Self-supervised on Graphs: Contrastive, Generative,or Predictive

Lirong Wu, *Student Member, IEEE,* Haitao Lin, Zhangyang Gao,
Cheng Tan, and Stan.Z.Li, *Fellow, IEEE*

Abstract—Deep learning on graphs has recently achieved remarkable success on a variety of tasks while such success relies heavily on the massive and carefully labeled data. However, precise annotations are generally very expensive and time-consuming. To address this problem, self-supervised learning (SSL) is emerging as a new paradigm for extracting informative knowledge through well-designed pretext tasks without relying on manual labels. In this survey, we extend the concept of SSL, which first emerged in the fields of computer vision and natural language processing, to present a timely and comprehensive review of the existing SSL techniques for graph data. Specifically, we divide existing graph SSL methods into three categories: contrastive, generative, and predictive. More importantly, unlike many other surveys that only provide a high-level description of published research, we present an additional mathematical summary of the existing works in a unified framework. Furthermore, to facilitate methodological development and empirical comparisons, we also summarize the commonly used datasets, evaluation metrics, downstream tasks, and open-source implementations of various algorithms. Finally, we discuss the technical challenges and potential future directions for improving graph self-supervised learning. Latest advances in graph SSL are summarized in a GitHub repository https://github.com/LirongWu/awesome-graph-self-supervised-learning.

Index Terms—Deep Learning, Self-supervised Learning, Graph Neural Networks, Unsupervised Learning, Survey.

1 Introduction

In recent years, deep learning on graphs has emerged as a popular research topic for the AI research community, due to their ability to naturally capture both graph structures and node/edge features. Various graph learning architectures [1–10] have been proposed and demonstrated their superior performance in various graph-related tasks, such as node classification, graph regression, and link prediction. However, most of works have focused on supervised or semi-supervised learning settings, where the model is trained by specific downstream tasks with abundant labeled data, which are often limited, expensive, and inaccessible. Due to the heavy reliance on the number and quality of labels, these supervised or semi-supervised methods are hardly applicable to real-world scenarios, especially those requiring expert knowledge for annotation, such as medicine, meteorology, etc. More importantly, these methods are prone to suffer from problems of over-fitting, poor generalization, and weak robustness [11].

Recent advances in SSL [12–22] have provided novel insights into reducing the dependency on excessive annotated labels and enable the training on massive unlabeled data. The primary goal of SSL is to learn transferable prior knowledge from abundant unlabeled data with well-designed pretext tasks and then generalize the learned knowledge to downstream tasks with specific supervision signals. Recently, SSL has shown its promising capability in the field of computer vision (CV) [12–17] and natural language processing (NLP) [18–22] for various tasks, such as image classification [12, 14, 23], image denoising [24–26], and text classification [18, 20, 21]. The self-supervised training with well-designed pretext tasks helps the model learn more generalized representations from abundant unlabeled data, resulting in better performance and generalization on downstream tasks. For example, Moco [12] and SimCLR [13] augment the data by various means and then train CNNs to capture dependencies between different augmentations of a sample. Besides, BERT [18] pre-trains the model with Masked LM and Next Sentence Prediction as pretext tasks, achieving state-of-the-art performance on up to 11 tasks compared to those conventional methods. Inspired by the success of SSL in CV and NLP, it is naturally an important and promising direction to apply SSL to graph domains to fully exploit graph structure information and rich unlabeled data. However, SSL on graphs has not been fully explored so far, and many important issues remain to be addressed.

Compared with image and language sequence data, the application of SSL to graph domains is of great importance and also has great potential research prospects. Firstly, along with node features and partially known labels, the graph data contains underlying graph structures, where a large number of pretext tasks can be designed to simultaneously capture the intrinsic relations of nodes. Secondly, real-world graphs are usually formed following specific rules, e.g., links between atoms in molecular graphs are bounded by valence bond theory. Thus, extensive expert knowledge can be incorporated as a priori into the design of pretext tasks. Finally, graph-structured data generally support transductive learning [3], such as node classification tasks, which means that the features of Train/Val/Test samples are all available during the training process, which makes it possible to design more feature-related pretext tasks. There has been some recent work pointing out that pre-trained mod-

• *Lirong Wu, Haitao Lin, Zhangyang Gao, Cheng Tan and Stan.Z.Li are with the AI Lab, School of Engineering, Westlake University, HangZhou 310000, China. E-mail: {wulirong, linhaitao, gaozhangyang, tancheng, stan.zq.li}@westlake.edu.cn*

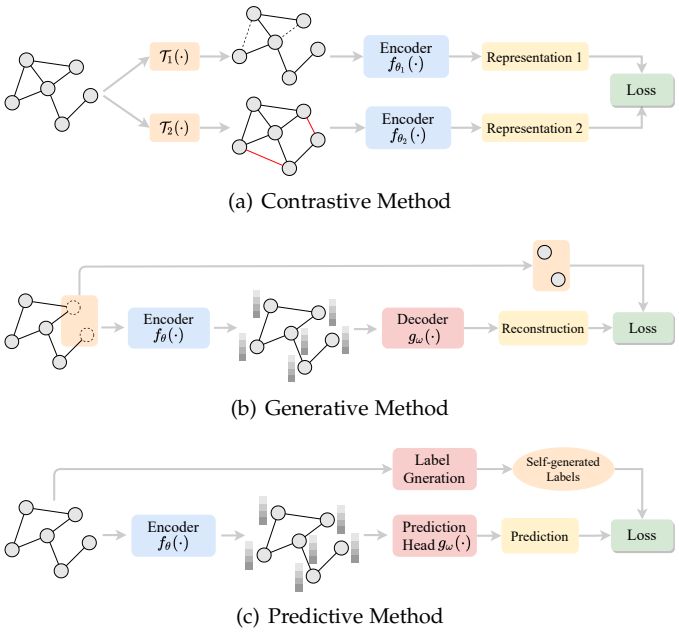


Fig. 1. A comparison among the contrastive, generative, and predictive method. **(a)**: the contrastive method contrasts the views generated from different augmentation $\mathcal{T}_1(\cdot)$ and $\mathcal{T}_2(\cdot)$. The information about the differences and sameness between (inter-data) data-data pairs are used as self-supervision signals. **(b)**: the generative method focuses on the (intra-data) information embedded in the graph, generally based on prtext tasks such as reconstruction, which exploit the attributes and structure of the graph itself as self-supervision signals. **(c)**: the predictive method generally self-generates labels some simple statistical analysis or expert knowledge, and designs prediction-based pretext tasks based on self-generated labels to handle the data-label relationship.

els can produce higher accuracy scores on unseen nodes, suggesting that pre-training is better suited for inductive learning than re-training from scratch [27].

Though some works have been proposed recently to apply SSL to graph domains and have achieved remarkable success [28–38], it is still very challenging to deal with the inherent differences between grid-like and structured-like data. Firstly, the topology of the image is a fixed grid and the text is a simple sequence, while graphs are not restricted to these rigid structures. Secondly, unlike the assumption of independent and identical sample distribution in image and text domains, nodes in the graph are correlated with each other rather than completely independent. This inspires us to design pretext tasks by considering both node attributes and graph structures. Finally, there exists a gap between self-supervised pretext tasks and downstream tasks due to the divergence of their optimization objectives. Inevitably, such divergence will significantly hurt the generalization of learned models. Therefore, it is crucial to reconsider the objectives of pretext tasks to better match that of downstream tasks and make them consistent with each other.

In this survey, we extend the concept of SSL, which first emerged in the fields of computer vision and natural language processing, to present a timely and comprehensive review of the existing SSL techniques for graph data. Specifically, we divide existing graph SSL methods into three categories: contrastive, generative, and predictive as shown in Fig. 1. More importantly, we provide an additional *mathematical summary* of existing works in a unified framework and summarize the commonly used datasets, evaluation metrics, downstream tasks, and open-source implementations of various algorithms. The core contributions of this survey are as follow:

- Present comprehensive and up-to-date reviews on existing graph SSL methods and divide them into three categories: contrastive, generative, and predictive, to improve clarity and accessibility of methods.
- Summarize the core mathematical ideas of recent research in graph SSL within a unified framework.
- Summarize the commonly used datasets, evaluation metrics, downstream tasks, and open-source implementations of surveyed graph SSL methods, setting the stage for developments of future works.
- Point out the technical limitations of current research and discuss promising directions on graph SSL.

Compared to the existing surveys on SSL [11], we purely focus on SSL for graph data and present a more mathematical review on the recent progress from the year 2019 to 2021. Though there have been two surveys on graph SSL, we argue that they are immature work with various flaws and shortcomings. For example, [39] clumsily *describes each method in 1-2 sentences*, lacking deep insight into the mathematical ideas and implementation details behind. Moreover, the number of reviewed methods in [40] are even fewer than half of ours, as it spends too much description on those *less important* contents, but ignores the core of graph SSL, i.e. the design of the pretext task. Compared with [39, 40], we build a unified framework focusing on *mathematical ideas*, code implementations, and evaluation protocols, and our advantages are as follow: (1) more mathematical details, striving to summarize each method with one equation; (2) more implementation details, including 42 datasets statistics (*vs* 20 datasets in [39]), evaluation metrics, and open-source code; (3) more fine-grained, clarified and rational taxonomy; (4) more surveyed works, 69 methods (*vs* 47 methods in [40] *vs* 18 methods in [39]); (5) more up-to-data review, with almost all surveyed works published after 2019.

2 Problem Statement

2.1 Notions and Definitions

Unless particularly specified, the notations used in this survey are illustrated in Table 1.

Definition 1 (Graph): We use $g = (\mathcal{V}, \mathcal{E})$ to denote a graph where $\mathcal{V}$ is the set of N nodes and $\mathcal{E}$ is the set of M edges. Let $v_i \in \mathcal{V}$ denote a node and $e_{i,j}$ denote an edge between node v_i and v_j. The l-hop neighborhood of a node v_i is denoted as $\mathcal{N}_i^{(l)} = \{v_j \in \mathcal{V} | d(v_i, v_j) \leq l\}$ where $d(v_i, v_j)$ is the shortest path length between node v_i and v_j. In particular, the 1-hop neighborhood of a node v_i is denoted as $\mathcal{N}_i = \mathcal{N}_i^{(1)} = \{v_j \in \mathcal{V} | e_{i,j} \in \mathcal{E}\}$. The graph structure can also be represented by an adjacency matrix $\mathbf{A} \in [0, 1]^{N \times N}$ with $A_{i,j} = 1$ if $e_{i,j} \in \mathcal{E}$ and $A_{i,j} = 0$ if $e_{i,j} \notin \mathcal{E}$.

Definition 2 (Attribute Graph): Attributed graph, an opposite concept to the unattributed one, refers to a graph where nodes or edges are associated with their own features (a.k.a attributes). For example, each node v_i in graph g may be associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^{d_0}$, such graph is

referred to an attributed graph $g = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ or $g = (\mathbf{A}, \mathbf{X})$, where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ is the node feature matrix. Meanwhile, an attributed graph $g = (\mathcal{V}, \mathcal{E}, \mathbf{X}^e)$ may have edge attributes $\mathbf{X}^e$, where $\mathbf{X}^e \in \mathbb{R}^{M \times b_0}$ is an edge feature matrix with $\mathbf{x}^e_{i,j} \in \mathbb{R}^{b_0}$ being the feature vector of edge $e_{i,j}$.

Definition 3 (Dynamic Graph): A dynamic graph is a special attributed graph where the node set, graph structure and node attributes may change dynamically over time. The dynamic graph is formalized as $g = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)}, \mathbf{X}^{(t)})$ or $g = (\mathbf{A}^{(t)}, \mathbf{X}^{(t)})$, where $\mathcal{E}^{(t)}$ represents the edge set at time step t and $\mathbf{A}^{(t)}_{i,j} = 1$ denotes a interaction between node v_i and v_j at time step t ($1 \leq t \leq T$).

