Self-Supervised Learning of Graph Neural Networks: A Unified Review

Yaochen Xie®, Zhao Xu, Jingtun Zhang®, Zhengyang Wang®, and Shuiwang Ji®, Senior Member, IEEE

Abstract—Deep models trained in supervised mode have achieved remarkable success on a variety of tasks. When labeled samples are limited, self-supervised learning (SSL) is emerging as a new paradigm for making use of large amounts of unlabeled samples. SSL has achieved promising performance on natural language and image learning tasks. Recently, there is a trend to extend such success to graph data using graph neural networks (GNNs). In this survey, we provide a unified review of different ways of training GNNs using SSL. Specifically, we categorize SSL methods into contrastive and predictive models. In either category, we provide a unified framework for methods as well as how these methods differ in each component under the framework. Our unified treatment of SSL methods for GNNs sheds light on the similarities and differences of various methods, setting the stage for developing new methods and algorithms. We also summarize different SSL settings and the corresponding datasets used in each setting. To facilitate methodological development and empirical comparison, we develop a standardized testbed for SSL in GNNs, including implementations of common baseline methods, datasets, and evaluation metrics.

Index Terms—Deep learning, graph analysis, graph neural networks, review, self-supervised learning, survey, unsupervised learning

1 INTRODUCTION

A deep model takes some data as its inputs and is trained to output desired predictions. A common way to train a deep model is to use the supervised mode in which a sufficient amount of input data and label pairs are given. However, since a large number of labels are required, the supervised training becomes inapplicable in many real-world scenarios, where labels are expensive, limited, imbalanced [1], or even unavailable. In such cases, self-supervised learning (SSL) enables the training of deep models on unlabeled data, removing the need of excessive annotated labels. When no labeled data is available, SSL serves as an approach to learn representations from unlabeled data itself. When a limited number of labeled data is available, SSL from unlabeled data can be used either as a pre-training process after which labeled data are used to fine-tune the pre-trained deep models for downstream tasks, or as an auxiliary training task that contributes to the performance of main tasks.

Recently, SSL has shown its promising capability in data restoration tasks, such as image super-resolution [2], image denoising [3], [4], [5], and single-cell analysis [6]. It has also achieved remarkable progress in representation learning for different data types, including language sequences [7], [8], [9], images [10], [11], [12], [13], and graphs with sequence models [14], [15] or spectral models [16]. The key idea of these methods is to define pretext training tasks to capture and use the dependencies among different dimensions of the input data, e.g., the spatial, temporal, or channel dimensions, with robustness and smoothness. Taking the image domain as an example, Doersch et al. [17], Noroozi and Favaro [18], and He et al. [19] design different pretext tasks to train convolutional neural networks (CNNs) to capture relationships between different crops from an image. Chen et al. [11] and Grill et al. [20] train CNNs to capture dependencies between different augmentations of an image.

Based on how the pretext training tasks are designed, SSL methods can be divided into two categories; namely contrastive models and predictive models. The major difference between the two categories is that contrastive models require data-data pairs for training, while predictive models require data-label pairs, where the labels are self-generated from the data, as illustrated in Fig. 1. Contrastive models usually utilize self-supervision to learn data representation or perform pre-training for downstream tasks. Given the data-data pairs, contrastive models perform discrimination between positive pairs and negative pairs. On the other hand, predictive models are trained in a supervised fashion, where the labels are generated based on certain properties of the input data or by selecting certain parts of the data. Predictive models usually consist of an encoder and one or more prediction heads. When applied as a representation learning or pre-training method, the prediction heads of a predictive model are removed in the downstream task.

In graph data analysis, SSL can potentially be of great importance to make use of a massive amount of unlabeled graphs such as molecular graphs [21], [22]. With the rapid development of graph neural networks (GNNs) [23], [24], [25], [26], [27], [28], [29], basic components of GNNs [30],

© 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
We provide thorough and up-to-date reviews on SSL methods for graph neural networks. To the best of our knowledge, our survey presents the first review of SSL specifically on graph data.

We unify existing contrastive learning methods for GNNs with a general framework. Specifically, we unify the contrastive objectives from the perspective of mutual information. From this fresh view, different ways to perform contrastive learning can be considered as performing three transformations to obtain views. We review theoretical and empirical studies and provide insights to guide the choice of each component in the framework.

We categorize and unify SSL methods with self-generated labels as predictive learning methods, and elucidate their connections and differences by different ways of obtaining labels.

- We summarize common settings of SSL tasks and commonly used datasets of various categories under different settings, setting the stage for developments of future methods.
- We develop a standardized testbed for applying SSL on GNNs, including implementations of common baseline methods and benchmarks, enabling convenient and flexible customization for future methods.

An overview of self-supervised learning methods of different categories is given in Fig. 2.

A recent work [62] provides thorough and general literature reviews on self-supervised learning for vision, natural language processing, and graph mining tasks. While both [62] and our work review SSL methods as contrastive ones and non-contrastive ones, we distinguish our reviews from [62] by the following distinct differences.

- Liu et al. [62] and our paper propose different taxonomies for SSL methods from different aspects of view. Specifically, [62] categorizes contrastive methods by the levels of contrast such as instance-instance and context-instance, where mutual-information is considered as one specific subcategory at the context-instance level. In contrast, our taxonomy provides a more unified view and framework from the theoretical grounding of the methods. Specifically, in our work, all contrastive methods are theoretically grounded by mutual information maximization, and the contrastive objectives are different upper bounds or estimators of mutual information. In our framework, different levels of contrast are determined by how views are selected or generated. We believe that the more unified view enables a more clear comparison and insightful understanding of commons and differences among SSL methods.

- Though adapted to graphs, the taxonomy proposed by [62] is mostly oriented by SSL methods for images. While SSL for graphs and images share some connections and similarities, there are remarkable differences between specific methods for the two types of data and some categorizations of images do not apply to graphs. For example, the relative position and the cluster discrimination methods categorized by [62] are image-specific contrastive methods and do not apply to graphs whereas view generation approaches for graphs and graph-specific predictive tasks are not discussed in [62]. Therefore, graph-specific SSL reviews such as ours are important and necessary for a better understanding of existing methods and benefit future studies.

Recently, another concurrent survey [63] provides reviews on SSL methods for GNNs. The work proposes a taxonomy with more subdivided categories including generation-based, auxiliary property-based, contrastive-based, and hybrid methods. While [63] aims to provide better coverage on existing SSL methods for review, our work focuses on providing a more timely and unified review under comparable frameworks and provide insights into future SSL studies.
2 PROBLEM FORMULATION

2.1 Notations
We consider an attributed undirected graph \( G = (V, E, \alpha) \), where \( V = \{v_1, \ldots, v|V|\} \) denotes the set of its nodes, \( E = \{e_1, \ldots, e|E|\} \) denotes the set of its edges and \( \alpha : V \to \mathbb{R}^d \) denotes the mapping from a node to its attributes of \( d \) dimensions. We denote the adjacency matrix of \( G \) by \( A \in \mathbb{R}^{|V| \times |V|} \), where \( A_{ij} = 1 \) if \((v_i, v_j) \in E \) for \( 1 \leq i,j \leq |V| \), and denote the feature matrix of \( G \) by \( X \in \mathbb{R}^{d \times |V|} \), where the \( ith \) row \( X_i = \alpha(v_i) \) for \( 1 \leq i \leq |V| \). A heterogeneous graph additionally includes elements \( \phi : V \to T_n \) that maps a node to a node type in \( T_n \) and \( \psi : E \to T_e \) that maps an edge to an edge type in \( T_e \).

Given the graph data \( (A, X) \) from the input space \( \mathcal{G} \), we are interested in the representation of the graph at either node-level or graph-level for any downstream prediction tasks. In general, we want to learn an encoder \( f \) such that the representation \( H = f(A, X) \) can achieve desired performance on a downstream prediction task. For node-level prediction tasks, we learn a node-level encoder \( f_n : \mathbb{R}^{d \times |V|} \times \mathbb{R}^{|V| \times d} \to \mathbb{R}^q \) that takes the graph data \( (A, X) \) as inputs and computes the representations for all nodes \( H_{node} = f_{node}(A, X) \in \mathbb{R}^{|V| \times q} \). For graph-level prediction tasks, we learn a graph-level encoder \( f_g : \mathbb{R}^{d \times |V|} \times \mathbb{R}^{|V| \times d} \to \mathbb{R}^q \) that computes a single vector \( h_{graph} = f_{graph}(A, X) \in \mathbb{R}^q \) as the representation of the given graph. Practically, graph-level encoders are usually constructed as a node-level encoder followed by a readout function. In many cases, a model for node-level representation learning may also be utilized to compute graph-level representations by adding an appropriate readout function, and vice versa (by removing the readout function).

We let \( P \) denote the distribution of unlabeled graphs over the input space \( \mathcal{G} \). Given a training dataset, the distribution \( P \) can be simply constructed as the uniform distribution over samples in the dataset. The self-supervision can contribute to the learning of graph encoders \( f \) by utilizing information from \( P \) and minimizing a self-supervised loss \( L_{ssl}(f, P) \) determined by a specifically designed self-supervised learning task.

2.2 Paradigms for Self-Supervised Learning
Typical training paradigms to apply the self-supervision include unsupervised representation learning, unsupervised pretraining, and auxiliary learning, as shown in Fig. 3. In unsupervised representation learning, only the distribution \( P \) of unlabeled graphs is available for the entire training
The unsupervised pretraining and supervised finetuning paradigm is considered as the most strategy to perform semi-supervised learning and transfer learning. For semi-supervised learning, the labeled graphs in the finetuning dataset is a portion of the pretraining dataset. For transfer learning, the pretraining and finetuning datasets are from different domains. Note that a similar learning setting called unsupervised domain adaptation has also been studied generally [66] or specifically in the image domain [67], where the encoder is pre-trained on labeled data but finetuned on unlabeled data under self-supervision. Since the paradigm is not specifically studied in the graph domain, we do not discuss the learning setting in detail in this survey.

The auxiliary learning, also known as joint training [65], aims to improve the performance of a supervised primary learning task by including an auxiliary task under self-supervision. Formally, we let $Q$ denote the joint distribution of graph data and labels for the primary learning task and $P$ denote the marginal of graph data. We want to learn both the decoder $f$ and the prediction head $h$, where $h$ is trained under supervision on $Q$ and $f$ is trained under both supervision and self-supervision on $P$. The learning problem is formulated as

$$f^*, h^* = \arg \min_{f, h} L_{sup}(f, h, Q) + \lambda L_{ssl}(f, P),$$

where $\lambda$ is a positive scalar weight that balances the two terms in the loss.

