learning approaches and heavily rely on the annotated labels, which limits their applications where labeled data is expensive and unavailable. 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 investigating whether one can take advantage of neighborhood strategies of nodes to help the self-supervised learning for 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. For instance, while the gender prediction in common dating networks lacks homophily (individual with gender male has a preference of making friends with people whose identities are female) [1], neighborhood distribution are very informative to the node gender labels, i.e., nodes with similar neighborhoods are likely to be similar. Figure 1 illustrates this motivation.

While the motivation is straightforward, there are two challenges to achieve it. The first challenge is how to capture the local neighborhood distribution in a self-supervised learning manner. The neighborhood distributions typically follow heterogeneous and diverse patterns. For instance, as shown in Figure 2 (c), a node usually connects with others due to the complex interaction of many latent factors, and therefore possesses distribution consisting of local mixing patterns wherein certain parts of the neighborhood are homophilous while others non-homophilous. Since there is no way to directly access the latent strategies of neighborhoods, the lack of supervision obstructs us from modeling the distribution of neighborhood. The second challenge is capturing the long-range semantic dependencies in self-supervised learning objectives. 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, due to its favorable performance in downstream task.

This paper presents a new self-supervised framework: the decoupled self-supervised learning (DSSL), which attempts to achieve a good balance among these two challenges. At the core of DSSL is the latent variables, which empower model 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 unified DSSL framework 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 including large-scale ones 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 [29], dating networks [1] and molecular networks [52]. Recently, various GNNs have been proposed to deal with non-homophilous graphs with different methods such as potential neighbor discovery [30, 21, 19], adaptive message propagation [6, 44], exploring high-frequency signals [2] and higher-order message passing [52, 5]. Despite their success, they typically consider the semi-supervised setting and are trained

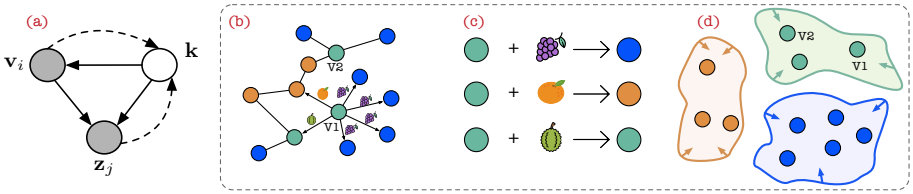



Figure 2: An illustration of (a) graphical model for DSSL. The categorical latent variable $k$ is used to instantiate a latent mixture-of-Gaussians $\mathbf{v}_i$, which is then decoded to $\mathbf{z}_j$ and (b) a toy subgraph example of non-homophilous graphs. The motivation behind this work. (c) Our model encodes the semantic shift in the process of decoupling local structure and (d) captures global semantic structure by semantic clusters.

with task-specific labeled data; while in practice, labels are often limited, expensive, and even inaccessible. In contrast, in this paper, we study the problem of self-supervised learning: learning generalizable and transferable node representations without relying on labels for non-homophilous graphs.

**Self-supervised Learning on Graphs.** Self-supervised learning holds great promise for improving representations when labeled data are scarce [5, 9, 10, 16]. In graph domain, earlier combinations of GNNs and self-supervised learning involve GraphSAGE [13], VGAE [24] and Graphite [12], which typically follow the traditional network-embedding methods [32, 37, 11] and adopt the link reconstruction or random walk principle. Since these methods over-emphasize node proximity at the cost of structural information [42, 36, 48], various graph contrastive learning methods have been proposed [42, 15, 48, 54, 38], which aim to learn representations by contrasting representation under differently augmented views and have achieve promising performance. However, they heavily rely on complex data- or task-specific augmentations, which are prohibitively difficult for non-homophilous graphs. The standard augmentation scheme usually have limited performance for non-homophilous graphs as shown in our experiments. There are also some recent works trying to design heuristic pretext tasks [49, 20, 17] on graphs, however, for non-homophilous graphs, building hand-crafted pretext tasks for each node tends to be infeasible when we know little knowledge underlying graph or the node has diverse context. Our work differs from the above methods and aims to answer the question of how to design 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 disentangle latent factors in the graph. There are a couple of works that explore the disentangled factors in the node-level [28, 27], edge-level [51] and graph-level [47]. Whereas, these methods require task-specific labels that can be extremely scarce for graph datasets. By contrast, we address the problem of learning generalizable and transferrable node representations on non-homophilous graphs without labels. A recent work called disentangled contrastive learning [25] learns disentangled graph representations without labeled graph. Nevertheless, they are still based on contrastive learning which requires prefabricated augmentations. By contrast, our framework is a generative model and does not rely on graph augmentations. Moreover, their goal is to conduct graph-level classification tasks which are different from ours. We instead tackle a node-level representation problem on non-homophilous graphs where connected nodes may not be similar to each other and we decouple the diverse neighborhood context of a node with a generative process.