Definition 4 (Heterogeneous Graph): Consider a graph $g = (\mathcal{V}, \mathcal{E})$ with a node type mapping function $f_v : \mathcal{V} \rightarrow \mathcal{Y}^v$ and an edge type mapping function $f_e : \mathcal{E} \rightarrow \mathcal{Y}^e$, where $\mathcal{Y}^v$ is the set of node types and $\mathcal{Y}^e$ is the set of edge types. For a graph with more than one type of node or edge, e.g. $|\mathcal{Y}^v| > 1$ or $|\mathcal{Y}^e| > 1$, we denote it as a heterogeneous graph, otherwise, it is a homogeneous graph. There are some special types of heterogeneous graphs: a bipartite graph with $|\mathcal{Y}^v| = 2$ and $|\mathcal{Y}^e| = 1$, and a multiplex graph with $|\mathcal{Y}^v| = 1$ and $|\mathcal{Y}^e| > 1$.

Definition 5 (Spatial-Temporal Graph): A spatial-temporal graph is a special dynamic graph, but noly the node attributes change over time with the node set and graph structure unchanged. The spatial-temporal graph is defined as $g = (\mathcal{V}, \mathcal{E}, \mathbf{X}^{(t)})$ or $g = (\mathbf{A}^{(t)}, \mathbf{X}^{(t)})$, where $\mathbf{X}^{(t)} \in \mathbb{R}^{N \times d_0}$ is node feature matrix at the time step t ($1 \leq t \leq T$).

2.2 Downstream Tasks

The downstream tasks on the graph can be divided into three categories: node-level, link-level, and graph-level tasks. A node-level graph encoder $f_\theta(\cdot)$ (e.g., based on a neural network) is often used to generate node embeddings for each node, or a graph-level graph encoder $f_\gamma(\cdot)$ is used to generate graph-level embeddings. Finally, the embeddings are fed into an optional prediction head $g_\omega(\cdot)$ to perform specific downstream tasks.

2.2.1 Node-level tasks

Node-level tasks focus on the properties of nodes, so node representation is indispensable to these tasks. Node classification is a typical node-level task where only a subset of node $\mathcal{V}_L$ with corresponding labels $\mathcal{Y}_L$ are known, and we denote the labeled data as $\mathcal{D}_L = (\mathcal{V}_L, \mathcal{Y}_L)$ and unlabeled data as $\mathcal{D}_U = (\mathcal{V}_U, \mathcal{Y}_U)$. Let $f_\theta : \mathcal{V} \rightarrow \mathcal{Y}$ be a graph encoder trained on labeled data $\mathcal{D}_L$ so that it can be used to infer the labels $\mathcal{Y}_U$ of unlabeled data. Thus, the objective function for node classification can be defined as minimizing loss $\mathcal{L}_{node}$,

$$\min_{\theta,\omega} \mathcal{L}_{node}\left(\mathbf{A}, \mathbf{X}, \theta, \omega\right) = \sum_{(v_i, y_i) \in \mathcal{D}_L} \ell\big(g_\omega(\mathbf{h}_i), y_i\big) \tag{1}$$

where $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$ and $\mathbf{h}_i$ is the embedding of node v_i in the node embedding matrix $\mathbf{H}$. θ, ω denote the parameters of $f_\theta(\cdot)$ and $g_\omega(\cdot)$, and $\ell(\cdot, \cdot)$ denotes the cross entropy loss.

2.2.2 Link-level tasks

Link-level tasks focus on the representation of paris nodes or properties of edges. Taking link prediction as an example, given two nodes, the goal is to discriminate if there is an edge between them. Thus, the objective function for link prediction can be defined as minimizing loss $\mathcal{L}_{link}$,

$$\min_{\theta,\omega} \mathcal{L}_{link}\left(\mathbf{A}, \mathbf{X}, \theta, \omega\right) = \sum_{v_i, v_j \in \mathcal{V}, i \neq j} \ell\big(g_\omega(\mathbf{h}_i, \mathbf{h}_j), \mathbf{A}_{i,j}\big) \tag{2}$$

where $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$ and $\mathbf{h}_i$ is the embedding of node v_i in the node embedding matrix $\mathbf{H}$. $g_\omega(\cdot)$ linearly maps the input to a 1-dimension value, θ, ω denote the parameters of $f_\theta(\cdot)$ and $g_\omega(\cdot)$, and $\ell(\cdot, \cdot)$ is the cross entropy loss.

2.2.3 Graph-level tasks

Graph-level tasks learn from multiple graphs in a dataset and predict the property of a single graph. For example, graph regression is a typical graph-level task where only a subset of graphs $\mathcal{G}_L$ with corresponding properties $\mathcal{P}_L$

TABLE 1
Notations used in this paper.

Notations	Descriptions
$\mathbb{R}^m$	m-dimensional Euclidean space
$a, \mathbf{a}, \mathbf{A}$	Scalar, vector, matrix
$\mathcal{G}$	The set of graphs, $\mathcal{G} = \{g_1, g_2, \cdots, g_{\lvert\mathcal{G}\rvert}\}$
g	A graph $g = (\mathcal{V}, \mathcal{E})$
$\mathcal{V}$	The set of nodes in graph g
v_i	A node $v_i \in V$
$\mathcal{E}$	The set of edges in graph g
$e_{i,j}$	An edge $e_{i,j} \in E$ between node v_i and node v_j
N	Number of nodes, $N = \lvert\mathcal{V}\rvert$
M	Number of edges, $M = \lvert\mathcal{E}\rvert$
$\mathbf{A} \in \mathbb{R}^{N \times N}$	A graph adjacency matrix
$\mathbf{A}^T$	The transpose of the matrix $\mathbf{A}$
$\mathbf{A}^n$	The n^{th} power of $\mathbf{A}$
$deg(v_i)$	Teh degree of node v_i
$\mathbf{D}$	The degree matrix of $\mathbf{A}$, $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$
$\mathbf{I}_N$	Identity matrix of dimension N
$\mathcal{N}_i^{(l)}$	l-hop Neighborhood set of node v_i
$\mathcal{N}_i$	1-hop Neighborhood set of node v_i
L	The layer number
l	The layer index, $1 \leq l \leq L$
T	The time step/iteration number
t	The time step/iteration index, $1 \leq t \leq T$
d_0	Dimension of node feature vectors
d_l	Dimension of node embeddings in the l-th layer
b_0	Dimension of edge feature vectors
$\mathbf{x}_i \in \mathbb{R}^{d_0}$	Feature vector of node v_i
$\mathbf{X} \in \mathbb{R}^{N \times d_0}$	Node feature matrix, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$
$\mathbf{X}^{(t)} \in \mathbb{R}^{N \times d_0}$	Node feature matrixat the time step t
$\mathbf{x}^e_{i,j} \in \mathbb{R}^{b_0}$	Feature vector of edge $e_{i,j}$
$\mathbf{X}^e \in \mathbb{R}^{M \times b_0}$	Edge feature matrix
$\mathbf{h}_i^{(l)} \in \mathbb{R}^{d_l}$	Node embedding of node v_i in the l-th layer
$\mathbf{H}^{(l)} \in \mathbb{R}^{N \times d_l}$	Embedding matrix of graph g in the l-th layer
$\mathbf{h}_i \in \mathbb{R}^{d_L}$	Node embedding in the L-th layer, $\mathbf{h}_i = \mathbf{h}_i^{(L)}$
$\mathbf{H} \in \mathbb{R}^{N \times d_L}$	Embedding matrix in the L-th layer, $\mathbf{H} = \mathbf{H}^{(L)}$
$\mathbf{h}_g \in \mathbb{R}^{N \times d_L}$	Graph-level representation of graph g
$\lvert\cdot\rvert$	The length of a set
$\odot$	Element-wise multiplication operation
$\lVert$	Vector concatenation
$\sigma(\cdot)$	The logistic sigmoid activation function
$\tanh(\cdot)$	The hyperbolic tangent activation function
$\mathrm{LeakyReLU}(\cdot)$	The LeakyReLU activation function
$\mathrm{READOUT}(\cdot)$	The readout function
$f_\theta, f_{\theta_1}, f_{\theta_2}, \cdots$	L-layer graph encoder to output $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$
$f_\gamma, f_{\gamma_1}, f_{\gamma_2}, \cdots$	Output graph-level representation $\mathbf{h}_g = f_\gamma(\mathbf{A}, \mathbf{X})$
$g_\omega, g_{\omega_1}, g_{\omega_2}, \cdots$	The prediction head
$\mathcal{T}, \mathcal{T}_1, \mathcal{T}_2, \cdots$	The data augmentation for graph data
$\mathbf{W}, \Theta, \theta, \gamma, \omega$	Learnable model parameters

are known, and we denote it as $\mathcal{D}_L = (\mathcal{G}_L, \mathcal{P}_L)$. Let $f_\gamma : \mathcal{G} \to \mathcal{P}$ be a graph encoder trained on labeled data $\mathcal{D}_L$ and then used to infer the properties $\mathcal{P}_U$ of unlabeled graphs $\mathcal{G}_U$. Thus, the objective function for graph regression can be defined as minimizing loss $\mathcal{L}_{graph}$:

$$\min_{\gamma,\omega} \mathcal{L}_{graph}\left(\mathbf{A}_i, \mathbf{X}_i, \gamma, \omega\right) = \sum_{(g_i,p_i)\in\mathcal{D}_L} \ell\big(g_\omega(\mathbf{h}_{g_i}), p_i\big) \quad (3)$$

where $\mathbf{h}_{g_i} = f_\gamma(\mathbf{A}, \mathbf{X})$ is the graph-level representation of graph g_i. $g_\omega(\cdot)$ linearly maps the input to a 1-dimension value, γ, ω denote the parameters of $f_\gamma(\cdot)$ and $g_\omega(\cdot)$, and $\ell(\cdot,\cdot)$ is used to measure differences between the predicted and ground-truth properties (e.g., mean absolute value).

2.3 Graph Neural Networks

Graph neural networks (GNN) [1–3] are a family of neural networks that have been widely used as the backbone encoder for feature extraction in most of the reviewed works. We introduce a general GNN framework in this subsection. A GNN involves two key computations for each node v_i at every layer: (1) AGGREGATE operation: aggregating messages from neighborhood $\mathcal{N}_i$; (2) UPDATE operation: updating node representation from its representation in the previous layer and the aggregated messages. Considering a L-layer GNN, the formulation of the l-th layer is as follows

$$\begin{aligned}\mathbf{a}_i^{(l)} &= \text{AGGREGATE}^{(l)}\left(\left\{\mathbf{h}_j^{(l-1)} : v_j \in \mathcal{N}_i\right\}\right)\\ \mathbf{h}_i^{(l)} &= \text{UPDATE}^{(k)}\left(\mathbf{h}_i^{(l-1)}, \mathbf{a}_i^{(l)}\right)\end{aligned} \quad (4)$$

where $0 \le l \le L$ and $\mathbf{h}_i^{(l)}$ is the embedding of node v_i in the l-th layer with $\mathbf{h}_i^{(0)} = \mathbf{x}_i$. For node-level or edge-level tasks, the node representation $\mathbf{h}_i^{(L)}$ can sometimes be used for downstream tasks directly. However, for graph-level tasks, an extra READOUT function is required to aggregate node features to obtain a graph-level representation $\mathbf{h}_g$, as follow

$$\mathbf{h}_g = \text{READOUT}\left(\left\{\mathbf{h}_i^{(L)} \mid v_i \in \mathcal{V}\right\}\right) \quad (5)$$

The design of these component functions is crucial, but it is beyond the scope of this paper. For a thorough review, we refer readers to the recent survey [41].

2.4 Training Strategy

Considering the relationship among bottleneck encoders, self-supervised pretext tasks, and downstream tasks, the training strategies can be divided into three categories: Pre-training and Fine-tuning (P&F), Joint Learning (JL), and Unsupervised Representation Learning (URL), with their detailed workflow shown in Fig. 2. Without loss of generality, we take node-level tasks with graph encoder $f_\theta(\cdot)$ as example; for graph-level tasks, we can use $f_\gamma(\cdot)$ instead.