3 CONTRASTIVE LEARNING

The study of contrastive learning has made significant progress in natural language processing and computer vision. Inspired by the success of contrastive learning in images, recent studies propose similar contrastive frameworks to enable self-supervised training on graph data. Given training graphs, contrastive learning aims to learn one or more encoders such that representations of similar graph instances agree with each other, and that representations of dissimilar graph instances disagree with each other. We unify existing approaches to constructing contrastive learning tasks into a general framework that learns to discriminate jointly sampled view pairs (e.g. two views belonging to the same instance) from independently sampled view pairs (e.g. views belonging to different instances). In particular, we obtain multiple views from each graph in the training dataset by applying different transformations. Two views generated from the same instance are usually considered as a positive pair and two views generated from different instances are considered as a negative pair. The agreement is usually measured by metrics related to the mutual information between two representations.

One major difference among graph contrastive learning methods lies in (a) the objective for discrimination task given view representations. In addition, due to the unique data structure of graphs, graph contrastive learning methods also differ in (b) approaches that views are obtained, and (c) graph encoders that compute the representations of views. A graph contrastive learning method can be determined by specifying its components (a)–(c). In this section, we summarize graph contrastive learning methods in a unified framework and then introduce (a) and (b) individually.
used in existing studies. In Appendix C, available in the online supplemental material, we introduce graph neural networks specifically adopted in contrastive learning as graph encoders and provide further comparisons and discussions on their effects in contrastive learning. Moreover, we summarize all contrastive methods being reviewed by this survey in Supplementary Table 1, available online, for more clear comparisons.

### 3.1 Overview of Contrastive Learning Framework

In general, key components that specify a contrastive learning framework include transformations that compute multiple views from each given graph, encoders that compute the representation for each view, and the learning objective to optimize parameters in encoders. An overview of the framework is shown in Fig. 4. Concretely, given a graph \((A, X)\) as a random variable distributed from \(P\), multiple transformations \(T_1, \ldots, T_k\) are applied to obtain different views \(w_1, \ldots, w_k\) of the graph. Then, a set of encoding networks \(f_1, \ldots, f_k\) take corresponding views as their inputs and output the representations \(h_1, \ldots, h_k\) of the graph from each views. Formally, we have

\[
\begin{align*}
    w_i &= T_i(A, X), \\
    h_i &= f_i(w_i), \quad i = 1, \ldots, k.
\end{align*}
\]

We assume \(w_i = (A_i, X_i) = T_i(A, X)\) in this survey since existing contrastive methods consider their views as graphs. However, note that not all views \(w_i\) are necessarily graphs or sub-graphs in a general sense. In addition, certain encoders can be identical to each other or share their weights.

During training, the contrastive objective aims to train encoders to maximize the agreement between view representations computed from the same graph instance. The agreement is usually measured by the mutual information \(I(h_i, h_j)\) between a pair of representations \(h_i\) and \(h_j\). We formalize the contrastive objective as

\[
\max_{\{h_i\}_{i=1}^k} \frac{1}{\sum_{i \neq j} \sigma_{ij}} \left[ \sum_{i \neq j} \sigma_{ij} I(h_i, h_j) \right], \quad (8)
\]

where \(\sigma_{ij} \in \{0, 1\}\) and \(\sigma_{ij} = 1\) if the mutual information is computed between \(h_i\) and \(h_j\), and \(\sigma_{ij} = 0\) otherwise, and \(h_i\) and \(h_j\) are considered as two random variables belonging to either a joint distribution or the product of two marginals. To enable efficient computation of the mutual information, certain estimators \(\hat{I}\) of the mutual information are usually used instead as the learning objective. Note that some contrastive methods apply projection heads [10], [49] to the representations. For the sake of uniformity, we consider such projection heads as parts of the computation in the mutual information estimation.

During inference, one can either use a single trained encoder to compute the representation or a combination of multiple view representations such as the linear combination

\[
\sum_{i=1}^k \alpha_i h_i,
\]

where \(\alpha_i\) are weights that can be learned or fixed.
or the concatenation as the final representation of a given graph. Three examples of using encoders in different ways during inference are illustrated in Fig. 5.

3.2 Contrastive Objectives Based on MI Estimations

Given a pair of random variables \((x, y)\), the mutual information \(I(x, y)\) measures the information that \(x\) and \(y\) share, given by

\[
I(x, y) = D_{KL}(p(x, y) \| p(x)p(y))
\]

\[
= \mathbb{E}_{p(x, y)} \left[ \log \frac{p(x, y)}{p(x)p(y)} \right],
\]  

(9)

(10)

where \(D_{KL}\) denotes the Kullback-Leibler (KL) divergence. The contrastive learning seeks to maximize the mutual information between two views as two random variables. In particular, it trains the encoders to be contrastive between representations of a positive pair of views that comes from the joint distribution \(p(v_i, v_j)\) and representations of a negative pair of views that comes from the product of marginals \(p(v_i)p(v_j)\).

In order to computationally estimate and maximize the mutual information in the contrastive learning, three typical lower-bounds to the mutual information are derived [68], namely, the Donsker-Varadhan representation \(\hat{I}^{(DV)}\) [69], [70], the Jensen-Shannon estimator \(\hat{I}^{(JS)}\) [71] and the noise-contrastive estimation \(\hat{I}^{(NCE)}\) (InfoNCE) [12], [72]. Among the three lower-bounds, \(\hat{I}^{(JS)}\) and \(\hat{I}^{(NCE)}\) are commonly used as objectives [10], [38], [41], [49] in the contrastive learning in graphs.

A mutual information estimation is usually computed based on a discriminator \(D : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}\) that maps the representations of two views to an agreement score between the two representations. The discriminator \(D\) can be either parametric or non-parametric. For example, the discriminator can optionally apply a set of projection heads [10], [49] to the representations \(h_1, \ldots, h_k\) before computing the pairwise similarity. We formalize the optional projection heads as \(g_i, \ldots, g_k\) such that

\[
z_i = g_i(h_i), \quad i = 1, \ldots, k,
\]

(11)

where \(g_i\) can be an identical mapping, a linear projection or an MLP. Parameterized \(g_i\) are optimized simultaneously with the encoders \(f_i\) in Eqn. (8), given by

\[
\max_{\{f_i, g_i\}_{i=1}^k} \frac{1}{|V|} \sum_{i \neq j} \sigma_{ij} \tilde{I}_{g_i(g_j)}(h_i, h_j),
\]

(12)

In following subsections, we introduce the three lower bounds as specific estimations of mutual information and a non-bound estimation of mutual information. We further compare and discuss the effect of different MI estimations in contrastive learning in Appendix D, available in the online supplemental material.

3.2.1 Donsker-Varadhan Estimator

The Donsker-Varadhan (DV) estimator, also known as the DV representation of the KL divergence, is a lower-bound to the mutual information and hence can be applied to maximize the mutual information. Given \(h_i\) and \(h_j\), the lower-bound is computed as

\[
\hat{I}^{(DV)}(h_i, h_j) = \mathbb{E}_{p(h_i)\| p(h_j)} [D(h_i, h_j)]
\]

\[
- \log \mathbb{E}_{p(h_i)\| p(h_j)} \left[ e^{D(h_i, h_j)} \right].
\]

(13)

where \(p(h_i), p(h_j)\) denotes the joint distribution of the two representations \(h_i, h_j\) and \(p(h_i)p(h_j)\) denotes the product of marginals. For simplicity and to include the graph data distribution \(P\), we assume transformations \(T_i\) to be deterministic and encoders \(f_i\) to be injective, and have \(p(h_i) = p(f_i(T_i(A, X)))\) and \(p(h_j)(A, X)\). We hence re-write Eqn. (13) as

\[
\hat{I}^{(DV)}(h_i, h_j) = \mathbb{E}_{(A, X) \sim P} [D(h_i, h_j)]
\]

\[
- \log \mathbb{E}_{(A, X) \sim P} \left[ e^{D(h_i, h_j)} \right].
\]

(14)

where \(h_i\) and \(h_j\) in the first term are computed from \((A, X)\) distributed from \(P\), \(h_i\) and \(h_j\) in the second term are computed from \((A, X)\) and \((A', X')\) independently and identically distributed from \(P\), respectively. In following formulations, we use the later notation includes \(P\).

3.2.2 Jensen-Shannon Estimator

Compared to the Donsker-Varadhan estimator, the Jensen-Shannon (JS) estimator enables more efficient estimation and optimization of the mutual information by computing the JS divergence between the joint distribution and the product of marginals.

Given two representations \(h_i\) and \(h_j\) computed from the random variable \((A, X)\) and a discriminator \(D, DG\) [38], InfoGraph [41], Hu et al. [43] and MVGRL [10] computes the JS estimator

\[
\hat{I}^{(JS)}(h_i, h_j) = \mathbb{E}_{(A, X) \sim P} \left[ \log (D(h_i, h_j)) \right]
\]

\[
+ \mathbb{E}_{(A, X) \sim P} \left[ \log (1 - D(h_i, h_j)) \right].
\]

(15)

where \(h_i\) and \(h_j\) in the first term are computed from \((A, X)\) distributed from \(P\), \(h_i\) and \(h_j\) in the second term are computed
from \((A, X)\) and \((A', X')\) identically and independently distributed from the distribution \(\mathcal{P}\). To further include edge features, Peng et al. [42] derives the graph mutual information (GMI) based on MI decomposition and optimizes GMI via JS estimator. Note that [41] and [10] depict a softplus version of the JS estimator

\[
\mathcal{I}^{\text{(JS-SP)}}(h_i, h_j) = \mathbb{E}_{(A, X) \sim \mathcal{P}} \left[ -sp(-D'(h_i, h_j)) \right] - \mathbb{E}_{[(A, (A', X')) \sim \mathcal{P} \times \mathcal{P}} \left[ sp(D'(h_i, h_j)) \right],
\]

where \(sp(x) = \log (1 + e^x)\). We consider the JS estimators in Eqs. (15) and (16) to be equivalent by letting \(D(h_i, h_j) = \text{sigmoid}(D'(h_i, h_j))\).

For the negative pairs of graphs \([(A, X), (A', X')] \sim \mathcal{P} \times \mathcal{P}\) in particular, DGI [38] talks about \(D(h_i, h_j) = \text{sigmoid}(D'(h_i, h_j))\). For node-level tasks, MVGLR [10] follows DGI to obtain negative samples by corrupting given graphs. InfoGraph [41] independently samples two graphs from the training dataset as a negative pair, which is followed by MVGLR for graph-level tasks. Discriminators in JS estimators usually compute the agreement score between two vectors by their inner product, i.e., \(D(h_i, h_j) = \text{sigmoid}(z_i^T z_j) = \text{sigmoid}(g(h_i)^T g(h_j))\).