# 3 Decoupled Self-supervised Learning

In this section, we describe our problem setting with respect to self-supervised node representation, 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 attribute matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$ is used to describe node features. The $i$-th row of $\mathbf{X}$, i.e., $\mathbf{x}_i$, is the feature vector 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 $\theta$ denotes the set of its parameters, such that the representation of node $v_i$, i.e., $f_\theta(\mathbf{X}, \mathbf{A})[i]$, 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 (DSSL), which can learn meaningful node representations for non-homophilous graphs by capturing its intrinsic graph structure. The

core idea of DSSL is to model the distributions of node neighbors via a mixture generative process in the representation learning scenario. Specifically, we model the generation of node 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 $\theta$ 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 observed neighbors for node $\mathbf{v}_i$ 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 [32, 37, 11, 24], which directly encourages 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, so that the model is more generalizable for 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 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 $\pi_k$ be its corresponding mixture probability. The generation of a node and its neighbor 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: $\boldsymbol{\mu}_{z_j} = \mathbf{v}_i + g_\theta(k)$. Here the projector $g_\theta(\cdot)$ denotes another single fully connected network which embeds latent variable $k$ (one-hot vector) to the representation space. In practice, to reduce complexity, we consider using isotropic Gaussian with tied variance, 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, $\sigma_1$ and $\sigma_2$ are hyperparameters. In alignment with this generative process, the joint distribution of observed links and the 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 from knowledge graph embedding [3], where two entities should be close to each other under a certain relation operation. However, unlike knowledge graph embedding, the relation context is latent variable and not observed in our unsupervised setting. Intuitively, instead of enforcing exact representation alignment of two linked nodes [24, 11, 14, 13, 37], our design can relax this exact homophily assumption and account for the semantic shift of representations between the central node and its neighbors via the inferred latent context $k$.

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 cluster prototype, which can make 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 our DSSL induced objective $\mathcal{L}_{DSSL}(\theta,\boldsymbol{\mu})$.

### 3.3 Evidence Lower Bound

Given the probabilistic framework above, we are interested in: (i) learning the model parameters $\theta$ 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 [22], and maximize the evidence lower-bound (ELBO) of Eq. (1), i.e., $\mathcal{L}_{DSSL}(\theta, \boldsymbol{\mu}) \geq \mathcal{L}_{DSSL}(\theta, \phi, \boldsymbol{\mu})$:

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

where $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$ is the introduced variational distribution parameterized by $\phi$ and $\mathcal{H}(\cdot)$ is the entropy operator. $\mathbf{z}_j$ is included in this variational posterior so that 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) maximize $\mathcal{L}_{DSSL}(\theta, \boldsymbol{\mu})$ 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):

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

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 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:

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

where $h_\phi$ denotes the inference predictor network parameterized by $\phi$ and $[\cdot, \cdot]$ denotes the concatenation operation. $h_\phi([\mathbf{v}_i; \mathbf{z}_j])[k]$ indicates the $k^{th}$ element, i.e., the logit corresponding the latent context $k$. Instead of introducing variational parameters individually for each link, we consider the amortization inference, which fits a shared network to calculate each local variational parameter and is more efficient.

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 [18], 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 a reparameterization distribution:

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

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

### 3.4 Algorithm Optimization

Recall that the overall optimization steps involves simultaneously training (1) the encoder $f_\theta$, (2) the projector $g_\theta$, (3) the inference predictor $h_\phi$ and (4) the prototype vectors $\boldsymbol{\mu}$. 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) Since we do not rely on negative samples, the objective admits trivial solutions, e.g., outputting the same representation for all nodes. (2) Updating prototype vectors without any constraints will lead to a degenerate solution, i.e., all nodes are assigned to a single (arbitrary) cluster.