2.4.1 Pre-training and Fine-tuning (P&F)

In this strategy, the model is trained in a two-stage paradigm [29]. The encoder $f_\theta(\cdot)$ is pre-trained with the pretext tasks, then the pre-trained parameters θ_{init} are used as the initialization of the encoder $f_{\theta_{init}}(\cdot)$. At the fine-tuning stage, the pre-trained encoder $f_{\theta_{init}}(\cdot)$ is fine-tuned with a prediction

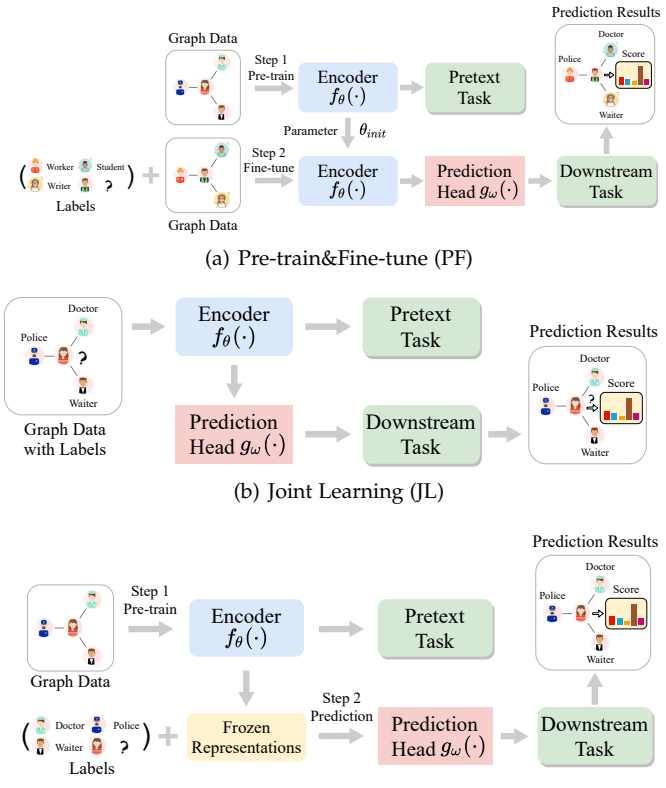


Fig. 2. An overview of training strategies for graph SSL. The training strategies can be divided into three categories: pre-training and fine-tuning, joint learning, and unsupervised representation learning. **(a)**: for the Pre-train&Fine-tune strategy, it first pre-trains the encoder $f_\theta(\cdot)$ with unlabeled nodes by the self-supervised pretext tasks. The pre-trained encoder's parameters θ_{init} are then used as the initialization of the encoder for supervised fine-tuning on downstream tasks. **(b)**: for the Joint Learning strategy, an auxiliary pretext task is included to help learn the supervised downstream task. The encoder is trained through both the pretext task and the downstream task simultaneously. **(c)**: for the Unsupervised Representation Learning strategy, it first pre-trains the encoder $f_\theta(\cdot)$ with unlabeled nodes by the self-supervised pretext tasks. The pre-trained encoder's parameters θ_{init} are then frozen and used in the supervised downstream task with additional labels.

head $g_\omega(\cdot)$ under the supervision of specific downstream tasks. The learning objective is formulated as

$$\theta^*, \omega^* = \arg\min_{(\theta,\omega)} \mathcal{L}_{task}(f_\theta, g_\omega) \quad (6)$$

with initialization

$$\theta_{init} = \arg\min_{\theta} \mathcal{L}_{ssl}(f_\theta) \quad (7)$$

where $\mathcal{L}_{task}$ and $\mathcal{L}_{ssl}$ is the loss function of downstream tasks and self-supervised pretext tasks, respectively.

2.4.2 Joint Learning

In this scheme, the encoder $f_\theta(\cdot)$ is jointly trained with a prediction head $g_\omega(\cdot)$ under the supervision of the pretext tasks and downstream tasks. The joint learning strategy can also be considered as a kind of multi-task learning or the pretext task is served as a regularization of the downstream task. The learning objective is formulated as

$$\theta^*, \omega^* = \arg\min_{(\theta,\omega)} \mathcal{L}_{task}(f_\theta, g_\omega) + \alpha \arg\min_{\theta} \mathcal{L}_{ssl}(f_\theta) \quad (8)$$

where α is a trade-off hyper-parameter that controls the contribution of $\mathcal{L}_{task}$ and $\mathcal{L}_{ssl}$.

2.4.3 Unsupervised Representation Learning

This strategy can also be considered as a two-stage paradigm, with the first stage similar to that of Pre-training. However, at the second stage, the pre-trained parameters θ_{init} are *frozen* and the model is trained on the *frozen representations* with the downstream task only. The learning objective is formulated as

$$\omega^* = \arg\min_{\omega} \mathcal{L}_{task}(f_{\theta_{init}}, g_\omega) \tag{9}$$

with initialization

$$\theta_{init} = \arg\min_{\theta} \mathcal{L}_{ssl}(f_\theta) \tag{10}$$

Compared with other schemes, unsupervised representation learning is more challenging since there is no supervision from the downstream task during pre-training stage.

3 Contrastive Learning

3.1 A Unified Perspective

Inspired by the recent advances in contrastive learning in domains such as CV and NLP, some works have been proposed to apply contrastive learning for graph data. However, most of these works simply present motivations or implementations from different perspectives, but *adopt very similar (or even the same) architectures and designs in practice*, which leads to the emergence of a large number of *duplicative efforts* and hinders the healthy development of the community. In this survey, we therefore review existing work from a unified perspective and unify them into a general framework, and present various designs for the three main modules for contrastive learning, e.g. data augmentation, pretext tasks, and contrastive objectives. In turn, the contributions of existing work can be essentially summarized as innovations in these three modules.

In practice, we usually generate multiple views for each instance in the dataset through a variety of data augmentations. Two views generated from the same instance are usually considered as a positive pair, while two views generated from different instances are considered as a negative pair. The primary goal of contrastive learning is to maximize the agreement of two jointly sampled positive pairs and minimize the agreement of two independently sampled negative pairs. The agreement between views is usually measured through Mutual Information (MI) estimation. Considering a given graph $g = (\mathbf{A}, \mathbf{X})$ distributed over $\mathcal{G}$, K different transformations $\mathcal{T}_1, \mathcal{T}_1, \cdots, \mathcal{T}_K$ can be applied to obtain multiple views $\{(\mathbf{A}_k, \mathbf{X}_k)\}_{k=1}^K$, defined as

$$\mathbf{A}_k, \mathbf{X}_k = \mathcal{T}_k(\mathbf{A}, \mathbf{X}); k = 1, 2, \cdots, K \tag{11}$$

Secondly, we can apply a set of graph encoders $\{f_{\theta_k}\}_{k=1}^K$ (may be *identical* or *share weights*) to generate different representations $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_K$ from each view, give by

$$\mathbf{h}_k = f_{\theta_k}(\mathbf{A}_i, \mathbf{X}_i); k = 1, 2, \cdots, K \tag{12}$$

The optimization goal of contrastive learning is to maximize the mutual information of two views from the same instance, formulated as

$$\max_{\theta_1, \theta_2, \cdots, \theta_K} \sum_i \sum_{j \neq i} \alpha_{i,j} \mathcal{MI}(\mathbf{h}_i, \mathbf{h}_j) \tag{13}$$

where $i, j \in \{1, 2, \cdots, K\}$, $\{\mathbf{h}_i\}_{i=1}^K$ are representations generated from $g = (\mathbf{A}, \mathbf{X})$, which are taken as positive samples. The negative samples to contrast with $\{\mathbf{h}_i\}_{i=1}^K$ are representations $\{\widetilde{\mathbf{h}}_i\}_{i=1}^K$ that are generated from another graph $\widetilde{g} = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}})$. Besides, we have $\alpha_{i,j} \in \{0, 1\}$, and their values vary in different schemes. $\mathcal{MI}(\mathbf{h}_i, \mathbf{h}_j)$ are the mutual information between two representations $\mathbf{h}_i$ and $\mathbf{h}_j$. Note that *depending on different pretext tasks*, $\{\mathbf{h}_k\}_{k=1}^K$ may not be at the same scale, either being a node-level, subgraph-level, or graph-level representation.

The design of the contrastive learning for graph data can be summarized as three main modules: (1) data augmentation strategy, (2) pretext task, and (3) contrastive objective. The design of graph encoder is not the focus of graph self-supervised learning and beyond the scope of this survey; for more details, please refer to the related survey [41].

3.2 Data Augmentation

The recent works in the CV domain show that the success of contrastive learning for visual representation learning relies heavily on well-designed data augmentation strategies, and in particular, certain kinds of augmentations play a very important role in improving performance. However, due to the inherent non-Euclidean properties of graphs, it is difficult to directly apply data augmentation strategies designed for images to the GNN domain. Here, we divide the data augmentation strategies for graph data into the following four categories: feature-based, structure-based, sampling-based, and adaptive augmentation. An overview of four types of augmentations is presented in Fig. 3.

3.2.1 Feature-based Augmentation

Given an input graph $(\mathbf{A}, \mathbf{X})$, a feature-based augmentation only performs transformation on the node feature matrix $\mathbf{X}$ or edge feature matrix $\mathbf{X}^e$. Without loss of generality, we take $\mathbf{X}$ as an example, give by

$$\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}} = \mathcal{T}(\mathbf{A}, \mathbf{X}) = \mathbf{A}, \mathcal{T}_{\mathbf{X}}(\mathbf{X}) \tag{14}$$

Attribute Masking. The attribute masking [29, 30, 42–44] randomly masks a small portion of attributes. We specify $\mathcal{T}_{\mathbf{X}}(\mathbf{X})$ for the attribute masking as

$$\mathcal{T}_{\mathbf{X}}(\mathbf{X}) = \mathbf{X} \odot (1 - \mathbf{L}) + \mathbf{M} \odot \mathbf{L} \tag{15}$$

where $\mathbf{L}$ is a masking location indicator matrix where $\mathbf{L}_{i,j} = 1$ if the j-th element of node v_i will be masked, otherwise $\mathbf{L}_{i,j} = 0$. $\mathbf{M}$ denotes a matrix with masking values. The matrix $\mathbf{L}$ is usually sampled by Bernoulli distribution or assigned manually. Besides, different schemes of $\mathbf{M}$ results in different augmentation strategies. For example, $\mathbf{M} = \mathbf{0}$ denotes a constant masking, $\mathbf{M} \sim N(\mathbf{0}, \mathbf{\Sigma})$ replaces the original values by Gaussian noise and $\mathbf{M} \sim N(\mathbf{X}, \mathbf{\Sigma})$ adds Gaussian noise to the input.

Attribute Shuffling. The attribute shuffling [28, 45–48] performs the row-wise shuffling on the attribute matrix $\mathbf{X}$. That is, the augmented graph consists of the same nodes as the original graph, but they are located in different places in the graph, and therefore receive different contextual information. We specify $\mathcal{T}_{\mathbf{X}}(\mathbf{X})$ for the attribute shuffling as

$$\mathcal{T}_{\mathbf{X}}(\mathbf{X}) = \mathbf{X}[idx, :] \tag{16}$$

where idx is a list containing numbers from 1 to $N(N = |\mathcal{V}|)$, but with a random arrangement.

3.2.2 Structue-based Augmentation

Given a graph $(\mathbf{A}, \mathbf{X})$, a structue-based augmentation only performs transformation on the adjacent matrix $\mathbf{A}$, as follow

$$\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}} = \mathcal{T}(\mathbf{A}, \mathbf{X}) = \mathcal{T}_{\mathbf{A}}(\mathbf{A}), \mathbf{X} \tag{17}$$

Edge Perturbation. The edge perturbation [33, 42, 49–51] perturbs structural connectivity through randomly adding or dropping a certain ratio of edges. We specify $\mathcal{T}_{\mathbf{A}}(\mathbf{A})$ for the edge perturbation as

$$\mathcal{T}_{\mathbf{A}}(\mathbf{A}) = \mathbf{A} \odot (1 - \mathbf{L}) + (1 - \mathbf{A}) \odot \mathbf{L} \tag{18}$$

where $\mathbf{L}$ is a perturbation location indicator matrix where $\mathbf{L}_{i,j} = \mathbf{L}_{j,i} = 1$ if the connection between node v_i and v_j will be disturbed, otherwise $\mathbf{L}_{i,j} = \mathbf{L}_{j,i} = 0$. For example, if $\mathbf{A}_{i,j} = 1$ and $\mathbf{L}_{i,j} = \mathbf{L}_{j,i} = 1$, the edge between node v_i and v_j will be removed; $\mathbf{A}_{i,j} = 0$ and $\mathbf{L}_{i,j} = \mathbf{L}_{j,i} = 1$, the edge between node v_i and v_j will be added. Different values in $\mathbf{L}$ result in different perturbation strategies, and more values set to 1 in $\mathbf{L}$, more server the perturbation is.