### 3.2.3 Infonce

InfoNCE \(\mathcal{I}^{\text{(NCE)}}(h_i, h_j)\) is another lower-bound to the mutual information \(\mathcal{I}\). It is shown by You et al. [49] that maximizing InfoNCE is equivalent to maximizing the Donsker-Varadhan estimator. Given the representations \(h_i\) and \(h_j\) of two views of random variable \((A, X)\), the discriminator \(D\), and the number of negative samples \(N\), the InfoNCE is formalized as

\[
\mathcal{I}^{\text{(NCE)}}(h_i, h_j) = \mathbb{E}_{(A, X) \sim \mathcal{P}} \left[ D(h_i, h_j) \right] - \mathbb{E}_{K \sim \mathcal{P}^N} \left[ \log \sum_{(A', X') \in K} e^{D(h_i, h_j')}/N(A, X) \right] \]

\[
= \mathbb{E}_{[(A, X), K] \sim \mathcal{P} \times \mathcal{P}^N} \left[ \log \sum_{(A', X') \in K} e^{D(h_i, h_j')}/N(A', X') \right] + \log N,
\]

where \(K\) consists of \(N\) random variables identically and independently distributed from \(\mathcal{P}\), \(h_i, h_j\) are the representations of the \(i\)th and \(j\)th views of \((A, X)\), and \(h_{j}'\) is the representation of the \(j\)th view of \((A', X')\).

In practice, we compute the InfoNCE on mini-batches of size \(N + 1\). For each sample \(x\) in a mini-batch \(B\), we consider the set of the rest \(N\) samples as a sample of \(K\). We then discard the constant term \(\log N\) in Eqn. (17) and minimize the loss

\[
\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N + 1} \sum_{x \in B} \log \frac{e^{D(h_i, h_j)}}{\sum_{x' \in B(x)} e^{D(h_i, h_j')}}.
\]

Intuitively, the optimization of InfoNCE loss aims to score the agreement between \(h_i\) and \(h_j\) of views from the same instance \(x\) higher than between \(h_i\) and \(h_{j}'\) from the rest \(N\) negative samples \(B \setminus \{x\}\). GraphCL [49] and GRACE [46] include additional contrast among the same view of different instances \(i.e., h_i\) and \(h_{j}'\) or different nodes in the same view (intra-view contrast [46]), leading to the optimization of lower bounds of InfoNCE, which are still lower bounds of MI.

Discriminators in typical InfoNCE compute the agreement score between two vectors by their inner product, i.e., \(D(h_i, h_j) = z_i^T z_j = g(h_i)^T g(h_j)\). A specific type of the InfoNCE loss, known as the NT-Xent [73] loss, includes normalization and a preset temperature parameter \(\tau\) in the computation of discriminator \(D\) in the InfoNCE loss, i.e., \(D(h_i, h_j) = g(h_i)^T g(h_j)/\tau\). The discriminator in You et al. [49] computes the agreement score between vectors with normalizations, i.e., \(D(h_i, h_j) = g(h_i)^T g(h_j)/\|g(h_i)\|\|g(h_j)\|\), where \(\|\|\) denotes the \(\ell_2\)-norm.

### 3.2.4 Other Mutual Information Estimators

There are other objectives that have been used in some studies, and optimizing these objectives can also encourage higher mutual information. Although the objectives differ from the above upper bound MI estimators.

However, these objectives may not be provable lower-bounds to the mutual information, and optimizing these objectives does not guarantee the maximization of the mutual information.

For example, Jiao et al. [44] proposes to minimize the triplet margin loss [74], which is commonly used in deep metric learning [75]. Given representations \(h_i, h_j\) and the discriminator \(D\), the triplet margin loss is formalized as

\[
\mathcal{L}_{\text{triplet}} = \mathbb{E}_{[(A, X), (A', X')] \sim \mathcal{P} \times \mathcal{P}} \left[ \max\{D(h_i, h_j) - D(h_i, h_j') + \epsilon, 0\} \right],
\]

where \(D(h_i, h_j)\) is computed as \(\text{sigmoid}(h_i^T h_j)\) or based on the euclidean distance \(\|h_i - h_j\|\) and \(\epsilon\) is the margin value. While the triplet loss differs from previous MI-based objectives in formulations, Khosla et al. [76] show that the triplet loss is a special case of the InfoNCE (NT-Xent) loss when there is only one negative sample where the margin value \(\epsilon\) corresponds to the temperature parameter \(\tau\) in NT-Xent. Moreover, the Bayesian Personalized Ranking (BPR) loss [77] used in Jiao et al. [44] is also equivalent to the InfoNCE loss when letting \(N = 1\) and \(D(h_i, h_j') = h_i^T h_j\).

### 3.2.5 Projection Heads: Parametric MI Estimation

Many contrastive learning studies [10], [41], [49] propose to include projection heads \(g\) when computing the MI estimators. For example, Sun et al. [41], Hassani and Khasahmadi [10] use 3-layer MLPs, You et al. [49] use 2-layer MLPs as the projection heads and Sun et al. [41] applies a linear projection to the graph-level representation. The projection heads are shown to significantly improve the contrastive learning performance [11]. For contrastive learning on heterogeneous graphs, it is common to apply individual projections to representations of different type of nodes. For example, Jiang et al. [78] adopt \(D(h_u, h_v) = [W_{\phi(v)} h_u]^T [W_{\phi(u)} h_u] = h_u^T W_R h_u\) for nodes \(u\) and \(v\) with types \(\phi(u)\) and \(\phi(v)\) connected by the relationship \(R\), where \(W_R = W_{\phi(u)}^T W_{\phi(v)}\).
We consider MI estimators that include projection heads as parametric estimators and those without projection heads as non-parametric estimators. Then a reasonable explanation to the observation that the contrastive methods with projection heads usually achieve better performance is that parametric estimators provide better estimation to the mutual information.

3.3 Graph View Generation

To generate views from a graph sample distributed from \( P \), one usually applies different types of graph transformations (or augmentations) \( T \). Here, we only consider cases where \( T \) still outputs the graph-structured data. We summarize the existing transformations applied to graph data in three categories, feature transformations, structure transformations and sampling-based transformations. Feature transformations can be formalized as

\[
T_{\text{feat}}(A, X) = (A, T_X(X)),
\]

where \( T_X : \mathbb{R}^{|V| \times d} \rightarrow \mathbb{R}^{|V| \times d} \) performs the transformation on the feature matrix \( X \). Structure transformations can be formalized as

\[
T_{\text{struct}}(A, X) = (T_A(A), X),
\]

where \( T_A : \mathbb{R}^{|V| \times |V|} \rightarrow \mathbb{R}^{|V| \times |V|} \) performs the transformation on the adjacency matrix \( A \). And the sampling-based transformations are in the form

\[
T_{\text{sample}}(A, X) = (A[S, S], X[S]),
\]

where \( S \subseteq V \) denotes a subset of nodes and \( [\cdot] \) selects certain rows (and columns) from a matrix based on indices of nodes in \( S \). We consider the transformations applied in the existing contrastive learning methods to generate different views as a single or a combination of instantiations of the three types of transformations above. Note that when node-level representations are of interest, the node-level contrasts are usually included. We consider nodes representations to be computed from views generated by ego-nets sampling from given graphs.

3.3.1 Feature Transformations

Given an input graph \((A, X)\), a feature transformation only performs transformation to the attribute matrix \( X \), i.e.,

\[
T(A, X) = (A, T_X(X)).
\]

Node attribute masking [49] is one of the most common ways to apply the feature transformations. The node attribute masking randomly masks a small portion of attributes of all nodes with constant or random values. Concretely, given the input attribute matrix \( X \), we specify \( T_X(X) \) for the node attribute masking as

\[
T_X^{(\text{mask})}(X) = X * (1 - 1_m) + M * 1_m,
\]

where \( * \) denotes the element-wise multiplication, \( M \) denotes a matrix with masking values and \( 1_m \) denotes the masking location indicator matrix. Given the masking ratio \( r \), elements in \( 1_m \) are set to 1 individually with a probability \( r \) and 0 with a probability \( 1 - r \). To employ adaptive masking, Zhu et al. [45] propose to sample \( 1_m \) with centrality-based probabilities, including degree centrality, eigenvector centrality, and PageRank centrality. The values in \( M \) specify different masking strategies. For example, \( M = 0 \) applies a constant masking, \( M \sim N(0, \Sigma) \) replaces the original values by Gaussian noise and \( M \sim N(X, \Sigma) \) adds Gaussian noise to the original values.

In addition to contrastive models such as [49], attributes masking is also commonly applied in predictive models [3], [6] for regularized reconstruction. The node attribute masking forces the encoders to capture better dependencies between the masked attributes and unmasked context attributes and recover the masked value from its context during encoding.

3.3.2 Structure Transformations

Given an input graph \((A, X)\), a structure transformation only performs transformation to the adjacency matrix \( A \) and \( X \) to be the same, i.e.,

\[
T(A, X) = (T_A(A), X).
\]

Existing contrastive methods apply two types of structure transformations, edge perturbation that randomly adds or drops edges between pairs of nodes and graph diffusion that creates new edges based on the accessibility from one node to another.

Edge perturbation [48], [49] randomly adds or drops edges in a given graph. Similarly to the node attribute masking, it applies masks to the adjacency matrix \( A \). In particular, we have

\[
T_{A}^{(\text{pert})}(A) = A * (1 - 1_p) + (1 - A) * 1_p,
\]

where \( * \) denotes the element-wise multiplication and \( 1_p \) denotes the perturbation location indicator matrix. Given the perturbation ratio \( r \), elements in \( 1_p \) are set to 1 individually with a probability \( r \) and 0 with a probability \( 1 - r \). In addition, \( 1_p \) is a symmetric matrix.

Diffusion [10] creates new connections between nodes based on random walks, aiming at generating a global view \((S, X)\) of the graph in contrast to the local view \((A, X)\). Two instantiations of diffusion transformations are proposed to use in [10], namely, the heat kernel \( T_A^{(\text{heat})} \) and the Personalized PageRank \( T_A^{(\text{PPR})} \), formulated as follows:

\[
T_A^{(\text{heat})}(A) = \exp(tAD^{-1} - t),
\]

\[
T_A^{(\text{PPR})}(A) = a \left(I_n - (1 - a)AD^{-1/2}AD^{-1/2}\right)^{-1},
\]

where \( D \in \mathbb{R}^{|V| \times |V|} \) is a diagonal degree matrix, \( a \) denotes the teleport probability in a random walk and \( t \) denotes the diffusion time.

Centrality-based edge removal [45] randomly removes edges based on pre-computed probabilities determined by the centrality score of each edge. Centrality-based probabilities for edge removal reflects the importance of each edge, where less important edges are more likely to be removed. In particular, the centrality score of an edge \((u, v) \in E \) is computed as \( w_{uv} = (\phi_u(u) + \phi_v(v))/2 \), where \( \phi_u(u) \) and \( \phi_v(v) \) are the centrality of nodes \( u \) and \( v \) connected by the edge and a higher centrality score leads to lower probability \( p_{uv} \) of edge removal.

3.3.3 Sampling-Based Transformations

We consider sampling-based transformations that sample node-induced sub-graphs from a given graph \((A, X)\), i.e.,
\[ T_{\text{sample}}(A, X) = (A[S,S], X[S]) \] with \( S \subseteq V \). Note that more generalized sub-graph sampling methods that sample both nodes from \( V \) and edges from \( E \) can be considered as a combination of the node-induced sub-graph sampling and the edge perturbation. As different sampling-based transformations are determined by the set \( S \) of sampled nodes from the node-set \( V \), we categorize the sampling-based transformations by how the set \( S \) is obtained. Existing contrastive methods apply three approaches to obtain the node subset \( S \): uniform sampling, random walk sampling, and ego-nets sampling.