To address the issue of trivial solutions, inspired by the recent works [16, 10], we consider an asymmetric encoder architecture which includes online and target encoders. Specifically, for each node pair $(v_i, v_j)$, the online encoder $f_\theta$ produces the representation of the central node $\mathbf{v}_i$; while the target encoder $f_\xi$ 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 $f_\theta$, while being blocked in the target encoder. The weights of the target encoder $\xi$ are instead undated via exponential moving average (EMA) of online encoder $\theta$:

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

where $\boldsymbol{\Gamma}(\cdot)$ indicates a stochastic gradient descent (SGD) 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 [10, 38]. To alleviate the second issue, besides the stochastic update, we also apply an analytical 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 detailed derivation is provided in Appendix A.3. Intuitively, $\pi_i(k)$ reflects the degree of relevance of node $v_i$ to the $k^{th}$ prototype. Instead of only updating the prototype vectors in a mini-batch, we also aggregate all the representation as the prototype based on the soft assignment probability at the end of epoch.

After the training is finished, we only keep the online encoder $f_\theta$ for the downstream task. Our full algorithm and network illustration are provided in Appendix B. The time complexity of our algorithm is $O(N \cdot d)$ where $d$ is the average node degree. Note that we omit the complexity introduced by the GNN encoder and dot-product computation, as it is orthogonal to the SSL design. Compared to the recent graph SSL approaches [31, 42, 54, 48, 15] which requires computation and memory quadratic in the number of nodes, our method scales linearly in the size of nodes, thus is trained in a more efficient way.

### 3.5 Theoretical Analysis

In this section, we provide theoretical 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 good downstream performance. Due to the space limitation, all proofs of theorems and corollaries are provided in Appendix C.

We denote the random variable of the input graph as $\mathbf{g}$ and the downstream label as $\mathbf{y}$. For clarity, we omit subscript $i$ in what follows. Note that downstream tasks could be the node classification, regression, or clustering. From an information-theoretic learning perspective, a desirable way is to maximize the mutual information $I(\mathbf{v}, \mathbf{y})$ between the node representation $\mathbf{v}$ and its 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 the designed self-supervised signal for different methods [40, 4, 50, 8]. In our method, we actually has two self-supervised signal: the global semantic cluster information inferred by $k$ and the local structural roles captured by the distributions of 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 Eq.* (5) *is equivalent to maximize the mutual information between the node representation* $\mathbf{v}$ *and the global semantic signal* $k$*, and maximize the conditional mutual information between* $\mathbf{v}$ *and the local signal* $\mathbf{z}$*, conditioned on the global signal* $k$*. Formally, we have the following relations:*

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

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 affect the downstream task $\mathbf{y}$ based on the information bottleneck principle [40, 8, 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 [40, 8, 4]. Specifically, we formulate our lightweight and reasonable assumption below, which serves as a foundation for analysis on downstream task performance.

**Assumption 1.** *Nodes with similar labels should have similar "local structural roles" and "global semantic clusters" in the non-homophilous graph. 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* $\mathbf{g}$ *except that in joint self-supervised signal* $(\mathbf{z}, k)$ *is relatively small:* $I(\mathbf{g}; \mathbf{y}|\mathbf{z}, k) \leq \epsilon$.

Intuitively, this assumption indicates that most of the task-relevant information in the input graph is contained in the self-supervised signal. Based on this assumption, we further give the following theorem which reveals that why the downstream tasks can benefit from the 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 can arrive at the following inequalities about the task-relevant information:*

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

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 $\epsilon$. Thus, we can guarantee a good downstream performance as long as the Assumption 1 is satisfied. It is also noteworthy that jointly utilizing local structure and global semantic as the self-supervised signal is expected to contain more task information. As further enlightenment, we can relate Eq. (11) with the Bayes error rate [40]:

**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:*

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

*where* $\text{Th}(x) = \min\{\max\{x, 0\}, 1 - 1/|M|\}$ *is a thresholded operation [40]. Similarly, we can obtain the Bayes 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 Bayes 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 have superior performance on downstream tasks.

# 4 Experiments

In this section, we empirically evaluate the proposed self-supervised learning method on real-world datasets, and analyze its behavior on graphs to gain further insights. We include additional experiments in Appendix E.

## 4.1 Experimental Setup

**Datasets.** We perform experiments on widely-used homophilic graph datasets: Cora, Citeseer, and Pubmed [35], as well as non-homophilic datasets: Texas, Cornell, Wisconsin [30], Penn94 and Twitch. Penn94 and Twitch are two relatively large non-homophilous graph datasets proposed by [34, 26]. We provide the detailed descriptions, statistics and homophily measures of datasets in Appendix D.1.