Node Insertion. The node insertion [51] adds K nodes $\mathcal{V}_a = \{v_{N+k}\}_{k=1}^{K}$ to node set $\mathcal{V}$ and add some edges between $\mathcal{V}_a$ and $\mathcal{V}$. For a structure transformation $\widetilde{\mathbf{A}} = \mathcal{T}_{\mathbf{A}}(\mathbf{A})$, we have $\widetilde{\mathbf{A}}_{:N,:N} = \mathbf{A}$. Given the connection ratio r, we have

$$p(\widetilde{\mathbf{A}}_{i,j} = \widetilde{\mathbf{A}}_{j,i} = 1) = r, p(\widetilde{\mathbf{A}}_{i,j} = \widetilde{\mathbf{A}}_{j,i} = 0) = 1 - r \tag{19}$$

for $N + 1 \leq i, j \leq N + K$.

Edge Diffusion. The edge diffusion [37, 52] creates new connections between nodes. One of classic diffusion transformations is the Personalized PageRank, formulated as

$$\mathcal{T}_{\mathbf{A}}(\mathbf{A}) = \alpha \left(\mathbf{I}_n - (1 - \alpha)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^{-1} \tag{20}$$

where α denotes the teleport probability in a random walk.

3.2.3 Sampling-based Augmentation

Given an input graph $(\mathbf{A}, \mathbf{X})$, a sampling-based augmentation performs transformation on both the adjacent matrix $\mathbf{A}$ and feature matrix $\mathbf{X}$, as follows

$$\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}} = \mathcal{T}(\mathbf{A}, \mathbf{X}) = \mathbf{A}[\mathcal{S}, \mathcal{S}], \mathbf{X}[\mathcal{S}, :] \tag{21}$$

where $\mathcal{S} \in \mathcal{V}$ and existing methods usually apply five strategies to obtain the node subset $\mathcal{S}$: uniform sampling, ego-nets sampling, random walk sampling, importance sampling, and knowledge-based sampling.

Uniform Sampling (Node Dropping). The uniform sampling [51] (also known as Node Dropping) uniformly samples a given number of nodes $\mathcal{S}$ from $\mathcal{V}$ and remove the remaining nodes $\mathcal{D} = \mathcal{V}/\mathcal{S}$ directly.

Ego-nets Sampling [30, 53, 54]. Given a typical graph encoder with L layers, the computation of the node representation only depends on its L-hop neighborhood. In particular, for each node v_i in a graph, the transformation $\mathcal{T}(\cdot)$ samples the L-ego net surrounding node v_i, with $\mathcal{S}$ defined as

$$\mathcal{S} = \{v_j \mid d(v_i, v_j) \leq L\} \tag{22}$$

where $d(v_i, v_j)$ is the shortest path length between node v_i and v_j. The Ego-nets Sampling is essentially a special version of Breadth-First Search (BFS) sampling.

Random Walk Sampling [37, 42, 55]. It starts a random walk on graph g from the ego node v_i. The walk iteratively travels to its neighborhood with the probability proportional to the edge weight. In addition, at each step, with a positive probability α the walk returns back to the starting node v_i. Finally, the visited nodes are collected into a node subset $\mathcal{S}$.

Importance Sampling [38]. For a given node v_i, we can sample a subgraph based on the importance of its neighboring nodes. Specifically, we can calculate an importance score matrix $\mathbf{M}$ denoted as

$$\mathbf{M} = \alpha \cdot (\mathbf{I}_n - (1 - \alpha) \cdot \mathbf{A}\mathbf{D}^{-1}) \tag{23}$$

where $\alpha \in [0, 1]$ is a hyper-parameter. For a given node v_i, the subgraph sampler chooses top-k important neighbors to constitute a subgraph with the index of chosen nodes denoted as $\mathcal{S} = \text{top\_ rank}(\mathbf{M}(i, :), k)$.

Knowledge Sampling [56]. The knowledge-based sampling incorporates domain knowledge into subgraph sampling. For example, the subgraph sampling process can be formalized as a *library-based matching* problem by counting the frequently occurring and bioinformatics substructures in the molecular graph and building libraries (or tables) for them.

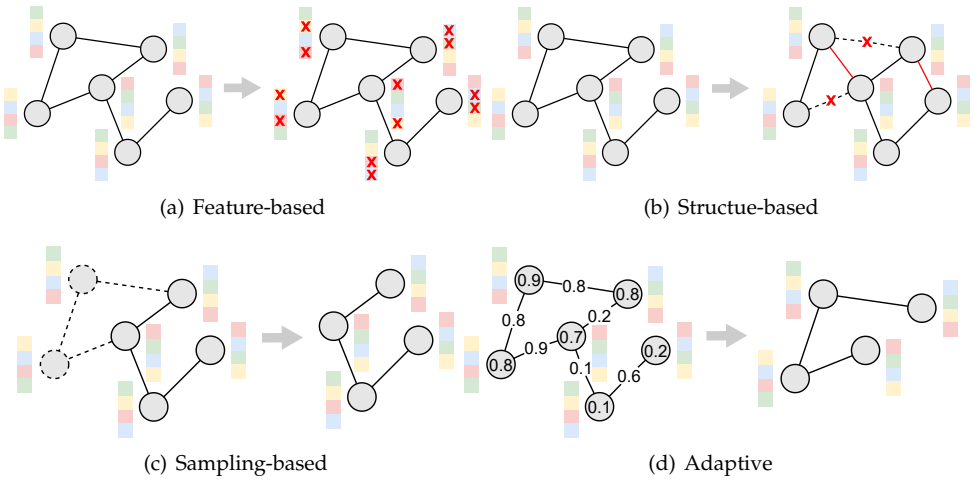
3.2.4 Adaptive Augmentation

The adaptive augmentation usually employs attention scores or gradients to guide the selection of nodes or edges.

Attention-based. The attention-based methods typically define importance scores for nodes or edges and then augment data based on their importance. For example, GCA [57] proposes to keeps important structures and attributes unchanged, while perturbing possibly unimportant edges and features. Specifically, the probability of edge removal and feature masking should be closely related to their importance. Given a node centrality measure $\varphi_c(\cdot) : \mathcal{V} \to \mathbb{R}^+$, it defines edge centrality as the average of two adjacent nodes' centrality scores, i.e., $s_{i,j} = \log \frac{\varphi_c(v_i) + \varphi_c(v_j)}{2}$. Then, the importance of the edge $e_{i,j}$ is defined as

$$p_{i,j} = \min \left(\frac{s_{\max} - s_{i,j}}{s_{\max} - \mu_s} \cdot p_e, p_\tau \right) \tag{24}$$

where p_e is a hyperparameter that controls the overall probability of removing edges, $s_{\max}$ and μ_s is the maximum and average of $\{s_{i,j}\}_{j=1}^{N}$ and $p_\tau < 1$ is a



(a) Feature-based (b) Structue-based

(c) Sampling-based (d) Adaptive

Fig. 3. A comparison of the feature-based, structue-based, sampling-based, and adaptive augmentation. The feature-based augmentation generally randomly (or manually) masks a small portion of node or edge attributes with constants or random values. The structue-based augmentation randomly (or manually) adds or removes small portions of edges from the graph, which includes methods like edge perturbation, node insertion, and edge diffusion. The sampling-based augmentation samples nodes and their connected edges from the graph under specific rules, which include Uniform Sampling, Ego-net Sampling, Random Walk Sampling, Importance Sampling, Knowledge Sampling, etc. The adaptive sampling adopts attention or gradient-based schemes to perform adaptive sampling based on the learned attention score or gradient magnitude. The numbers in the Fig. 3(d) are the importance scores of the nodes and edges, and we sample the most important 4 nodes and 3 edges as an example.

cut-off probability, used to truncate the probabilities since extremely high removal probabilities will lead to overly corrupted graph structures. The node centrality can be defined as degree centrality, Eigenvector centrality [58], or PageRank centrality [59], which results in three different variants. Note that performing attribute masking based on node importance is the same as above and will not be repeated here.

Gradient-based. Unlike the handcrafted or simple uniform edge removal and insertion as in GRACE, GROC [60] adaptively performs gradient-based augmentation guided by edge gradient information. Specifically, it first applies two stochastic transformations $\mathcal{T}_1(\cdot)$ and $\mathcal{T}_2(\cdot)$ to $(\mathbf{A}, \mathbf{X})$ to obtain two views, masking node attributes independently with probability r_1 and r_2 and then computing the contrastive loss $\mathcal{L}_{self}$ between these two views. For a given node v_i, an edge removal candidate set is defined as

$$\mathcal{S}^- = \left\{(v_i, v_k) \middle| v_k \in \mathcal{N}_i^{(l)}\right\} \tag{25}$$

, and an edge insertation candidate set is defined as

$$\mathcal{S}^+ = \left\{(v_i, v_k) \middle| v_k \in \left(\cup_{v_m \in \mathcal{B}} \mathcal{N}_m^{(l)} \backslash \mathcal{N}_i^{(l)}\right)\right\} \tag{26}$$

where $\mathcal{B}$ is a node batch, and the $\mathcal{S}^+$ is restricted to the set of edges (v_i, v_k) where v_i is an anchor node, and v_k is within the l-hop neighborhood of some other anchors $v_m \neq v_i$ but not within the l-hop neighborhood of node v_i. Finally, we backpropagate the loss $\mathcal{L}_{self}$ to obtain gradient intensity values for each edge in $\mathcal{S}^-$ and $\mathcal{S}^+$. A further gradient-adaptive augmentations are applied on the views by the adversarial transformations, which remove a subset of edges with *minimal* edge gradient values in $\mathcal{S}^-$ and insert a subset of edges with the *maximal* edge gradient values in $\mathcal{S}^+$.

3.3 Pretext Task

The contrastive learning aims to maximize the agreement of two jointly sampled positive pairs. Depending on the definition of a graph view, the scale of the view may be local, contextual, or global, corresponding to the node-level, subgraph-level, or graph-level information in the graph. Therefore, contrastive learning may contrast two graph views *at the same or different scales*, which leads to two categories: (1) Contrasting with the same-scale and (2) Contrasting with the cross-scale. The two views in the same-scale contrasting, either positive or negative pairs, are at the same scale, such as node-node pairs and graph-graph pairs, while the two views in the cross-scale contrasting have different scales, such as node-subgraph or node-graph contrasting. In this section, we categorize the existing methods in detail from these two perspectives and present them in a unified mathematical framework. A general framework for contrastive methods is presented in Fig. 4.

3.3.1 *Contrasting with the same-scale*

The same-scale contrastive learning is further refined into 3 categories based on the different scales of the views: local-local, context-context, and global-global contrasting.