Uniform sampling and nodes dropping can be considered as the two simplest sampling-based transformation approaches. The transformation in [10] samples sub-graphs by uniformly sampling a given number of nodes \( S \) from \( V \) and edges of the sampled nodes. In addition, transformation methods in [49] include node dropping as one of the graph transformations, where each node has a certain probability to be dropped from the graph. We denote the set of dropped nodes by \( D \) and we have \( S = V \setminus D \).

Ego-nets sampling can be considered as a sampling-based transformation to unify the contrast performed between graph-level representation and node-level representations in the general contrastive learning framework, such as in DGI [38], InfoGraph [41] and MVGRL [10]. In other words, we consider that node-level representations are computed by node-level encoders from certain views, namely, ego-nets, of a given graph. Given a typical graph encoder with \( L \) layers, the computation of the representation of each node \( v_i \) only depends on its \( L \)-hop neighborhood, also known as the \( L \)-ego-net of node \( v_i \). We hence consider the computing of node-level representations as performing \( L \)-ego-net sampling and a node-level encoder with \( L \) layers. In particular, for each node \( v_i \) in a given graph, the transformation \( T_i \) samples the \( L \)-ego net surrounding node \( v_i \) as the view \( w_i \), computed as

\[ w_i = T_i(A, X) = (A[N_L(v_i); N_L(v_i)], X[N_L(v_i)]), \quad (27) \]

\[ N_L(v_i) = \{ v : d(v, v_i) \leq L \}, \quad (28) \]

where \( L \) denotes the number of layers in the node-level encoder \( f_L \). \( d \) denotes the shortest distance between two nodes and \( (A[\cdot], X[\cdot]) \) selects a sub-graph from \( (A, X) \).

Random walk sampling is proposed in GCC [48] to sample sub-graphs based on random walks starting from a given node. The subset of nodes \( S \subseteq V \) is collected iteratively. At each iteration, the walk has a probability \( p_{ij} \) to travel from node \( v_i \) to node \( v_j \) and has a probability of \( p_v = 0.8 \) to return to the start node. GCC considers the random walk sampling with restart as a further transformation of the \( r \)-ego-net centered at the start node. Given the center (start) node, the random walk sampling can be hence considered as a stronger sampling-based transformation than the ego-nets sampling.

Network Schema and Meta-path views are proposed in HeCo [79] as two specific views for the contrastive learning of heterogeneous graphs. Given a target node of type \( t \), the network schema view is a special case of \( 1 \)-ego nets consisting of neighbor nodes whose types are connected to the target node type in the network schema, and excluding nodes with the same type \( t \). When computing the representation for a network schema view, aggregations are computed individually for each node type. The meta-path view consists of all meta-paths between the target node and other nodes of the same type. When computing the representation for a meta-path view, nodes of other types are masked and the information is aggregated along individual meta-paths.

### 3.3.4 Discussions of Graph View Generation

Currently, there is no theoretical analysis guiding the generation of the view for graphs. However, Tian et al. [80] theoretically and empirically analyze the problem in a general view and image domain, considering the generation of the view from the aspect of mutual information. In particular, a good view generation should minimize the MI between two views \( I(v_1, v_2) \), subject to \( I(v_1, y) = I(v_2, y) = I(x, y) \). Intuitively, to guarantee that contrastive learning works, the generated views \( v_i \) should not affect the information that determines the prediction for the downstream task, under which restrictions, stronger disagreement between views leads to better learning results. Following the above idea, AD-GCL [81] proposes to generate views of graphs that achieve the above minimum under constraints by parameterizing the above types of transformations and propose learnable transformations. In particular, the transformations are learned in an adversarial manner - the transformation (views generator) is trained to minimize \( I(v_1, v_2) \) subject to \( I(v_1, y) = I(v_2, y) = I(x, y) \), whereas the encoder is trained to maximize \( I(v_1, v_2) \). Following a similar principle, InfoGCL [82] proposes to discretely select optimal views from a list of candidate views based on the mutual information with downstream tasks.

From the manifold point of view, a recent analytic study [83] proposes the expansion assumption and explains the data augmentation as to prompt the continuity in the neighborhood for each instance. It indicates similar requirements for the view generating by augmentation, i.e., an ideal augmentation should satisfy the following two conditions, 1) the (augmentation) neighborhood of an instance does not intersect the neighborhood of instances that belong to the other class in the downstream task, 2) the neighborhood of an instance should be as large as possible, subject to 1.

To this end, the learning on datasets with different downstream tasks may benefit from different types of transformations. For example, the property of a social network to be predicted in a downstream task may be more tolerant of minor changes in node attributes, for which the feature transformations can be more suitable. On the other hand, the property of a molecule usually depends on bonds in some functional groups, for which the edge perturbation may harm the learning while the sub-graph sampling could help. Empirically, You et al. [49] observes similar results. For example, edge perturbation is found contributory to the performance on social network datasets but harmful to some molecule data.

### 4 Predictive Learning

Compared with contrastive learning methods, predictive learning methods train the graph encoder \( f \) together with a prediction head \( g \) under the supervision of informative labels self-generated for free. We use the term “predictive” instead of “generative” categorized by Liu et al. [62] to avoid confusion, as not all methods introduced in this section are necessarily generative models. Categorized by how the prediction
targets are obtained, we summarize predictive learning frameworks for graphs into (1) graph reconstruction that learns to reconstruct certain parts of given graphs, (2) invariance regularization that aims to directly learn robust and informative representations, (3) graph property prediction that learns to model non-trivial properties of given graphs, and (4) multi-stage self-training with pseudo-labels. In this section, we let \( H \in \mathbb{R}^{1 \times q} \) denote the desired node-level representation and \( h_i \) denote the representation of node \( v_i \). The general frameworks of four types of predictive learning methods are shown in Fig. 6. We summary all predictive methods being reviewed by this survey in Supplementary Table 2, available online, for a more clear comparison.

4.1 Graph Reconstruction

Graph reconstruction provides a natural self-supervision for the training of graph neural networks. The prediction targets in graph reconstruction are certain parts of the given graphs such as the attribute of a subset of nodes or the existence of edge between a pair of nodes. In graph reconstruction tasks, the prediction head \( g \) is usually called the decoder which reconstructs a graph from its representation.

4.1.1 Non-Probabilistic Graph Autoencoders

The autoencoders, first proposed in [84], have been widely applied for the learning of data representations. Given the success in the image domain and natural language modeling, various variations of graph autoencoder [85] are proposed to learn graph representations. Aiming at learning the graph encoder \( f \), graph autoencoders are trained to reconstruct certain parts of an input graph, given restricted access to the graph or under certain regularization to avoid identical mapping.

\( \hat{A} = g(H) = \sigma(HH^T), \)
\( H = f(A, X), \)  

and is optimized by the binary cross-entropy loss between \( \hat{A} \) and \( A \). As GAE is originally proposed to learn node-level representations for link prediction problem, it assumes two linked nodes should have similar representations. GraphSAGE [86] introduces a similar framework with the self-supervision of the adjacency matrix based on a different objective including negative sampling. In addition, a recent work SuperGAT [87] includes the GAE objective as a self-supervised auxiliary loss during training a graph attention network to guide the learning of more expressive attention operators. Similarly, SimP-GCN [88] applies node-pair similarity as a substitute of the adjacency matrix to construct a self-supervised auxiliary task.

MGAE [50] follows the idea of denoising autoencoder [89]. Given a graph \( (A, X) \), MGAE performs reconstructions on multiple randomly corrupted feature matrices \( \{X_i\}_{i=1}^m \) with a single-layer autoencoder \( f_\theta \) and the objective

\[ \sum_{i=1}^m \|X - f_\theta(A, \hat{X}_i)\|^2 + \lambda \|\theta\|^2, \]

where \( \theta \) denotes the weights in the single-layer encoder, \( \lambda \) denotes the hyper-parameter for \( \ell^2 \)-regularization, and \( H_i = f_\theta(A, \hat{X}_i) \) is considered as the reconstructed representation. To enable non-linearity, [50] proposes to stack multiple single-layer autoencoders. Formally, given the reconstructed representation \( H^{(\ell-1)} \) at the \((\ell - 1)\)th layer, the \( \ell \)th layer is trained by optimizing

\[ \sum_{i=1}^m \|H^{(\ell-1)} - H_i^{(\ell)}\|^2 + \lambda \|\theta_i\|^2, \]

\[ H_i^{(\ell)} = f_\theta(A, \hat{H}^{(\ell-1)}_i), \]

where \( \hat{h}_i^{(\ell-1)} \) denotes the corrupted representation from the \((\ell - 1)\)th layer. The reconstructed representation at the last
layer is then considered as the representation for downstream tasks.

GALA [51] introduces a multi-layer autoencoder with symmetric encoder and decoder, unlike GAE and MGAE. Motivated by the Laplacian smoothing [90] effect of GCN encoders, GALA designs the decoder by performing Laplacian sharpening [90], which prompts the decoded representation of each node to be dissimilar to the centroid of its neighbors. A Laplacian sharpening layer in the decoder \( g \) in computed by

\[
\hat{X}^{(i)} = 2\hat{X}^{(i-1)} - D^{-1}AX^{(i-1)},
\]

(34)

where \( \hat{X}^{(i)} \) and \( \hat{X}^{(i-1)} \) denote the decoded representation and \( D \) denotes the degree matrix. GALA reconstructs the feature matrix by optimizing the mean squared error \( \|X - \hat{X}\|^2 \) with

\[
\hat{X} = g(A, H), \quad H = f(A, X).
\]

(35)

Attribute masking [43], also referred to as graph completion [91], is another strategy to pre-train graph encoder \( f \) under the graph autoencoder framework by reconstructing masked node attributes. The encoder \( f \) computes the node-level representations \( H \) given the graph with its node attributes randomly masked. And a linear projection is applied to \( H \) as the decoder \( g \) to reconstruct the masked attributes. When the edge attributes are also available, one can also perform reconstruction on the masked edge attributes. Although the attribute masking is not explicitly named as graph autoencoders, we categorize it as graph autoencoders since the encoders are trained by performing reconstruction on the entire or certain parts of the input graph.

4.1.2 Variational Graph Autoencoders

Although sharing a similar encoder-decoder structure with standard autoencoders, variational autoencoders as generative models are built upon a different mathematical foundation assuming an existing prior distribution of latent representation that generates the observed data. Primarily targeted in learning the generation of the observed data, variational graph autoencoders also shown promising performance in learning good graph representations.