**Baselines.** To evaluate the effectiveness of our proposed DSSL, we consider the following representative unsupervised and self-supervised learning methods for the node representation task, including Deepwalk [32], LINE [37], Struc2vec [33], GAE [24], VGAE [24], DGI [42], GraphCL [48], MVGRL [15] and BGRL [38]. The detailed description of baselines and implementations are given in Appendix D.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 [42, 38], 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 same random split with train/val/test split ratio of 60%/20%/20% for the training of downstream linear classifier [26]. 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) [43] as the evaluation metric for clustering.

**Setup.** For all self-supervised methods, we consider a two-layer GCN [23] as the encoder and randomly initialize parameters. We also test methods with other encoders such as GAT [41] (see Appendix E.1). 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 D.3.

## 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 10 runs. Since node clustering results have a similar tendency, we provide them in Appendix E.2. 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.

Table 1: Experimental results (%) with standard deviations on the node classification task. The best and second best performance under each dataset are marked with blue 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** |

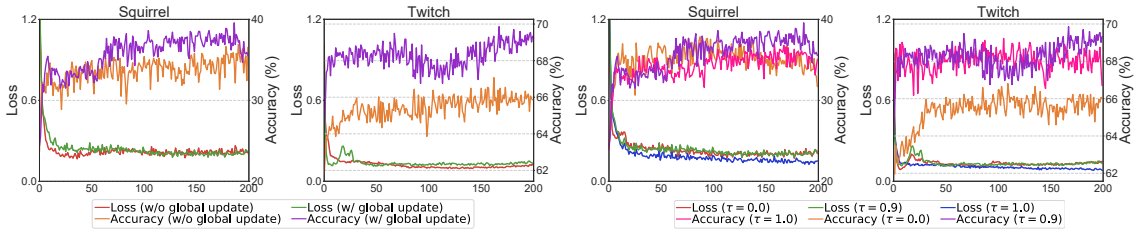



Figure 3: (Left) Performance w and w/o global update on prototypes. (Right) Performance with varying $\tau$.

Especially, DSSL achieves an 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 D.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.

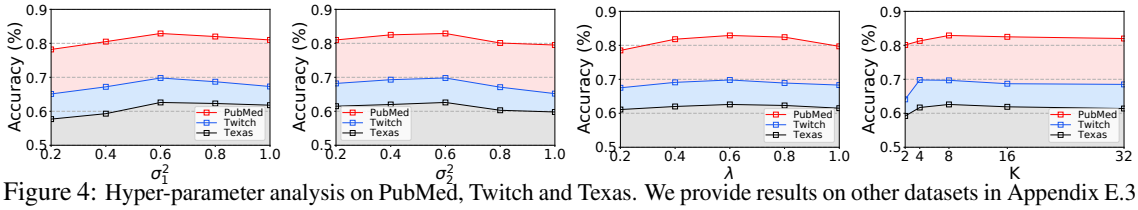



Figure 4: Hyper-parameter analysis on PubMed, Twitch and Texas. We provide results on other datasets in Appendix E.3

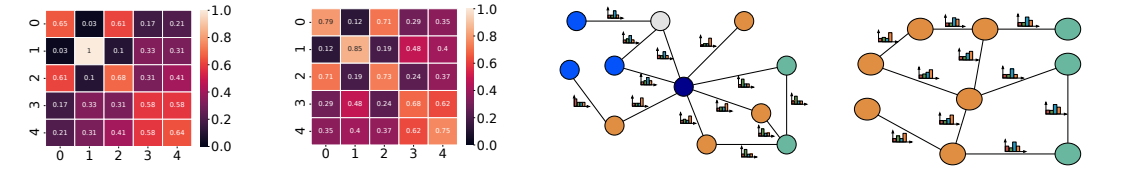
### 4.3 Ablation Study and Parameter Analysis

In this section, we conduct an ablation study to validate our motivation and design, and investigate the sensitivity of hyper-parameters. We refer readers to Appendix E.3 for more results.

**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 many discriminative information and it does not perform very well. $\mathcal{L}_{local}$ as a key component of DSSL alone produces better results than the baselines. We also observe that considering semantic shift and personalized posterior can improve the performance a lot which

Table 2: Node classification accuracy (%) on the Texas dataset. Ablation results for other datasets are given in Appendix E.3.

| 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** |




(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). More results can be found in Appendix E.4.

demonstrates our key motivations. The full model (last row) achieves the best performance, which illustrates that different components designed are complementary to each other.