3.3.1.1 Global-Global Contrasting

GraphCL [42]. Four types of graph augmentations $\{\mathcal{T}_k\}_{k=1}^4$ are applied to incorporate various priors: (1) Node Dropping $\mathcal{T}_1(\cdot)$; (2) Edge Perturbation $\mathcal{T}_2(\cdot)$; (3) Attribute Masking $\mathcal{T}_3(\cdot)$; (4) Subgraph Sampling $\mathcal{T}_4(\cdot)$. Given a graph $g_i = (\mathbf{A}_i, \mathbf{X}_i) \in \mathcal{G}$, it first applies a series of graph augmentations $\mathcal{T}(\cdot)$ randomly selected from $\{\mathcal{T}_k\}_{k=1}^4$ to generate an augmented graph $\widetilde{g}_i = (\widetilde{\mathbf{A}}_i, \widetilde{\mathbf{X}}_i) = \mathcal{T}(\mathbf{A}_i, \mathbf{X}_i)$, and then learns to predict whether two graphs originate from the same graph or not. Specifically, a shared graph encoder $f_\theta(\cdot)$ and a READOUT function are applied to obtain

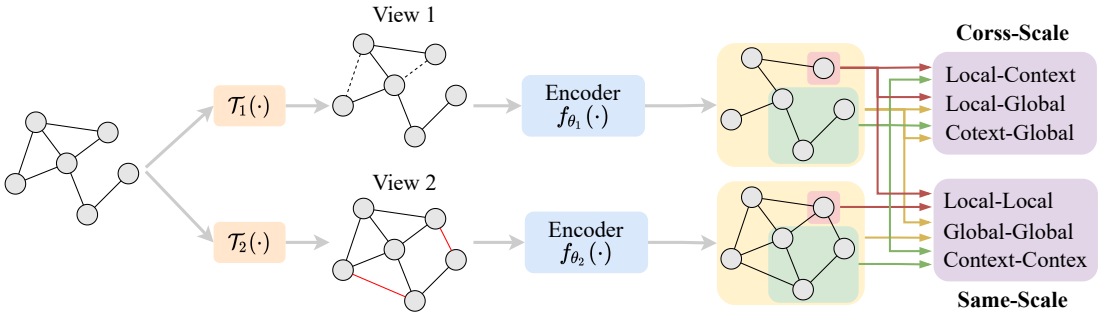


Fig. 4. A general framework for contrastive learning methods. The design of the contrastive methods for graphs can be determined by three main modules: data augmentation strategies, pretext tasks, and contrastive objectives. Different views can be generated by a single or a combination of augmentation $\mathcal{T}_1(\cdot)$ and $\mathcal{T}_2(\cdot)$. For graph encoder $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$, the commonly used graph neural networks include GAE [61], VGAE [61], MGAE [62] and GALA [63], etc. However, the design of graph encoder is not the focus of graph SSL and thus beyond the scope of this survey. The two contrasting components may be local, contextual, or global, corresponding to node-level (marked in red), subgraph-level (marked in green), or graph-level (marked in yellow) information in the graph. The contrastive learning can thus contrast two views (*at the same or different scales*, which leads to two categories of algorithm: (1) same-scale contrasting, including local-local, context-context, and global-global contrasting; and (2) cross-scale contrasting, including local-context, local-global, and context-global contrasting.

graph-level representations $\mathbf{h}_{g_i} = \text{READOUT}(f_\theta(\mathbf{A}_i, \mathbf{X}_i))$ and $\widetilde{\mathbf{h}}_{\widetilde{g}_i} = \text{READOUT}(f_\theta(\widetilde{\mathbf{A}}_i, \widetilde{\mathbf{X}}_i))$. Finally, the learning objective is defined as follows

$$\max_{\theta} \frac{1}{|\mathcal{G}|} \sum_{g_i \in \mathcal{G}} \mathcal{MI}\left(\mathbf{h}_{g_i}, \widetilde{\mathbf{h}}_{\widetilde{g}_i}\right) \tag{27}$$

where the negative samples to contrast with $\mathbf{h}_{g_i}$ is $Neg(\mathbf{h}_{g_i}) = \{\widetilde{\mathbf{h}}_{\widetilde{g}_j}\}_{g_j \in \mathcal{G}, j \neq i}$. **Contrastive Self-supervised Learning (CSSL)** [51] follows a very similar (almost the same) framework to GraphGL, differing only in the way the data is augmented. Along with node dropping, it also considers node insertion as an important augmentation strategy. Specifically, it randomly selects a strongly-connected subgraph S, remove all edges in S, add a new node v_i, and add an edge between v_i and each node in S.

Iterative Graph Self-Distillation (IGSD) [50] is a graph self-supervised learning paradigm that iteratively performs the teacher-student distillation with graph augmentations. Specifically, given a graph $g_i = (\mathbf{A}_i, \mathbf{X}_i) \in \mathcal{G}$, it performs a series of augmentation transformations $\mathcal{T}(\cdot)$ to generate an augmented graph $\widetilde{g}_i = (\widetilde{\mathbf{A}}_i, \widetilde{\mathbf{X}}_i) = \mathcal{T}(\mathbf{A}_i, \mathbf{X}_i)$. Then two graph encoder $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$ as well as a READOUT function are applied to obtain the graph-level representations $\mathbf{h}_{g_i} = \text{READOUT}\left(f_{\theta_1}(\mathbf{A}_i, \mathbf{X}_i)\right)$ and $\widetilde{\mathbf{h}}_{\widetilde{g}_i} = \text{READOUT}(f_{\theta_2}(\widetilde{\mathbf{A}}_i, \widetilde{\mathbf{X}}_i))$, respectively. Moreover, a prediction head $g_\omega(\cdot)$ is used in the student network to obtain $g_\omega(\mathbf{h}_{g_i})$ and $g_\omega(\widetilde{\mathbf{h}}_{\widetilde{g}_i})$. To contrast latents $\mathbf{h}_{g_i}$ and $\widetilde{\mathbf{h}}_{\widetilde{g}_i}$, the symmetric consistency loss is defined as

$$\mathcal{D}\left(\mathbf{h}_{g_i}, \widetilde{\mathbf{h}}_{\widetilde{g}_i}\right) = \left\| g_\omega(\mathbf{h}_{g_i}) - \widetilde{\mathbf{h}}_{\widetilde{g}_i} \right\|^2 + \left\| g_\omega(\widetilde{\mathbf{h}}_{\widetilde{g}_i}) - \mathbf{h}_{g_i} \right\|^2 \tag{28}$$

Finally, the learning objective is defined as follows

$$\max_{\theta_1, \omega} \frac{1}{|\mathcal{G}|} \sum_{g_i \in \mathcal{G}} \log \frac{\exp\left(\mathcal{D}(\mathbf{h}_{g_i}, \widetilde{\mathbf{h}}_{\widetilde{g}_i})\right)}{\sum_{j=1}^{N} \exp\left(\mathcal{D}(\mathbf{h}_{g_i}, \widetilde{\mathbf{h}}_{\widetilde{g}_j})\right)} \tag{29}$$

where the negative samples to contrast with $\mathbf{h}_{g_i}$ is $Neg(\mathbf{h}_{g_i}) = \{\widetilde{\mathbf{h}}_{\widetilde{g}_j}\}_{g_j \in \mathcal{G}, j \neq i}$. The parameter θ_2 of teacher network $f_{\theta_2}(\cdot)$ are updated as an exponential moving average (EMA) of the student network parameters θ_1, as follows

$$\theta_2 \longleftarrow \alpha\theta_2 + (1-\alpha)\theta_1 \tag{30}$$

where $\alpha \in [0, 1)$ is the momentum weight to control the speed of θ_2 evolving.

Domain-Agnostic Contrastive Learning (DACL) [64] uses the *Mixup* noise to create positive and negative examples by mixing data samples differently either at the input or hidden spaces. Specifically, given a graph $g_i = (\mathbf{A}_i, \mathbf{X}_i) \in \mathcal{G}$, it applies a graph encoder $f_\theta(\cdot)$ and a READOUT function to obtain the graph-level representation $\mathbf{h}_{g_i} = \text{READOUT}(f_\theta(\mathbf{A}_i, \mathbf{X}_i))$. Then it performs the mixup transformations to generate two positive pairs , e.g., $\mathbf{h}_{g_i}^{(1)} = \lambda_1 \mathbf{h}_{g_i} + (1-\lambda_1)\mathbf{h}_{g_m}$ and $\mathbf{h}_{g_i}^{(2)} = \lambda_2 \mathbf{h}_{g_i} + (1-\lambda_2)\mathbf{h}_{g_n}$, where λ_1 and λ_2 are sampled from a Gaussian distribution and $\mathbf{h}_{g_m}$ and $\mathbf{h}_{g_n}$ are randomly sampled from $\{\mathbf{h}_{g_k}\}_{k=1,k\neq i}^{N}$, respectively. Finally, the learning objective is defined as

$$\max_{\theta, \omega} \frac{1}{|\mathcal{G}|} \sum_{g_i \in \mathcal{G}} \mathcal{MI}\left(g_\omega(\mathbf{h}_{g_i}^{(1)}), g_\omega(\mathbf{h}_{g_i}^{(2)})\right) \tag{31}$$

where $g_\omega(\cdot)$ is a nonlinear prediction head, and the negative samples to contrast with $g_\omega(\mathbf{h}_{g_i}^{(1)})$ is $Neg\left(g_\omega(\mathbf{h}_{g_i}^{(1)})\right) = \{g_\omega(\mathbf{h}_{g_j}^{(c)})\}_{j \neq i; c=1,2}$.

Label Contrastive Coding (LCC) [65] is proposed to encourage intra-class compactness and inter-class separability. To power contrastive learning, LCC introduces a dynamic label memory bank and a momentum updated encoder. Specifically, the query graph (g_q, y_q) and key graph (g_k, y_k) are encoded by two graph encoder $f_{\gamma_q}(\cdot)$ and $f_{\gamma_k}(\cdot)$ to obtain graph-level representations $\mathbf{h}_{g_q}$ and $\mathbf{h}_{g_k}$ respectively. If $\mathbf{h}_{g_q}$ and $\mathbf{h}_{g_k}$ have the same label, they are considered as the positive pair, otherwise, they are the negative pair. The label contrastive loss encourages the model to distinguish

the positive pair from the negative pair. For the encoded query (g_q, y_q), its label contrastive loss is calculated by

$$\max_{\gamma_q} \log \frac{\sum_{i=1}^{m} \mathbf{1}_{y_i=y_q} \cdot \exp\left(\mathbf{h}_{g_q} \cdot \mathbf{h}_{g_k}^{(i)}/\tau\right)}{\sum_{i=1}^{m} \exp\left(\mathbf{h}_{g_q} \cdot \mathbf{h}_{g_k}^{(i)}/\tau\right)} \tag{32}$$

where m is the size of memory bank, τ is the temperature hyper-parameter, and $\mathbf{1}_{y_i=y_q}$ is an indicator function to determine whether the label of i-th key graph $g_k^{(i)}$ in the memory bank is the same as the query graph g_q. The parameter γ_k of $f_{\gamma_k}(\cdot)$ follows a momentum-based update mechanism as Moco [12], given by

$$\gamma_k \longleftarrow \alpha\gamma_k + (1-\alpha)\gamma_q \tag{33}$$

where $\alpha \in [0, 1)$ is the momentum weight to control the speed of γ_k evolving.

3.3.1.2 Context-Context Contrasting

Graph Contrastive Coding (GCC) [55] is a graph self-supervised pre-training framework, that captures the universal graph topological properties across multiple graphs. Specifically, it first samples multiple subgraphs for each graph $g \in \mathcal{G}$ by random walk with restart and collect them in to a memory bank $\mathcal{S}$. Then the query graph $g_q \in \mathcal{S}$ and key graphs $g_k \in \mathcal{S}$ are encoded by two graph encoders $f_{\gamma_q}(\cdot)$ and $f_{\gamma_k}(\cdot)$ to obtain low-dimensional graph-level representations $\mathbf{h}_{g_q}$ and $\mathbf{h}_{g_k}$, respectively. If $\mathbf{h}_{g_q}$ and $\mathbf{h}_{g_k}$ are sampled from the same graph, they are considerd as the positive pair, otherwise they are the negative pair. For the encoded query (g_q, y_q) where y_q is the index of graph it sampled from, its graph contrastive loss is calculated by

$$\max_{\gamma_q} \log \frac{\sum_{i=1}^{|\mathcal{S}|} \mathbf{1}_{y_i=y_q} \cdot \exp\left(\mathbf{h}_{g_q} \cdot \mathbf{h}_{g_k}^{(i)}/\tau\right)}{\sum_{i=1}^{|\mathcal{S}|} \exp\left(\mathbf{h}_{g_q} \cdot \mathbf{h}_{g_k}^{(i)}/\tau\right)} \tag{34}$$

where $\mathbf{1}_{y_i=y_q}$ is an indicator function to determine whether the i-th key graph $g_k^{(i)}$ in the memory bank and query graph g_q are *sampled from the same graph*. The parameter γ_k of $f_{\gamma_k}(\cdot)$ follows a momentum-based updating as in Equ. 30.