VGAE [39] introduces the simplest version of variational graph autoencoders. VGAE performs reconstruction on the adjacency matrix and is composed a inference model (encoder) \( q(H|A, X) = \prod_{i=1}^n \mathcal{N}(h_i|\mu_i(A, X), \Sigma_i(A, X)) \) parameterized by graph neural networks \( \mu, \Sigma \) and a generative model (decoder) \( p(A|H) \) modeled by the inner product of latent variables. VGAE optimizes the variational lower bound

\[
\mathbb{E}_{q(H|A, X)}[\log p(A|H)] - \text{KL}[q(H|A, X)||p(H)],
\]

(36)

where \( \text{KL}(\cdot) \) denotes the KL-divergence and \( p(H) \) denotes the prior given by the Gaussian distribution.

ARGA/ARVGA [52] propose to regularize the autoencoder with an adversarial network [92] which enforces the distribution of the latent variable to match the Gaussian prior. In addition to the encoder and decoder, a discriminator is trained to distinguish fake data generated by the encoder and the real data sampled from the Gaussian distribution. As the adversarial regularization is provably an equivalence of the JS-divergence between the distribution of the latent variable and the Gaussian prior, ARGA/ARVGA can achieve a similar effect to VGAE but with stronger regularization.

SIG-VAE [53] replace the inference model in the variational graph autoencoder a hierarchy of multiple stochastic layers to enable more flexible model of the latent variable. In particular, the inference model is given by \( p(H|A, X) = p(H|A, X, \mu, \Sigma) \), where \( \mu \) and \( \Sigma \) are considered as random variables computed by stacked stochastic layers with noise injected to each layer. The marginalized \( p(H|A, X) \) is hence not necessarily a Gaussian distribution and enabled higher flexibility and expressivity.

There exist other variations of the variational graph autoencoders such as JTVAE [93] and DGVAE [94]. However, those variational graph autoencoders focus on the generation of graphs. As we mainly consider the learning of representations, we omit the introduction to those methods.

4.1.3 Autoregressive Reconstruction

Following the idea of GPT [95] for the generative pre-training of language models, GPT-GNN [54] proposes a autoregressive framework to perform reconstruction on given graphs. Ash both variational autoencoders and the autoregressive models are generative and based on reconstruction, graph autoregressive models differ from that they perform reconstruction iteratively. In particular, GPT-GNN consists of a graph encoder \( f \), decoders \( g_0 \) and \( g_1 \) for node generation and edge generation, respectively. Given a graph with its nodes and edges randomly masked, GPT-GNN generates one masked node and its edges at a time and optimizes the likelihood of the node and edges generated in the current iteration. GPT-GNN iteratively generates nodes and edges until all masked nodes are generated.

4.2 Representation Invariance Regularization

Adopting predictive objectives based on invariance regularization is recently trending for both image and graph domains. Methods adopting invariance regularization directly computes losses on representations and usually follows a similar framework to contrastive learning, i.e., to obtain two augmented graphs of the given graph, and compute the representations of the two graphs, but their objective does not include any contrast nor requires paired or negative samples. Hence they are categorized as predictive approaches. In particular, the objective seeks to minimize the difference between representation of two distorted graphs, encouraging representations of the graphs to be invariant to random distortions. Certain approaches are introduced to enable the learning informative representations, preventing trivial solutions to be learned.

Inspired by BYOL [20] in the image domain, BGRL [47] proposes a variation of contrastive learning framework, which eliminates the need of negative samples. Given a mini-batch of graphs \( B \), it compute node representations \( H_{x,a} \) and \( H_{x,b} \) of two augmented graphs from each \( x \) in \( B \) and minimize the following invariance-based loss with a parametric predictor \( p_h \).
\[
L_{\text{BYOL}} = \mathbb{E}_{B \times \mathcal{P}_N} \left[ -\frac{2}{N} \sum_{x \in B} [p_b(H_{x,a})]^T H_{x,b} \right].
\]

As no negative sample is included, certain mechanisms and restrictions, such as updating an offline encoder with exponential moving average \([20]\) and batch normalization, are required in the framework to prevent degenerate solutions and achieve similar effect of optimizing MI bound objectives. BGRL and BYOL are commonly acknowledged as variations of contrastive methods. While the framework of BGRL follows the typical contrastive framework, the computation of the above invariance-based objective does not require paired samples or negative samples.

CCA-SSG \([60]\) proposes an invariance-based objective inspired by a well-studied idea of canonical correlation analysis \([96], [97]\). The proposed objective consists of an invariance term minimizing the difference between two representations and a decorrelation term minimizing the correlation among dimensions of the representations, formulated as

\[
L_{\text{CCA}} = \mathbb{E}_{B \times \mathcal{P}_N} \left[ \| H_a - H_b \|^2 
+ \lambda \left( \| H_a^T H_a - I \|^2 + \| H_b^T H_b - I \|^2 \right) \right].
\]

where \(H_a\) and \(H_b\) are batched node representations of graphs with two augmentations \(a\) and \(b\). Similarly to Barlow-Twins \([98]\), CCA-SSG uses batched representations to estimate the correlations among different dimensions. Both CCA-SSG and Barlow-Twins encourage the learning of informative representation by reducing the correlation or redundancies among dimensions.

LaGraph \([61]\) proposes another invariance-based objective based on the assumption that all observed graph data have their latent counterparts, analogically to inaccessible clean counterparts of observed noisy data such as images. The proposed objective is derived as an self-supervised upper bound to the supervised latent graph prediction loss, formulated as

\[
L_{\text{LaGraph}} = \mathbb{E}_{(A,X) \sim \mathcal{P}_J} \left[ \| \mathcal{D}(A,H) - X \|^2 / |V| 
+ \lambda \left( \| 1_{/J} \circ (H - H') \|^2 / |J| \right)^{1/2} \right].
\]

where \(\mathcal{D}\) is a decoder network, \(J\) is a random subset of node indices, \(H\) is the node representation matrix of the given graph, \(H'\) is the representation of the graph whose nodes in \(J\) are masked, and \(1_{/J} \circ \) means the invariance is computed on the masked nodes only. Different from the above two methods, LaGraph computes the representations for the original graph and its masked version, instead of two randomly augmented graphs, and only computes the invariance on masked nodes. The characteristics come from the derivation of the objective.

One intuition behind the invariance regularization-based methods is that the learned representation are expected to contain enough information of the given data but be invariance to distortions on the data. Both CCA-SSG and LaGraph discuss the relationships between invariance-based method and the Information Bottleneck principle \([99]\) indicating the above intuition. Moreover, LaGraph further discusses its relationship with denoising autoencoder and mutual information-based methods.

### 4.3 Graph Property Prediction

In addition to the reconstruction, an efficient way to perform self-supervised predictive learning is to design the prediction tasks based on informative graph properties that are not explicitly provided in the graph data. Commonly applied properties for self-supervised training include topology properties, statistical properties, and domain-knowledge involved properties.

\(S^2\) GRL \([55]\) generalizes the adjacency matrix reconstruction task to a \(k\)-hop connectivity prediction task between two given nodes, motivated by the fact that the interaction between two nodes is not limited to their direct connection. In particular, given encoded representations of any pair of nodes, the prediction head performs classification on the absolute difference between the representations. \(S^2\) GRL then trains the encoder and prediction head to classify the hop counts between the pair of nodes.

Meta-path prediction \([57]\) provides a self-supervision for heterogeneous graphs, such as molecules, which include multiple types of nodes and edges. A meta-path of length \(\ell\) is defined as a sequence \((t_1, \ldots, t_\ell)\) where \(t_i\) denotes the type of the \(i\)th edge in the path. Given two nodes in a heterogeneous graph and \(K\) meta-paths, the encoder \(f\) and prediction heads \(g_i\) \((i = 1, \ldots, K)\) are trained to predict if the two nodes are connected by each of the meta-paths as a binary classification task. In \([57]\), the predictions of the \(K\) meta-paths are included as \(K\) auxiliary learning tasks in addition to the main learning task.

GROVER \([56]\) performs self-supervised learning on molecular graph data with two predictive learning tasks. In contextual property prediction, the encoder and prediction head is trained to predict the "atom-bond-count" relationship within the \(k\)-hop neighborhood of a given node (atom), e.g. "O-double_bond-2" if there are two atoms "O" connected to the given atom with double bonds. In addition, a graph-level motif prediction task is applied to involve the self-supervision of domain knowledge. For molecular graphs, the motifs are instantiated by the functional groups in molecules. Given a list of motifs, the graph-level prediction head predicts the existence of each motif, as a multi-label classification task.

### 4.4 Self-Training With Pseudo-Labels

Instead of the labels obtained from input graphs, the prediction targets in self-training methods are pseudo-labels obtained from the prediction in a previous stage utilizing a small portion of labeled data \([1]\) or even randomly initialized. The self-trained graph neural networks can be either applied under a semi-supervised setting or further fine-tuned for downstream tasks. We consider the node-level classification for an instance.

Under the node-level semi-supervised setting, the multi-stage self-training \([100]\) is proposed to utilize the labeled nodes to guide the training on unlabeled nodes. Concretely, given both the labeled node set and unlabeled node set, the graph neural network is first trained on the labeled set.
After the training, it performs prediction on the unlabeled set and the predicted labels with high confidence are considered as the pseudo-labels and moved to the labeled node set. Then a fresh graph neural network is trained on the updated labeled set and the above operations are performed multiple times.

M3S [58] applies DeepCluster [101] and an aligning mechanism to generate pseudo-labels on the basis of multi-stage self-training. In particular, a $K$-mean cluster is performed on node-level representations at each stage and the labels obtained from clustering are then aligned with the given true labels. A node with clustered pseudo-label is added to the labeled set for self-training in the next stage only if it matches the prediction of the classifier in the current stage. Compared to the basic multi-stage self-training, M3S considers the DeepCluster and the aligning mechanism as a self-checking mechanism and hence provides stronger self-supervision.

ICF-GCN [59] proposes to optimize the GCN model and pseudo-labels for nodes simultaneously in an Expectation-Maximization (EM) manner. In particular, the E-step updates the GCN based on the given pseudo-labels whereas the M-step updates the pseudo-labels based on the GCN predictions. Similarly to M3S, ICF-GCN performs clustering on hidden representations to obtain GCN predicted classes. To avoid the alignment issue, both pseudo-labels and the clustered node classes are represented in relational matrix of shape $V_j \times C_2^V_j$, where an element value 1 indicates two nodes belong to the same class and 0 indicates different classes.

A recent study [83] provides the theoretical justification for the self-training with pseudo-labels based on an assumption of the expansion property and generalizes the self-training methods from semi-supervised settings into the unsupervised setting. Intuitively, the examples with correct pseudo-labels will be utilized to denoise the incorrectly pseudo-labeled examples and high accuracy can be achieved due to the expansion assumption. Under the unsupervised setting, it theoretically shows that a classifier trained with arbitrarily assigned pseudo-labels can still achieve good accuracy for some permutation of the classes.