**Effect of Global Update on Prototypes**. We also conduct the ablation studies to gain insights into the global update on 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 in previous sections, 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.

**Effect of Asymmetric Architecture**. We then explore the effect of target decay rate $\tau$ 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 online network and DSSL demonstrates a degenerated performance and severe overfitting phenomena on Squirrel.

**Hyper-parameters Analysis.** We investigate the hyper-parameters most essential to our framework design, i.e., the standard deviation $\sigma_1$ and $\sigma_2$, the temperature of the Gumbel Softmax $\gamma$, and the total number of factors $K$. The corresponding results are shown in Figure 4. $\sigma_1^2$ resembles the temperature scaling in Eq. (5). We observe $\sigma_1^2$ 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 the all three datasets. case. $\sigma_2^2$ balances the local and global loss in Eq. (5). We can find that having large values of $\sigma_2^2$ does not improve the performance, as the local loss plays more essential role in the proposed model. Further, we observe that DSSL is not very sensitive to the Gumbel softmax temperature $\lambda$, 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 performance slightly drop due to the introduced high model complexity.

### 4.4 Visualization and Case Study

In this section, to understand how DSSL uncovers the latent patterns in neighborhoods, we provide the visualization and case study results. In Figure 5 (a), we first calculate the cross-class neighborhood similarity (see Appendix E.4 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 leaned latent factor distribution shares a similar pattern with the cross-class neighborhood similarity, which meets our expectation that the latent factors can capture the latent semantic information related to neighborhood patterns. To evaluate if DSSL can learn the semantic shift of between different neighbors, we plot the latent factor, i.e., $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$ for individual links. In Figures 5 (b) and (c) , we plot the subgraph of a randomly selected node, and the distribution $q_\phi(k|\mathbf{v}_i, \mathbf{z}_j)$ ($K$=4) and 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 our latent factors can decouple the diverse semantics in the local neighborhoods.

## 5 Conclusions

In this paper, we study the problem of learning unsupervised node representations 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 benchmarks showed that DSSL could learn meaningful class-discriminative representations.

# 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] 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.

[8] 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.

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

[10] 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.

[11] 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.

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

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

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

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

[16] 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.

[17] 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.

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

[19] 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.

[20] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141*, 2020.

[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 Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, 2020.

[32] 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.

[33] 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.

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

[35] 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.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

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

[50] 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.

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

[52] 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.

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

[54] 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.

[55] 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] see Appendix F.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see Appendix F.
   (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] see our supplemental material.
   (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] We report standard deviation of performance metrics across multiple runs.
   (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)? [Yes] We provide computer hardware information in Appendix D.3.
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? [Yes] We cite all creators.
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
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]

Table 3: Experimental results (%) with standard deviations on the node clustering task. The best and second best performance under each dataset are marked with blue and underline, respectively.

| Method | Cora | Citeseer | Pubmed | Texas | Cornell | Squirrel | Penn94 | Twitch |
|---|---|---|---|---|---|---|---|---|
| Deepwalk | 31.15±0.05 | 31.23±0.06 | 34.59±0.09 | 16.90±0.13 | 10.23±0.18 | 16.97±0.13 | 13.24±0.16 | 25.08±0.14 |
| LINE | 33.78±0.03 | 33.37±0.04 | 38.31±0.06 | 18.61±0.15 | 13.57±0.14 | 18.59±0.12 | 14.49±0.09 | 26.81±0.07 |
| Struc2vec | 20.17±0.05 | 19.31±0.08 | 15.13±0.03 | 22.14±0.10 | 9.18±0.12 | 14.85±0.15 | 12.58±0.17 | 22.49±0.14 |
| 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** |

# A The Omitted Derivations

## A.1 The Evidence Lower-Bound

## A.2 The Optimization Loss

## A.3 Derivation of Global Update

# B The Training Algorithm and Network Architecture

## B.1 The Training Algorithm of DSSL

## B.2 The Network Architecture of DSSL

# C Theoretical Analysis

# D Experimental Details

## D.1 Datasets Description and Statistics

## D.2 Baselines and Implementations

## D.3 Experimental setup

# E Additional Experimental Results

## E.1 Results with Other Encoders

## E.2 Node Clustering Results

## E.3 Additional Results on Ablation Study and Parameter Analysis

## E.4 Additional Results on Visualization and Case Study

# F Discussion and Broader impact