3.3.1.3 Local-Local Contrasting

GRACE [43]. Rather than contrasting global-global views as GraphCL and CSSL, GRACE focuses on contrasting views at the node level. Specifically, given a graph $g = (\mathbf{A}, \mathbf{X})$, it first generates two augmentatd graph views $g^{(1)} = (\mathbf{A}^{(1)}, \mathbf{X}^{(1)}) = \mathcal{T}_1(\mathbf{A}, \mathbf{X})$ and $g^{(2)} = (\mathbf{A}^{(1)}, \mathbf{X}^{(2)}) = \mathcal{T}_2(\mathbf{A}, \mathbf{X})$. Then it applies a shared graph encoder $f_\theta(\cdot)$ to generate their node embedding matrix $\mathbf{H}^{(1)} = f_\theta(\mathbf{A}^{(1)}, \mathbf{X}^{(1)})$ and $\mathbf{H}^{(2)} = f_\theta(\mathbf{A}^{(2)}, \mathbf{X}^{(2)})$. Finally, the pairwise objective for each positive pair $(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)})$ is defined as follows

$$\mathcal{L}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)}) = \log \frac{e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)})/\tau}}{e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)})/\tau} + Neg} \tag{35}$$

where Neg is defined as

$$Neg = \sum_{k=1}^{N} \mathbf{1}_{k\neq i} \left[e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(1)})/\tau} + e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(2)})/\tau}\right] \tag{36}$$

where $e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(1)})/\tau}$ is the *intra-view* negative pair and $e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(2)})/\tau}$ is the *inter-view* negative pair. The overall objective to be maximized is then defined as,

$$\max_{\theta} \frac{1}{2N} \sum_{i=1}^{N} \left[\mathcal{L}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)}) + \mathcal{L}(\mathbf{h}_i^{(2)}, \mathbf{h}_i^{(1)})\right] \tag{37}$$

GCA [57] and GROC [60] adopt the same framework and objective with GRACE but with more flexible and *adaptive* data augmentation strategies.

STDGI [45] extents the idea of GRACE [43] to spatial-temporal graphs. Specifically, given two graphs $g_t = (\mathbf{A}, \mathbf{X}^{(t)})$ and $g_{t+k} = (\mathbf{A}, \mathbf{X}^{(t+k)})$ at time t and $t+k$, a shared graph encoder $f_\theta(\cdot)$ is applied to obtain the node embedding matrix $\mathbf{H}^{(t)} = f_\theta(\mathbf{A}, \mathbf{X}^{(t)})$. Besides, it generates an augmentatd graph $\widetilde{g}_{t+k} = (\mathbf{A}, \widetilde{\mathbf{X}}^{(t+k)}) = \mathcal{T}(\mathbf{A}, \mathbf{X}^{(t+k)})$ by randomly permuting the node features. Finally, the learning objective is defined as follows

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{MI}\left(\mathbf{h}_i^{(t)}, \mathbf{x}_i^{(t+k)}\right) \tag{38}$$

where the negative samples to contrast with $\mathbf{h}_i^{(t)}$ is $Neg(\mathbf{h}_i^{(t)}) = \widetilde{\mathbf{x}}_i^{(t+k)}$.

Cross-layer Contrasting (GMI) [35]. Given a graph $g = (\mathbf{A}, \mathbf{X})$, a graph encoder $f_\theta(\cdot)$ is applied to obtain the node embedding matrix $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$. Then the Cross-layer Node Contrasting can be defined as

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{MI}\left(\mathbf{h}_i, \mathbf{x}_i\right) \tag{39}$$

where the negative samples to contrast with $\mathbf{h}_i$ is $Neg(\mathbf{h}_i) = \{\mathbf{x}_j \mid v_j \in \mathcal{N}_i\}$. Similarly, the Cross-layer Edge Contrasting can be defined as

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{v_j \in \mathcal{N}_i} \mathcal{MI}\left(\mathbf{w}_{i,j}, \mathbf{a}_{i,j}\right) \tag{40}$$

where $\mathbf{w}_{i,j} = \sigma(\mathbf{h}_i \mathbf{h}_j^T)$, and the negative samples to contrast with $\mathbf{w}_{i,j}$ are $Neg(\mathbf{w}_{i,j}) = \{\mathbf{a}_{i,k} \mid v_k \in \mathcal{N}_i \quad \text{and} \quad k \neq j\}$, where $\mathbf{a}_{i,k}$ is the edge weight between node v_i and node v_k.

KS2L [66] is a novel self-supervised knowledge distillation framework, with two complementary intra- and cross-model knowledge distillation modules. Given a graph $g = (\mathbf{A}, \mathbf{X})$, it first applies two linear mapping functions to obtain $\mathbf{Z}^{(1)} = g_{\omega_1}(\mathbf{X})$ and $\mathbf{Z}^{(2)} = g_{\omega_2}(\mathbf{X})$ and then uses two graph encoder $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$ to obtain node embedding matrices $\mathbf{H}^{(1)} = f_{\theta_1}(\mathbf{A}, \mathbf{Z}^{(1)})$ and $\mathbf{H}^{(2)} = f_{\theta_2}(\mathbf{A}, \mathbf{Z}^{(2)})$. Finally, the learning objective is defined as follows

$$\max_{\theta_1, \theta_2, \omega_1, \omega_2} \sum_{e_{i,j} \in \mathcal{E}} \mathcal{MI}\left[\left(\mathbf{h}_i^{(1)}, \mathbf{z}_j^{(1)}\right) + \left(\mathbf{h}_i^{(1)}, \mathbf{z}_j^{(2)}\right)\right] \tag{41}$$

where the first term is used for *intra-model* knowledge distillation, and the negative samples to contrast with $\mathbf{h}_i^{(1)}$ is $Neg(\mathbf{h}_i^{(1)}) = \{\mathbf{z}_k^{(1)}\}_{e_{i,k} \notin \mathcal{E}}$. The second term is used for *cross-model* knowledge distillation, and the negative samples to contrast with $\mathbf{h}_i^{(1)}$ is $Neg(\mathbf{h}_i^{(1)}) = \{\mathbf{z}_k^{(2)}\}_{e_{i,k} \notin \mathcal{E}}$.

CG^3 [67] adopts a similar framework to GRACE [43], but also draws on the successful practice of LCL[65] to perfrom node-level label contasting. Specifically, given a graph $g = (\mathbf{A}, \mathbf{X})$, it first applies a localized graph convolution encoder $f_{\theta_1}(\cdot)$ and a hierarchical graph convolution encoder $f_{\theta_2}(\cdot)$ to generate their node embedding matrix $\mathbf{H}^{(1)} = f_{\theta_1}(\mathbf{A}, \mathbf{X})$ and $\mathbf{H}^{(2)} = f_{\theta_2}(\mathbf{A}, \mathbf{X})$. To incorporate the scarce yet valuable label information for training, it proposes to use a supervised label contrastive loss as follows:

$$\max_{\theta} \frac{1}{2N} \sum_{i=1}^{N} \left[\mathcal{L}_{lc}^{(1)} + \mathcal{L}_{lc}^{(2)} \right] \tag{42}$$

where $\mathcal{L}_{lc}^{(1)}$ and $\mathcal{L}_{lc}^{(2)}$ are defined as

$$\mathcal{L}_{lc}^{(1)} = \log \frac{\sum_{k=1}^{m} \mathbf{1}_{y_k=y_i} \cdot \exp\left(\mathbf{h}_i^{(1)} \cdot \mathbf{h}_k^{(2)}\right)}{\sum_{j=1}^{m} \exp\left(\mathbf{h}_i^{(1)} \cdot \mathbf{h}_j^{(2)}\right)} \tag{43}$$

$$\mathcal{L}_{lc}^{(2)} = \log \frac{\sum_{k=1}^{m} \mathbf{1}_{y_k=y_i} \cdot \exp\left(\mathbf{h}_i^{(2)} \cdot \mathbf{h}_k^{(1)}\right)}{\sum_{j=1}^{m} \exp\left(\mathbf{h}_i^{(2)} \cdot \mathbf{h}_j^{(1)}\right)} \tag{44}$$

where m is the number of labeled nodes, and $\mathbf{1}_{y_k=y_i}$ is an indicator function to determine whether the label of node v_k is the same as node v_i.

BGRL [44]. Inspired by BYOL, BGRL proposes to perform the self-supervised learning that *does not require negative samples*, thus getting rid of the potentially quadratic bottleneck. Specifically, given a graph $g = (\mathbf{A}, \mathbf{X})$, it first generates two augmentatd graph views $g^{(1)} = (\mathbf{A}^{(1)}, \mathbf{X}^{(1)}) = \mathcal{T}_1(\mathbf{A}, \mathbf{X})$ and $g^{(2)} = (\mathbf{A}^{(1)}, \mathbf{X}^{(2)}) = \mathcal{T}_2(\mathbf{A}, \mathbf{X})$. Then it applies two graph encoders $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$ to generate their node embedding matrix $\mathbf{H}^{(1)} = f_{\theta_1}(\mathbf{A}^{(1)}, \mathbf{X}^{(1)})$ and $\mathbf{H}^{(2)} = f_{\theta_2}(\mathbf{A}^{(2)}, \mathbf{X}^{(2)})$. Moreover, a node-level prediction head $g_\omega(\cdot)$ is used to output $\mathbf{Z}^{(1)} = g_\omega(\mathbf{H}^{(1)})$. Finally, the learning objective is defined as follows

$$\max_{\theta_1} \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbf{z}_i^{(1)} (\mathbf{h}_i^{(2)})^T}{\|\mathbf{z}_i^{(1)}\| \|\mathbf{h}_i^{(2)}\|} \tag{45}$$

where the parameter θ_2 are then updated as an exponential moving average (EMA) of parameters θ_1, as done in Equ. 30.

SelfGNN [52] differs from BGRL only in the definition of the objective function. Unlike Equ. 45, SelfGNN defines the implicit contrastive term directly in the form of MSE,

$$\max_{\theta_1} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{z}_i^{(1)} - \mathbf{h}_i^{(2)}\|^2 \tag{46}$$

PT-DGNN [68] is proposed to perfrom pre-training on dynamic graph $g = (\mathbf{A}^{(t)}, \mathbf{X})$, where $\mathbf{A}_{i,j}^{(t)} = 1$ denotes the interaction between node v_i and v_j at time t, that is, t is the timestamp of edge $e_{i,j}$. It first applies the dynamic subgraph sampling (DySS) to obtain a subgraph $g_S = (\mathbf{A}_s, \mathbf{X}_s)$ for pre-training. DySS mainly includes three steps: 1) Randomly select s nodes as the sampling initial points; 2) Put the first-order neighbors of these nodes into the candidate pool and save their timestamp as weight to calculate sampling probability; 3) Select final s nodes according to sampling probability. Furthermore, it genetates an augmentated graph g_a by performing the attribute masking and edge perturbation to obtain $g_a = (\mathbf{A}_a, \mathbf{X}_a) = \mathcal{T}(\mathbf{A}_s, \mathbf{X}_s)$, where the masked node set is denoted as $\mathcal{S}$ and the masked edge set is denoted as $\mathcal{M}$. Then a shared graph encoder $f_\theta(\cdot)$ is applied to obtain node embedding matrix $\mathbf{H} = f_\theta(\mathbf{A}_a, \mathbf{X}_a)$. Finally, the objective of dynamic edge generation is defined as

$$\max_{\theta} \sum_{v_i \in \mathcal{V}_a} \sum_{e_{i,j} \in \mathcal{M}} \log \frac{\exp\left(\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j)\right)}{\sum_{v_k \in \mathcal{V}_a} \exp\left(\mathcal{D}(\mathbf{h}_i, \mathbf{h}_k)\right)} \tag{47}$$

where $\mathcal{V}_a$ is the node set of graph g_a. The learning objective of the node attribute generation is defined as follows

$$\min_{\theta, \omega} \sum_{v_i \in \mathcal{S}} \|g_\omega(\mathbf{h}_i) - \mathbf{x}_i\|^2 \tag{48}$$

where $g_\omega(\cdot)$ is a node-level nonlinear mapping function.