## 5 Summary of Learning Tasks and Datasets

The self-supervised learning methods are usually applied to and evaluated on two common types of graph-related learning tasks, the graph-level inductive learning and the node-level transductive learning. The graph-level inductive learning aims to learn models predicting graph-level properties, and the models are trained and perform prediction on different sets of graphs. On the other hand, the node-level transductive learning aims to learn models performs node-level property prediction, trained and performing prediction on the same sets of large graphs. In this section, we summarize datasets under the two types of learning tasks. The statistics of common datasets are shown in Table 1.

### 5.1 Graph-Level Inductive Learning

Graph-level learning tasks are performed as inductive learning tasks on multiple graphs [38]. Commonly used datasets for graph-level learning tasks can be divided into three types, chemical molecule datasets, protein datasets, and social network datasets.

**Chemical Molecular Property Prediction.** In a molecular graph, each node represents an atom in a molecule where the atom index is indicated by the node attribute and each edge represents a bond in the molecule. Datasets for chemical molecular property prediction are also categorized as

| Datasets       | Learning tasks          | Task level          | Category                           | # graphs | Avg. nodes | Avg. edges | # classes |
|----------------|-------------------------|---------------------|------------------------------------|----------|------------|------------|-----------|
| NCII           | Unsupervised or semi-supervised classification | Graph               | Small molecules                    | 4110     | 29.87      | 32.30      | 2         |
| MUTAG          |                         |                     | Small molecules                    | 1113     | 17.93      | 19.79      | 2         |
| PTC-MR         |                         |                     | Small molecules                    | 344      | 14.29      | 14.69      | 2         |
| PROTEINS/DD    |                         |                     | Bioinformatics (proteins)          | 1178     | 39.06      | 72.82      | 2         |
| COLLAB/B       |                         |                     | Bioinformatics (proteins)          | 188      | 284.32     | 715.66     | 2         |
| RDT-B          |                         |                     | Social networks                    | 5000     | 74.49      | 2457.78    | 2         |
| RDT-MSK        |                         |                     | Social networks                    | 2000     | 429.63     | 497.75     | 2         |
| IMDB-B         |                         |                     | Social networks                    | 4999     | 508.52     | 594.87     | 5         |
| BBBP           |                         |                     | Social networks                    | 1000     | 19.77      | 96.53      | 2         |
| Tox21          |                         |                     | Small molecules                    | 2039     | 24.05      | 25.94      | 12        |
| ToxCast        |                         |                     | Small molecules                    | 7831     | 18.51      | 25.94      | 12         |
| SIDER          |                         |                     | Small molecules                    | 8575     | 18.78      | 19.26      | 167       |
| ClinTox        |                         |                     | Small molecules                    | 1427     | 33.64      | 35.36      | 27        |
| MUV            |                         |                     | Small molecules                    | 1478     | 26.13      | 27.86      | 2         |
| HIV            |                         |                     | Small molecules                    | 90087    | 24.23      | 26.28      | 17        |
| BACE           |                         |                     | Small molecules                    | 41127    | 25.53      | 27.48      | 2         |
| CORA           |                         |                     | Citation network                   | 1        | 2,708      | 5,429      | 7         |
| CITESEER       |                         |                     | Citation network                   | 1        | 3,327      | 4,732      | 6         |
| PUBMED         |                         |                     | Citation network                   | 1        | 19,717     | 44,338     | 3         |
| Coauthor CS    |                         |                     | Citation network                   | 1        | 18,333     | 81,894     | 15        |
| Coauthor Phy.  |                         |                     | Citation network                   | 1        | 34,493     | 247,962    | 5         |
| Amazon Photos  |                         |                     | E-commerce network                 | 1        | 7,650      | 119,081    | 8         |
| Amazon Comp.   |                         |                     | E-commerce network                 | 1        | 13,752     | 245,661    | 10        |
| PPI            |                         |                     | Bioinformatics (proteins)          | 24       | 56,944     | 818,716    | 121       |
| Flickr         |                         |                     | Social network                     | 1        | 89,250     | 899,765    | 7         |
| Reddit         |                         |                     | Social network                     | 1        | 232,965    | 11,609,919 | 50        |

Unsupervised classification refers to performing unsupervised representation learning followed by linear classification.
small molecule datasets in TUDataset [102]. Traditional molecule classification datasets such as NCI1 [103] and MUTAG [104] are the most commonly used datasets for unsupervised graph representation learning in self-supervision related studies [10], [49]. In addition, the molecule property prediction models can be also trained in a self-supervised pre-training and finetuning fashion for semi-supervised learning and transfer learning. Recent works [43], [49], [56] build their pre-training molecule dataset by sampling unlabeled molecules from the ZINC15 [105] database containing 750 million molecules. MoleculeNet [21] also provides a collection of molecular graph datasets for molecule property prediction, which is suitable for downstream graph classification. Among all the datasets in MoleculeNet, the classification datasets such as BBBP, Tox21, and HIV are used for the evaluation of several self-supervised learning methods [43], [49], [56].

Protein Biological Function Prediction. The protein is a particular type of molecule but is represented differently by graph data. In a protein graph, nodes represent amino acids and an edge indicates the two connected nodes are less than 6 Angstroms apart. Datasets for chemical molecular property prediction are also categorized as bioinformatics datasets in TUDataset. Similar to the chemical molecular datasets, protein datasets can be used in both unsupervised representation learning, such as PROTEINS [106] and DD [107], and in the two-stage training.

Social Network Property Prediction. A social network graph dataset considers each entity (e.g. a user or an author) as a node and their social connections as edges. As social networks in different datasets represent differently, social network graph datasets are not typically used for transfer learning. Social network graph datasets used in recent self-supervised studies [48], [49] are typical datasets for graph classification [108] such as COLLAB, REDDIT-B and IMDB-B.

5.2 Node-Level Transductive Learning

Node-level learning tasks can be performed as transductive learning tasks on large graphs [38], where are nodes and the complete graph structure, together with labels of a portion of nodes, are available for training. The citation network datasets [109], including CORA [110], CITESEER [111] and PUBMED [112] are commonly used for node-level transductive learning. There are three typical ways to use the citation network datasets. Contrastive learning methods [10], [38], [41] are usually evaluated on the social network datasets by performing unsupervised representation learning followed by a linear classification with fixed representations, whereas predictive learning studies usually perform unsupervised representation learning followed by clustering [50], [51] or semi-supervised link prediction [39], [55] on the social network datasets.

Motivated by the concern that current GNN evaluations becomes saturated on above citation network datasets, Schuru et al. [113] construct four additional node-level datasets, Coauthor-CS, Coauthor-Physics from the Microsoft Academic Graph [114], Amazon-Photos, and Amazon-Computers from the Amazon Co-purchase Graph [115]. The four datasets contain larger graphs with more nodes and edges. And their learning tasks are hence more challenging compared to the citation network datasets.

5.3 Node-Level Inductive Learning

Node-level inductive learning performs training and testing on separate subsets. There are two typical ways to split the nodes for training and testing. First, in cases that all nodes are from the same large graph, a random subset of nodes are selected for testing and is masked out together with their edges during training, in contrast to the transductive learning where all nodes and the complete graph structure are used during training. Two commonly used node-level inductive learning datasets in the first case are Reddit [86] and Flickr [116]. Each node in Reddit represents a post and posts are connected by edges if they are commented by the same user. The node classification task is to predict which community the post belongs to. For the Flickr dataset, each node represents an image uploaded to Flickr and an edge between nodes indicates that they share common properties such as the same geographic location or commented by the same user. The task is to predict the class (based on tags) each image belongs to.

In cases that all nodes belong to separate graphs, the training and testing nodes are split by graphs. Zitnik and Leskovec [117] build a inductive learning dataset in the second case containing 395 K unlabeled protein obtained from protein–protein interaction (PPI) networks and perform fine-tuning on PPI networks consisting of 88 K proteins labeled with 40 fine-grained biological functions, obtained from Zitnik et al. [118]. Each node in a graph represents a protein and an edge indicates the existence of interaction between the proteins. The task of PPI is to predict the gene ontology sets a protein belongs to.

6 An Open-Source Library

We develop an open-source library DIG: Dive into Graphs [119], which includes a module, known as sslgraph, for self-supervised learning of GNNs. DIG-sslgraph is based on Pytorch [120] and Pytorch Geometric [121] and aims at easy implementation and standardized evaluation of SSL methods for graph neural networks. In particular, we provide a unified and highly customizable framework for contrastive learning methods, standardized data interface, and evaluation tools that are compatible for evaluating both contrastive methods and predictive methods. The overview of the library is shown in Supplementary Fig. 1, available online.

Given the developed unified contrastive framework as a base class, particular contrastive learning methods can be easily implemented by specifying their objective, functions for view generation, and their encoders. We also pre-implement four state-of-the-art contrastive methods for either node-level or graph-level tasks based on the unified framework, including InfoGraph [41], GRACE [46], MVGRL [58], and GraphCL [49]. The provided data interface includes datasets from TUDataset [102] and the citation network dataset [109]. Other datasets from Pytorch Geometric and new datasets processed by Pytorch Geometric are also supported by our data interface. The provided evaluation tools and data interface allow standardized evaluations of SSL methods and fair comparisons with existing SSL methods under
common evaluation settings, including unsupervised graph-level representation learning, semi-supervised graph classification (or transfer learning, depending on the datasets), and unsupervised node-level representation learning. Altogether, our open-source library provides a complete and extensible framework for developing and evaluating SSL methods for GNNs. To show the efficiency and effectiveness of the DIG-sslgraph library, we compare the training time, memory consumption, and downstream accuracy of four SSL methods on multiple datasets between the original implementations and DIG counterparts. The results are shown and discussed in Appendix G, available in the online supplemental material.

7 CHALLENGES AND FUTURE DIRECTIONS

While existing SSL approaches have shown promising effectiveness on learning from graph data, there still exist several challenges due to the more complicated structure and more diverse tasks of graphs. In this section, we discuss the remaining challenges as well as potential directions for future studies on Graph self-supervised learning.

The optimal views generation w.r.t. specific downstream tasks are still unclear for contrastive methods. The performance of the learned representation or pre-trained model on down-stream tasks heavily depends on the selection of transformations to generate views. The optimal view generation also depends on specific downstream tasks. However, there is still no way to obtain the optimal view for each down-stream task even the task is available. Several studies have explored approaches to utilizing better views for contrast based on adversarial learning [81] or searching [82], but views generated by the two approaches are still not optimal due to their limited search space for graph transformations. In particular, Xu et al. [82] only considers a finite set of graph transformations whose search space is also limited. In addition, Suresh et al. [81] only considers parameterized structural transformations, and more complicated learnable view generation involving feature-space, structure-space, and sampling-based transformation are still challenging and are limited by the development of graph generation studies [122], [123], [124], [125]. Moreover, the above methods require available downstream tasks at the pretraining stage. They become inapplicable when downstream tasks are unavailable or in the unsupervised representation learning setting. Hence it is also desired to study universally optimal views under the downstream task-agnostic setting.