3.3.2 Contrasting with the cross-scale

Based on the different scales of two views, we further refined the scope of cross-scale contrastive into three categories: local-global, local-context, and context-global contrasting.

3.3.2.1 Local-Global Contrasting

Deep Graph Infomax (DGI) [28] is proposed to contrast the patch representations and corresponding high-level summary of graphs. First, it applies an augmentation transformation $\mathcal{T}(\cdot)$ to obtain a negative example $\widetilde{g} = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}) = \mathcal{T}(\mathbf{A}, \mathbf{X})$. Then it passes these two graph through two graph encoder $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$ to obtain node embedding matrices $\widetilde{\mathbf{H}} = f_{\theta_1}(\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}})$ and $\mathbf{H} = f_{\theta_2}(\mathbf{A}, \mathbf{X})$. Beside, a $\mathrm{READOUT}()$ function is used to obtain the graph-level representaion $\widetilde{\mathbf{h}}_{\widetilde{g}} = \mathrm{READOUT}(\widetilde{\mathbf{H}})$. Finally, the learning objective is defined as follows

$$\max_{\theta_1, \theta_2} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{MI}\left(\widetilde{\mathbf{h}}_{\widetilde{g}}, \mathbf{h}_i\right) \tag{49}$$

where $\mathbf{h}_i$ is the node embedding of node v_i, and the negative samples to contrast with $\widetilde{\mathbf{h}}_{\widetilde{g}}$ is $Neg(\widetilde{\mathbf{h}}_{\widetilde{g}}) = \{\mathbf{h}_j\}_{v_j \in \mathcal{V}, j \neq i}$.

High-order Deep Multiplex Infomax (HDMI) [47] is proposed to achieve *high-order* mutual information maximization. Given a graph $g = (\mathbf{A}, \mathbf{X})$, it first applies an augmentation transformation $\mathcal{T}(\cdot)$ to obtain $\widetilde{g} = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}) = \mathcal{T}(\mathbf{A}, \mathbf{X})$. Then a shared graph encoder $f_\theta(\cdot)$ is applied to obtain node embedding matrices $\widetilde{\mathbf{H}} = f_\theta(\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}})$ and $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$. Beside, a $\mathrm{READOUT}()$ function is used to obtain the graph-level representaion $\mathbf{h}_g = \mathrm{READOUT}(\mathbf{H})$. Finally, the learning objective is defined as follows

$$\max_{\theta, \lambda} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left[\mathcal{MI}\left(\mathbf{h}_g, \mathbf{h}_i\right) + \mathcal{MI}\left(\mathbf{x}_i, \mathbf{h}_i\right) + \mathcal{MI}\left(\mathbf{h}_i, \mathbf{m}_i\right)\right] \tag{50}$$

where $\mathbf{m}_i = \sigma(\mathbf{W}_m[\sigma(\mathbf{W}_h \mathbf{x}_i); \sigma(\mathbf{W}_g \mathbf{h}_g)])$, and $\lambda = (\mathbf{W}_m, \mathbf{W}_h, \mathbf{W}_g)$ are the weight parameters. The negative samples to contrast with $\mathbf{h}_g$ in the *first* term is $Neg(\mathbf{h}_g) = \widetilde{\mathbf{h}}_i$. The negative samples to contrast with $\mathbf{x}_i$ in the *second* term is $Neg(\mathbf{x}_i) = \widetilde{\mathbf{h}}_i$. The negative samples to contrast with $\mathbf{h}_i$ in the *third* term

is $Neg(\mathbf{h}_i) = \widetilde{\mathbf{m}}_i = \sigma(\mathbf{W}_m[\sigma(\mathbf{W}_h\widetilde{\mathbf{x}}_i); \sigma(\mathbf{W}_g\mathbf{h}_g)])$.

Deep Multiplex Graph Infomax (DMGI) [69] extents the idea of DGI to multiplex graphs where nodes are connected by multiple types of relations. For each relation type $r \in \mathcal{R}$ (corresponding to a relation graph $g^{(r)} = (\mathbf{A}^{(r)}, \mathbf{X})$), a relation-type graph encoder $f_{\theta_r}(\cdot)$ is applied to obtain the relation-specific node embedding matrix $\mathbf{H}^{(r)} = f_{\theta_r}(\mathbf{A}^{(r)}, \mathbf{X})$. Besides, it employs a READOUT function to obatain the graph-level representation $\mathbf{h}_{g^{(r)}} = \text{READOUT}(\mathbf{H}^{(r)})$. Finally, it independently maximizes the MI between the node embeddings $\mathbf{H}^{(r)} = \left\{\mathbf{h}_1^{(r)}, \mathbf{h}_2^{(r)}, \ldots, \mathbf{h}_N^{(r)}\right\}$ and the graph-level summary $\mathbf{h}_{g^{(r)}}$ pertaining to each graph $g^{(r)}$ ($r \in \mathcal{R}$), and the learning objective is defined as follows:

$$\max_{\theta_r} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{MI}\left(\mathbf{h}_{g^{(r)}}, \mathbf{h}_i^{(r)}\right), r \in \mathcal{R} \tag{51}$$

where the negative samples to contrast with $\mathbf{h}_{g^{(r)}}$ is $Neg(\mathbf{h}_{g^{(r)}}) = \{\mathbf{h}_j^{(r)}\}_{v_j \in \mathcal{V}, j \neq i}$. Similarly, it can learn another embedding matrix from the augmented graph $\widetilde{g}^{(r)} = (\widetilde{\mathbf{A}}^{(r)}, \widetilde{\mathbf{X}}) = \mathcal{T}(\mathbf{A}^{(r)}, \mathbf{X})$ and also maximize the MI of the node embeddings $\widetilde{\mathbf{H}}^{(r)} = \left\{\widetilde{\mathbf{h}}_1^{(r)}, \widetilde{\mathbf{h}}_2^{(r)}, \ldots, \widetilde{\mathbf{h}}_N^{(r)}\right\}$ and the graph-level summary $\widetilde{\mathbf{h}}_{\widetilde{g}^{(r)}}$. However, as each $\mathbf{H}^{(r)}$ is trained independently for each $r \in \mathcal{R}$, these embedding matrices may fail to take advantage of the multiplexity of the network. To this end, we learn another consensus embedding matrix $\mathbf{Z}$ on which every relation-specific node embedding matrix $\mathbf{H}^{(r)}$ and $\widetilde{\mathbf{H}}^{(r)}$ can agree with each other by optimizing the following learning objective:

$$\ell_{\text{cs}} = \left[\mathbf{Z} - \frac{1}{|\mathcal{R}|}\sum_{r \in \mathcal{R}} \mathbf{H}^{(r)}\right]^2 - \left[\mathbf{Z} - \frac{1}{|\mathcal{R}|}\sum_{r \in \mathcal{R}} \widetilde{\mathbf{H}}^{(r)}\right]^2 \tag{52}$$

where $\mathbf{Z}$ is defined as a set of *trainable* parameters.

MVGRL [37] maximize the MI between the cross-view representations of nodes and large-scale graphs. For each graph $g = (\mathbf{A}, \mathbf{X}) \in \mathcal{G}$, it first applies an edge diffusion augmentation to obtain $\widetilde{g} = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}) = \mathcal{T}(\mathbf{A}, \mathbf{X})$ and then samples two subgraph $g^{(1)} = (\mathbf{A}^{(1)}, \mathbf{X}^{(1)}) = \mathcal{T}_1(\mathbf{A}, \mathbf{X})$ and $g^{(2)} = (\mathbf{A}^{(w)}, \mathbf{X}^{(2)}) = \mathcal{T}_2(\mathbf{A}, \mathbf{X})$ from it. Then two graph encoder $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$ and a shared prejection head $g_{\omega_1}(\cdot)$ are applied to obtain node embedding matrices $\mathbf{H}^{(1)} = g_{\omega_1}(f_{\theta_1}(\mathbf{A}^{(1)}, \mathbf{X}^{(1)}))$ and $\mathbf{H}^{(2)} = g_{\omega_2}(f_{\theta_1}(\mathbf{A}^{(2)}, \mathbf{X}^{(2)}))$. In addition, a READOUT function and another prejection head $g_{\omega_2}(\cdot)$ are use to obtain graph-level representation $\mathbf{h}_g^{(1)} = f_{\omega_2}(\text{READOUT}(\mathbf{H}^{(1)}))$ and $\mathbf{h}_g^{(2)} = f_{\omega_2}(\text{READOUT}(\mathbf{H}^{(2)}))$. The learning objective is defined as follows:

$$\max_{\theta_1, \theta_2, \omega_1, \omega_2} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left[\mathcal{MI}(\mathbf{h}_g^{(1)}, \mathbf{h}_i^{(2)}) + \mathcal{MI}(\mathbf{h}_g^{(2)}, \mathbf{h}_i^{(1)})\right] \tag{53}$$

where the negative samples to contrast with $\mathbf{h}_g^{(1)}$ is $Neg(\mathbf{h}_g^{(1)}) = \{\mathbf{h}_j^{(2)}\}_{v_j \in \mathcal{V}, j \neq i}$ and the negative samples to contrast with $\mathbf{h}_g^{(2)}$ is $Neg(\mathbf{h}_g^{(2)}) = \{\mathbf{h}_j^{(1)}\}_{v_j \in \mathcal{V}, j \neq i}$.

HDGI [48] A meta-path of length l is a sequence of nodes connected with heterogeneous edges, i.e., $\Phi : v_1 \xrightarrow{t_1} v_2 \xrightarrow{t_2} \ldots \xrightarrow{t_l} v_l$, where $t_l \in \mathcal{T}^e$ denote the type of l-th edge in the meta-path. Given a meta-path Φ, if there exist instances of the meta-path Φ between node v_i and node v_j, it defines that v_i and v_j are connected neighbors based on the meta-path Φ, thus obtaining a *meta-path based adjacent matrix* $\mathbf{A}^{\Phi}$. Given a meta-path set $\{\Phi_k\}_{k=1}^K$, it can obtain K meta-path based adjacent matrix $\left\{\mathbf{A}^{\Phi_k}\right\}_{k=1}^K$. The HDGI first applies an augmentation transformation to obtain $\widetilde{\mathbf{X}}, \{\widetilde{\mathbf{A}}^{\Phi_k}\}_{k=1}^K = \mathcal{T}(\mathbf{X}, \{\mathbf{A}^{\Phi_k}\}_{k=1}^K)$. Then K graph encoder are applied to obtain node embedding matrices $\mathbf{H}^{(k)} = f_{\theta_k}(\mathbf{A}^{\Phi_k}, \mathbf{X})$ and $\widetilde{\mathbf{H}}^{(k)} = f_{\theta_k}(\widetilde{\mathbf{A}}^{\Phi_k}, \widetilde{\mathbf{X}})$ $(1 \leq k \leq K)$. To obtain the more general node representations, the attention mechanism can be applied to fuse these representations $\mathbf{H}^{fuse} = \sum_{k=1}^K s_k \mathbf{H}^{(k)}$, where the attention scores $\{s_k\}_{k=1}^N$ are defined as follow

$$s_k = \frac{\exp(e^{(k)})}{\sum_{k=1}^K \exp(e^{(k)})}, e^{(k)} = \frac{1}{N}\sum_{i=1}^N \tanh\left(\mathbf{q}^T[\mathbf{W}\mathbf{h}_i^{(k)} + \mathbf{b}]\right) \tag{54}$$

where $1 \leq k \leq K$, $\mathbf{W}$ and $\mathbf{b}$ are shared weight matrix and shared bias vector, and $\mathbf{q}$ is a shared attention vector. Similarity, it can obtain the fused representation $\widetilde{\mathbf{H}}^{fuse} = \sum_{k=1}^K \widetilde{s}_k \widetilde{\mathbf{H}}^{(k)}$. Besides, a READOUT$(\cdot)$ function are further applied to obtain global-level representation $\mathbf{h}_g = \text{READOUT}(\mathbf{H}^{fuse})$. The objective is defined as:

$$\max_{\theta_1, \theta_2, \cdots, \theta_K} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{MI}\left(\mathbf{h}_g, \mathbf{h}_i^{fuse}\right) \tag{55}$$

where the negative samples to contrast with $\mathbf{h}_g$ is $Neg(\mathbf{h}_g) = \widetilde{\mathbf{h}}_j^{fuse}$.