There is no unified theory or theoretical framework for predictive methods. Unlike contrastive methods grounded by the problem of mutual information maximization, the predictive methods, especially the graph property prediction and invariance regularization-based methods, utilize different pretext learning tasks motivated by individual hypotheses and based on empirical studies. However, they lack guidance from unified theoretical frameworks to design specific pretext tasks for different downstream tasks. The information bottleneck principle may be used to interpret the effectiveness of several predictive methods but further study and investigation are desired.

Richer domain knowledge can be better utilized as self-supervision. For graph machine learning tasks oriented by other research areas such as biomedical researches and quantum physics, existing constraints and rules from domain knowledge naturally contain rich supervision benefiting the learning of downstream tasks. Including domain knowledge has shown to be effective for both contrastive methods [126] and predictive methods [56]. Currently, only simple domain knowledge-based tasks such as the motif prediction [56] are adopted. Future studies on designing novel tasks better utilizing domain knowledge such as functional groups and quantum mechanisms are promising directions.

Scaling-up and efficiency issues are to be addressed. Many existing SSL approaches suffer from memory issues and computing efficiency issues in terms of time consumption when scaled up to larger graphs. For contrastive methods, the scaling-up issue becomes more critical as their performance usually relies on a larger number of samples in a mini-batch. In addition, as the contrastive framework requires computing representations of multiple views, their memory consumption is times higher than predictive approaches. When the computing of view generation for contrastive methods or the graph properties computation for predictive methods is heavy, the computation time may increase sharply as the graph scales up. The above issues prevent the application of existing methods to extremely large graphs in industrial scenarios or other research areas (e.g., protein networks and particles in materials). Currently, studies addressing the scaling-up issue for SSL methods are still lacking and less explored.

Explainability of SSL for GNN requires further studies. The explainability for GNNs is critical in multiple application scenarios to assure the reliability and security of GNN models. For example, in drug discovery, it is important to understand which functional group in a molecular graph leads to the GNN decision for a property prediction. A survey work [127] provides a thorough introduction to existing explanation methods for GNNs. However, existing SOTA studies focus on the explanation under supervised setting and require downstream tasks to perform explanation, and only limited methods, such as gradient-based methods, can be adapted to explain pre-trained GNN encoders without given the downstream prediction head. A recent work [128] proposes the task-agnostic explanation framework to enable high-quality GNN explanations without downstream tasks. The task-agnostic framework can be utilized to explain GNNs trained through SSL. More studies and investigations are desired in this direction.

8 CONCLUSION

Despite recent successes in natural language processing and computer vision, the self-supervised learning applied to graph data is still an emerging field and has significant challenges to be addressed. Existing methods employ self-supervision to graph neural networks through either contrastive learning or predictive learning. We summarize current self-supervised learning methods and provide unified reviews for the two approaches. We unify existing contrastive learning methods for GNNs with a general contrastive framework, which performs mutual information maximization on different views of given graphs with appropriate MI estimators. We demonstrate that any existing method can be realized by determinating its MI estimator, views generation, and graph
encoder. We further provide detailed descriptions of existing options for the components and discussions to guide the choice of those components. For predictive learning, we categorize existing methods into graph reconstruction, property prediction, and self-training based on how labels are generated from the data. A thorough review is provided for methods in all three types of predictive learning. Finally, we summarize common graph datasets of different domains and introduce what learning tasks the datasets are involved in to provide a clear view to conduct future evaluation experiments. Altogether, our unified treatment of SSL in GNNs in terms of methodologies, datasets, evaluations, and open-source software is anticipated to foster methodological developments and facilitate empirical comparisons.

Acknowledgments

The authors would like to thank the scientific community for providing valuable feedback and comments, which lead to improvements of this work.

References

[1] Y. Yang and Z. Xu, “Rethinking the value of labels for improving class-imbalanced learning,” in Proc. 34th Int. Conf. Adv. Neural Inf. Process. Syst., 2020, pp. 19290–19301.
[2] V. Lempitsky, A. Vedaldi, and D. Ulyanov, “Deep image prior,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9446–9454.
[3] Y. Xie, Z. Wang, and S. Ji, “Nois2Same: Optimizing a self-supervised bound for image denoising,” in Proc. 34th Int. Conf. Adv. Neural Inf. Process. Syst., 2020, pp. 20320–20330.
[4] S. Laine, T. Karras, J. Lehtinen, and T. Aila, “High-quality self-supervised deep image denoising,” Adv. Neural Inf. Process. Syst., vol. 32, pp. 6970–6980, 2019.
[5] A. Krull, T.-O. Buchholz, and F. Jug, “Nois2Void-learning denoising from single noisy images,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 2124–2132.
[6] J. Batson and L. Royer, “Nois2Self: Blind denoising by self-supervision,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 524–533.
[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2019, pp. 4171–4186.
[8] J. Wu, N. Xue, and X. Wang, “Self-supervised dialogue learning,” in Proc. 57th Ann. Meeting Assoc. Comput. Linguistics, 2019, pp. 3857–3867.
[9] H. Wang et al., “Self-supervised learning for contextualized extractive summarization,” in Proc. 57th Ann. Meeting Assoc. Comput. Linguistics, 2019, pp. 2221–2227.
[10] K. Hassan and A. H. Khasahmadi, “Contrastive multi-view representation learning on graphs,” in Proc. 37th Int. Conf. Mach. Learn., 2020, pp. 4116–4126.
[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 1597–1607.
[12] A. V. D. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” 2018, arXiv:1807.03748.
[13] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” 2019, arXiv:1906.05849.
[14] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2014, pp. 701–710.
[15] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, “graph2vec: Learning distributed representations of graphs,” 2017, arXiv:1707.05005.
[16] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller, “SQR: Self-supervised spectral graph representation learning,” 2018, arXiv:1811.06237.
[17] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 1422–1430.
[18] M. Norouzi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 69–84.
[19] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 9726–9735.
[20] J. M. Girid et al., “Bootstrap your own latent - a new approach to self-supervised learning,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 21271–21284.
[21] Z. Wu et al., “Moleculenet: A benchmark for molecular machine learning,” Chem. Sci., vol. 9, no. 2, pp. 513–530, 2018.
[22] Z. Wang et al., “Advanced graph and sequence neural networks for molecular property prediction and drug discovery,” 2020, arXiv:2002.01983.
[23] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2017.
[24] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” Int. Conf. Learn. Representations, 2019.
[25] H. Gao and S. Ji, “Graph U-nets,” in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 2083–2092.
[26] M. Liu, Z. Wang, and S. Ji, “Non-local graph neural networks,” 2020, arXiv:2005.14612.
[27] L. Cai, J. Li, J. Wang, and S. Ji, “Line graph neural networks for link prediction,” 2020, arXiv:2010.10046.
[28] M. Liu, H. Gao, and S. Ji, “Towards deeper graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. X343–X351.
[29] L. Cai and S. Ji, “A multi-scale approach for graph link prediction,” in Proc. 34th AAAI Conf. Artif. Intell., 2020, pp. 3308–3315.
[30] H. Gao, Z. Wang, and S. Ji, “Large-scale learnable graph convolutional networks,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 1416–1424.
[31] H. Gao, Y. Chen, and S. Ji, “Learning graph pooling and hybrid convolutional operations for text representations,” in Proc. Web Conf., 2019, pp. 2743–2749.
[32] H. Gao, Y. Liu, and S. Ji, “Topology-aware graph pooling networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 12, pp. 4512–4518, Dec. 2021.
[33] Z. Wang and S. Ji, “Second-order pooling for graph neural networks,” IEEE Trans. Pattern Anal. Mach. Intell., early access, Jun. 01, 2020, doi: 10.1109/TPAMI.2020.2999032.
[34] Y. Yuan and S. Ji, “StructPool: Structured graph pooling via conditional random fields,” in Proc. 8th Int. Conf. Learn. Representations, 2020.
[35] H. Gao and S. Ji, “Graph representation learning via hard and channel-wise attention networks,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2019, pp. 741–749.
[36] H. Yuan, J. Tang, X. Hu, and S. Ji, “XGNN: Towards model-level explanations of graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 430–438.
[37] Y. Liu, H. Yuan, L. Cai, and S. Ji, “Deep learning of high-order interactions for protein interface prediction,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 679–687.
[38] P. Velickovic, W. Fedus, W. L. Hamilton, P. Lio, Y. Bengio, and D. Hjelm, “Deep graph infomax,” in Int. Conf. Learn. Representations, 2019.
[39] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” 2016, arXiv:1611.07308.
[40] Y. Ma and J. Tang, Deep Learn. on Graphs. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[41] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, “Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization,” in Proc. Int. Conf. Learn. Representations, 2019.
[42] Z. Peng et al., “Graph representation learning via graphical mutual information maximization,” in Proc. Web Conf., 2020, pp. 259–270.
[43] W. Hu et al., “Strategies for pre-training graph neural networks,” in Proc. Int. Conf. Learn. Representations, 2020.
[44] Y. Yao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu, “Sub-graph contrast for scalable self-supervised graph representation learning,” in Proc. IEEE Int. Conf. Data Mining, 2020, pp. 222–231.
[45] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, “Graph contrastive learning with adaptive augmentation,” in Proc. Web Conf., 2021, pp. 2069–2080.
et al., “Deep metric learning using triple network,” in Proc. Int. Workshop Similarity-Based Pattern Recognit., 2015, pp. 84–92.

P. Khosla et al., “Supervised contrastive learning,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 18661–18673.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “BPR: Bayesian personalized ranking from implicit feedback,” in Proc. 25th Conf. Uncertainty Artif. Intell., 2009, pp. 452–461.

X. Jiang, T. Jia, Y. Fang, C. Shi, Z. Lin, and H. Wang, “Pre-training on large-scale homogeneous graph,” in Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining, 2021, pp. 756–766.

X. Wang, N. Liu, H. Han, and C. Shi, “Self-supervised heterogeneous graph neural network with co-contrastive learning,” in Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining, 2021, pp. 12310–12320.

K. Sohn, “Improved deep metric learning with multi-class n-pair loss objective,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 1857–1865.

F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 815–823.

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, “Barlow twins: Self-supervised learning via redundancy reduction,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 12310–12320.

M. Tognin and M. F. Tommasi, “Canonical correlation analysis: An overview with application to learning methods,” Neurocomput., vol. 16, no. 12, pp. 2639–2664, 2004.

M. Aytekin, F. Cour, J. Dambre, X. Guo, and W. Zhang, “Self-supervised learning of complex motion patterns,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 12310–12320.

K. Sohn, “Improved deep metric learning with multi-class n-pair loss objective,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 1857–1865.

F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. 815–823.

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, “Barlow twins: Self-supervised learning via redundancy reduction,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 12310–12320.

Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, “Graph contrastive learning with augmentations,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 5812–5823.

C. Wang, S. Fan, G. Long, X. Zhu, and J. Jiang, “Mgae: Marginalized autoencoder for graph clustering,” in Proc. ACM Conf. Knowl. Discov. Data Mining, 2020, pp. 889–899.

J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, “Symmetric graph convolutional autoencoder for unsupervised graph representation learning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 6518–6527.

S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, “Adversarially regularized contrastive learning,” in Proc. 27th Int. Conf. Artif. Intell., 2018, pp. 2659–2615.

A. Hasanzadeh, E. Hajiramezanali, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, “Semi-implicit graph variational auto-encoders,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 10712–10723.

Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, “GPT-GNN: Generative pre-training of graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 1866–1877.

Z. Peng, Y. Dong, M. Luo, X.-M. Wu, and Q. Zheng, “Self-supervised graph representation learning via global context prediction,” 2020, arXiv:2003.01604.

Y. Kong et al., “Self-supervised graph transformer on large-scale molecular data,” in Proc. 34th Int. Conf. Adv. Neural Inf. Process. Syst., 2020, pp. 12539–12571.

D. Hwang, J. Park, S. Kwon, K.-M. Kim, J.-W. Ha, and H. J. Kim, “Self-supervised auxiliary learning with meta-paths for heterogeneous graphs,” 2020, arXiv:2007.08294.

K. Sun, Z. Lin, and Z. Zhu, “Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 5892–5899.

Z. Hu, G. Kou, H. Zhang, N. Li, K. Yang, and L. Liu, “Rectifying pseudo labels: Iterative feature clustering for graph representation learning,” in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage., 2021, pp. 720–729.

H. Zhang, Q. Wu, J. Yan, D. Wipf, and S. Y. Philip, “From canonical correlation analysis to self-supervised graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 76–89.

Y. Xie, Z. Xu, and S. Ji, “Self-supervised representation learning via latent graph prediction,” 2022, arXiv:2202.08333.

X. Li et al., “Self-supervised learning: Generative or contrastive,” 2020, arXiv:2006.08218.

Y. Liu, S. Pan, M. Jin, C. Zhou, F. Xia, and P. S. Yu, “Graph self-supervised learning: A survey,” 2021, arXiv:2103.00111.

Y. Wang, J. Zhang, S. Guo, H. Yin, C. Li, and H. Chen, “Decoupling representation learning and classification for GNN-based anomaly detection,” in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2021, pp. 1239–1248.

W. Jin et al., “Self-supervised learning on graphs: Deep insights and new direction,” 2020, arXiv:2006.10141.

Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 1180–1189.

Y. Sun, E. Tzeng, T. Darrell, and A. A. Efros, “Unsupervised domain adaptation through self-supervision,” 2019, arXiv:1909.11825.

R. D. Hjelm et al., “Learning deep representations by mutual information estimation and maximization,” in Proc. Int. Conf. Learn. Representations, 2019.

M. D. Donsker and S. R. S. Varadhan, “Asymptotic evaluation of correlation analysis: An overview with application to learning methods,” Neurocomput., vol. 16, no. 12, pp. 2639–2664, 2004.

M. Aytekin, F. Cour, J. Dambre, X. Guo, and W. Zhang, “Self-supervised learning of complex motion patterns,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 12310–12320.

Y. You, T. Chen, Z. Wang, and Y. Shen, “When does self-supervision help graph convolutional networks?,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 10871–10880.

I. Goodfellow et al., “Generative adversarial nets,” in Proc. Adv. Neural Inf. Process. Syst., 2014, pp. 2672–2680.

W. Jin, R. Barzilay, and J. Taik, “Junction tree variational autoencoder for molecular graph generation,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 2323–2332.

J. Li, T. Y. J. Li, H. Zhang, K. Zhao, Y. Rong, and H. Cheng, “Difficult graph variational autoencoder,” in Proc. 34th Int. Conf. Adv. Neural Inf. Process. Syst., 2020, pp. 5274–5283.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” OpenAI Blog, vol. 1, no. 8, 2019, Art. no. 9.

H. Hotelling, “Relations between two sets of variates,” in Breakthroughs in Statistics, Berlin, Germany: Springer, 1992, pp. 162–190.

I. T. Jolliffe and J. Shrowt-Taylor, “Canonical correlation analysis: An overview with application to learning methods,” Neurocomput., vol. 16, no. 12, pp. 2639–2664, 2004.

J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, “Barlow twins: Self-supervised learning via redundancy reduction,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 12310–12320.
XIE ET AL.: SELF-SUPERVISED LEARNING OF GRAPH NEURAL NETWORKS: A UNIFIED REVIEW

2429

[99] N. Tishby, F. C. Pereira, and W. Bialek, “The information bottleneck method,” in Proc. 37th Annul. Allerton Conf. Commun., Control Comput., 1999, pp. 368–377.

[100] Q. Li, Z. Han, and X.-M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in Proc. AAAI Conf. Intell. Syst. Appl., 2021, pp. 5358–5364.

[101] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 132–149.

[102] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, “Tudataset: A collection of benchmark datasets for learning with graphs,” in Proc. ICDM Workshop Graph Representation Learn. Beyond, 2020.

[103] N. Wale and G. Karypis, “Comparison of descriptor spaces for chemical compound retrieval and classification,” in Proc. 6th Int. Conf. Data Mining, 2006, pp. 678–689.

[104] A. K. Deb Nath, R. L. Lopez de Compadre, G. Deb Nath, A. J. Shusterman, and C. Hansch, “Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds correlation with molecular orbital energies and hydrophobicity,” J. Med. Chem., vol. 34, no. 2, pp. 786–797, 1991.

[105] T. Sterling and J. J. Irwin, “Zinc 15-ligand discovery for everyone,” J. Chem. Inf. Model., vol. 55, no. 11, pp. 2324–2337, 2015.

[106] K. M. Borgwardt, C. S. Ong, S. Schoning, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, “Protein function prediction via graph kernels,” Bioinf., vol. 21, pp. i47–i56, 2005.

[107] F. Dobson and A. J. Doig, “Distinguishing enzyme structures from non-enzymes without alignments,” J. Mol. Biol., vol. 330, no. 4, pp. 771–783, 2003.

[108] P. Yanardag and S. Vishwanathan, “Deep graph kernels,” in Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2015, pp. 1365–1374.

[109] Z. Yang, W. Cohen, and R. Salakhudinov, “Revisiting semi-supervised learning with graph embeddings,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 40–48.

[110] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Automating the construction of internet portals with machine learning,” Inf. Retrieval, vol. 3, no. 2, pp. 127–163, 2000.

[111] C. L. Giles, K. D. Bollacker, and S. Lawrence, “Citeseer: An automatic citation indexing system,” in Proc. 3rd ACM Conf. Digit. Libraries, 1998, pp. 89–98.

[112] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, “Collective classification in network data,” AI Mag., vol. 29, no. 3, pp. 93–93, 2008.

[113] O. Schur, M. Mumme, A. Bochevski, and S. Gunnewann, “Pitfalls of graph neural network evaluation,” 2018, arXiv:1811.05868.

[114] A. Sinha et al., “An overview of microsoft academic service (MAS) and applications,” in Proc. 24th Int. Conf. World Wide Web, 2015, pp. 243–246.

[115] J. McAuley, C. Targett, Q. Shi, and A. Van DenHengel, “Image-based recommendations on styles and substitutes,” in Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2015, pp. 43–52.

[116] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, “GraphSAINT: Graph sampling based inductive learning method,” in Proc. Int. Conf. Learn. Representations, 2020.

[117] M. Zitnik and J. Leskovec, “Predicting multicellular function through multi-layer tissue networks,” Bioinf., vol. 33, no. 14, pp. i190–i198, 2017.

[118] M. Zitnik et al., “Evolution of resilience in protein interactomes across the tree of life,” Proc. Natl. Acad. Sci. USA, vol. 116, no. 10, pp. 4426–4433, 2019.

[119] M. Liu et al., “DGL: A turnkey library for diving into graph deep learning research,” J. Mach. Learn. Res., vol. 22, no. 240, pp. 1–9, 2021.

[120] A. Paszke et al., “Pytorch: An imperative style, high-performance deep learning library,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 8024–8035.

[121] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch geometric,” in Proc. ICLR Workshop Representation Learning on Graphs Mutifolds, 2019.

[122] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, “GraphRNN: Generating realistic graphs with deep auto-regressive models,” in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 5708–5717.

[123] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang, “Graphaf: A flow-based autoregressive model for molecular graph generation,” in Proc. Int. Conf. Learn. Representations, 2020.

[124] C. Zhang and F. Wang, “Mflow: An invertible flow model for generating molecular graphs,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 617–626.

[125] Y. Luo, K. Yan, and S. Ji, “Graphdf: A discrete flow model for molecular graph generation,” in Proc. 38th Int. Conf. Mach. Learn., 2020, pp. 7192–7203.

[126] P. Li et al., “Pairwise half-graph discrimination: A simple graph-level self-supervised strategy for pre-training graph neural networks,” in Proc. 13th Int. Joint Conf. Artif. Intell., 2021, pp. 2694–2700.

[127] H. Yuan, H. Yu, S. Gui, and S. Ji, “Explainability in graph neural networks: A taxonomic survey,” 2020, arXiv:2012.15445.

[128] Y. Xie et al., “Task-agnostic graph explanations,” 2022, arXiv:2202.08335.

Yaochen Xie received the BS degree in statistics from the University of Science and Technology of China, Hefei, China, in 2018. Currently, she is working towards the PhD degree with the Department of Computer Science and Engineering, Texas A&M University. Her research interests include machine learning, deep learning, and graph data mining.

Zhao Xu received the MS degree in biomedical engineering from the University of Michigan, Ann Arbor, Michigan, in 2017. Currently, she is working toward the PhD degree with the Department of Computer Science and Engineering, Texas A&M University. Her research interests include machine learning, deep learning, and data mining.

Jingtun Zhang received the BSc degree in computer science and technology from the University of Science and Technology of China, Hefei, China, in 2020. Currently, he is working toward the MSc degree with the Department of Computer Science and Engineering, Texas A&M University, College Station, Texas. His research interests include machine learning, software engineering, and data mining.

Zhengyang Wang received the MSc degree in machine learning and computer science from New York University, New York City, New York, in 2015, and the PhD degree in computer science from Texas A&M University, College Station, Texas. He is currently an applied scientist with Amazon.com LLC. His research interests include machine learning, deep learning, and data mining.

Shuivang Ji (Senior Member, IEEE) received the PhD degree in computer science from Arizona State University, Tempe, Arizona, in 2010. Currently, he is a professor and presidential impact fellow with the Department of Computer Science and Engineering, Texas A&M University, College Station, Texas. His research interests include machine learning, deep learning, graph and image analysis, and quantum systems. He received the National Science Foundation CAREER Award, in 2014. He is currently an associate editor for IEEE Transactions on Pattern Analysis and Machine Intelligence, ACM Transactions on Knowledge Discovery from Data, and ACM Computing Surveys. He regularly serves as an area chair or equivalent roles for data mining and machine learning conferences, including AAAI, ICLR, ICLM, UJCAI, KDD, and NeurIPS.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.