3.3.2.2 Local-Context Contrasting

SUBG-CON [38] is proposed by utilizing the strong correlation between central (anchor) nodes and their surrounding subgraphs to capture contextual structure information. Given a graph $g = (\mathbf{A}, \mathbf{X})$, SUBG-CON first picks up an anchor node set $\mathcal{S}$ from $\mathcal{V}$ and then samples their context subgraph $\{g_i = (\mathbf{A}^{(i)}, \mathbf{X}^{(i)})\}_{i=1}^{|\mathcal{S}|}$ by the importance sampling strategy. Then a shared graph encoder $f_\theta(\cdot)$ and a READOUT$(\cdot)$ function are applied to obtain node embedding matrices $\{\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \cdots, \mathbf{H}^{(|\mathcal{V}|)}\}$ where $\mathbf{H}^{(i)} = f_\theta(\mathbf{A}^{(i)}, \mathbf{X}^{(i)})$ and graph-level representations $\{\mathbf{h}_{g_1}, \mathbf{h}_{g_2}, \cdots, \mathbf{h}_{g_{|\mathcal{V}|}}\}$ where $\mathbf{h}_{g_i} = \text{READOUT}(\mathbf{H}^{(i)})$. Finally, the learning objective is defined as follows

$$\max_{\theta} \frac{1}{|\mathcal{S}|} \sum_{v_i \in \mathcal{S}} \mathcal{MI}\left(\mathbf{h}_i^{(i)}, \mathbf{h}_{g_i}\right) \tag{56}$$

where $\mathbf{h}_i^{(i)}$ is the node representation of anchor node v_i in the node embedding matrix $\mathbf{H}^{(i)}$. The negative samples to contrast with $\mathbf{h}_i^{(i)}$ is $Neg(\mathbf{h}_i^{(i)}) = \{\mathbf{h}_{g_j}\}_{v_j \in \mathcal{S}, j \neq i}$.

Context Prediction [30] is proposed to maps nodes appearing in similar structural contexts to similar embeddings. It first picks up an anchor node set $\mathcal{S}$ from $\mathcal{V}$. For a given anchor node $v_i \in \mathcal{S}$, it defines its *K-hop neighborhood graph* $g_i^{(1)} = (\mathbf{A}_i^{(1)}, \mathbf{X}_i^{(1)})$ as all nodes and edges that are at most

K-hops away from v_i in the graph. The *context graph* represents a subgraph $g_2 = (\mathbf{A}_i^{(2)}, \mathbf{X}_i^{(2)})$ that is between r_1-hops (we require $r_1 < K$) and r_2-hops away from v_i (i.e., it is a ring of width $r_2 - r_1$). Two graph encoder $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$ are then applied to encode two graphs as node embedding matrix $\mathbf{H}_i^{(1)} = f_{\theta_1}(\mathbf{A}_i^{(1)}, \mathbf{X}_i^{(1)})$ and $\mathbf{H}_i^{(2)} = f_{\theta_2}(\mathbf{A}_i^{(2)}, \mathbf{X}_i^{(2)})$. Besides, a READOUT$(\cdot)$ function are further applied to obtain one subgraph-level representation, as follows

$$\mathbf{h}_i^{(g)} = \frac{1}{|\mathcal{M}_i|} \sum_{v_m \in \mathcal{M}_i} (\mathbf{H}_i^{(2)})_{v_m} \tag{57}$$

where $\mathcal{M}_i$ is the set of those nodes that are shared between the neighborhood graph $g_i^{(1)}$ and the context graph $g_i^{(2)}$, and we refer to those nodes as *context anchor nodes*. Finally, the learning objective is defined as follows

$$\max_{\theta_1, \theta_2} \frac{1}{|\mathcal{S}|} \sum_{v_i \in \mathcal{S}} \mathcal{MI}\left((\mathbf{H}_i^{(1)})_{v_i}, \mathbf{h}_i^{(g)}\right) \tag{58}$$

where $(\mathbf{H}_i^{(1)})_{v_i}$ is the node representation of anchor node v_i in the node embedding matrix $\mathbf{H}_i^{(1)}$. The negative samples to contrast with $(\mathbf{H}_i^{(1)})_{v_i}$ is $Neg((\mathbf{H}_i^{(1)})_{v_i}) = \{\mathbf{h}_j^{(g)}\}_{v_j \in \mathcal{S}, j \neq i}$.

Graph InfoClust (GIC) [70] relies on a framework similar to DGI. However, in addition to contrast node-global views, GIC also maximize the MI between node representations and their corresponding cluster embeddings. Give a graph $g = (\mathbf{A}, \mathbf{X})$, it first applies an augmentation transformation to obtain $\widetilde{g} = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}) = \mathcal{T}(\mathbf{A}, \mathbf{X})$. Then a shared graph encoder $f_\theta(\cdot)$ is applied to obtain node embedding matrix $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$ and $\widetilde{\mathbf{H}} = f_\theta(\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}})$. Furthermore, an unsupervised clustering algorithm is used to group nodes into K clusters $\mathcal{C} = \{C_1, C_2, \cdots, C_K\}$, and we obtain the cluster centers by the following

$$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{v_i \in C_k} \mathbf{h}_i, k = 1, 2, \cdots, K \tag{59}$$

To compute the cluster embedding $\mathbf{z}_i$ for each node v_i, we apply a weighted average of the summaries of the cluster centers to which node v_i belongs, as follows

$$\mathbf{z}_i = \sigma\left(\sum_{k=1}^{K} r_{ik} \boldsymbol{\mu}_k\right) \tag{60}$$

where r_{ik} is the probability that node v_i is assigned to cluster k, and is a soft-assignment value with $\sum_k r_{ik} = 1, \forall i$. For example, $r_{i,k}$ can be defined as $r_{i,k} = \frac{\exp(\mathbf{h}_i \boldsymbol{\mu}_k^T)}{\sum_{j=1}^{K} \exp(\mathbf{h}_i \boldsymbol{\mu}_j^T)}$. Finally, the learning objective is defined as follows

$$\max_{\theta} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{MI}\left(\mathbf{h}_i, \mathbf{z}_i\right) \tag{61}$$

where the negative samples to contrast with $\mathbf{h}_i$ is $Neg(\mathbf{h}_i) = \{\mathbf{z}_j\}_{v_j \in \mathcal{V}, j \neq i}$.

3.3.2.3 Context-Global Contrasting

MICRO-Graph [56]. The key challenge to conducting subgraph-level contrastive is to sample semantically informative subgraphs. For molecular graphs, the graph motifs, which are frequently-occurring subgraph patterns (e.g. functional groups) can be exploited for better subgraph sampling. Specifically, the motif learning is formulated as a differentiable clustering problem, and EM-clustering is adopted to group significant subgraphs into several motifs, thus obtaining a motifs table. Given two graph $g^{(1)} = (\mathbf{A}^{(1)}, \mathbf{X}^{(1)}), g^{(2)} = (\mathbf{A}^{(2)}, \mathbf{X}^{(2)}) \in \mathcal{G}$, it first applies a shared graph encoder $f_\theta(\cdot)$ to learn their node embedding matrices $\mathbf{H}^{(1)} = f_\theta(\mathbf{A}^{(1)}, \mathbf{X}^{(1)})$ and $\mathbf{H}^{(2)} = f_\theta(\mathbf{A}^{(2)}, \mathbf{X}^{(2)})$. Then it leverages learned motifs table to sample K motif-like subgraphs from $g^{(1)}$ and $g^{(2)}$ and obtain their correspongding embedding matrix $\{\mathbf{H}_1^{(1)}, \mathbf{H}_2^{(1)}, \cdots, \mathbf{H}_K^{(1)}\}$ and $\{\mathbf{H}_1^{(2)}, \mathbf{H}_2^{(2)}, \cdots, \mathbf{H}_K^{(2)}\}$. Then a READOUT function is applied to obtain graph-level and subgraph-level representations, denoted as $\mathbf{h}_g^{(1)}$, $\{\mathbf{h}_1^{(1)}, \mathbf{h}_2^{(1)}, \cdots, \mathbf{h}_K^{(1)}\}$ and $\mathbf{h}_g^{(2)}$, $\{\mathbf{h}_1^{(2)}, \mathbf{h}_2^{(2)}, \cdots, \mathbf{h}_K^{(2)}\}$. Finally, the objective is defined as

$$\max_{\theta} \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \sum_{k=1}^{K} \left[\mathcal{MI}\left(\mathbf{h}_g^{(1)}, \mathbf{h}_k^{(1)}\right) + \mathcal{MI}\left(\mathbf{h}_g^{(2)}, \mathbf{h}_k^{(2)}\right)\right] \tag{62}$$

where the negative samples to contrast with $\mathbf{h}_g^{(1)}$ is $Neg(\mathbf{h}_g^{(1)}) = \{\mathbf{h}_j^{(2)}\}_{j=1}^{K}$ and the negative samples to contrast with $\mathbf{h}_g^{(2)}$ is $Neg(\mathbf{h}_g^{(2)}) = \{\mathbf{h}_j^{(1)}\}_{j=1}^{K}$

InfoGraph [71] aims to obtain embeddings *at the whole graph level* for unsupervised learning. Give a graph $g = (\mathbf{A}, \mathbf{X})$, it first applies an augmentation to obtain $\widetilde{g} = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}) = \mathcal{T}(\mathbf{A}, \mathbf{X})$. Then a shared L-layer graph encoder $f_\theta(\cdot)$ is applied to obtain node embedding matrix sequences $\{\mathbf{H}^{(l)}\}_{l=1}^{L}$ and $\{\widetilde{\mathbf{H}}^{(l)}\}_{l=1}^{L}$ obtain from *each layer*. Then it concats the representations learned from each layer, $\mathbf{h}_i = \text{CONCAT}(\{\mathbf{h}_i^{(l)}\}_{l=1}^{L})$ and $\widetilde{\mathbf{h}}_i = \text{CONCAT}(\{\widetilde{\mathbf{h}}_i^{(l)}\}_{l=1}^{L})$, where $\mathbf{h}_i^{(l)}$ is the embedding of node v_i in the node embedding matrix $\mathbf{H}^{(l)}$ obtained from the l-th layer of the graph encoder. In addition, a READOUT function is used to obatain graph-level representations $\mathbf{h}_g = \text{READOUT}(\{\mathbf{h}_i\}_{i=1}^{N})$. Finally, the learning objective is defined as follows

$$\max_{\theta} \sum_{g \in \mathcal{G}} \frac{1}{|g|} \sum_{v_i \in g} \mathcal{MI}\left(\mathbf{h}_g, \mathbf{h}_i\right) \tag{63}$$

where the negative samples to contrast with $\mathbf{h}_g$ is $Neg(\mathbf{h}_g) = \{\widetilde{\mathbf{h}}_i\}_{v_i \in \mathcal{V}}$.

SUGAR [72] is a novel hierarchical subgraph-level framework, which encourages subgraph embedding to be mindful of the global structural properties by maximizing their mutual information. For a given $g = (\mathbf{A}, \mathbf{X}) \in \mathcal{G}$, it first applies an augmentation transformation to obtain $\widetilde{g} = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}) = \mathcal{T}(\mathbf{A}, \mathbf{X})$. Then it samples subgraphs from each graph and collect them into a subgraph candidate pool. Besides, a shared graph encoder $f_\gamma(\cdot)$ and a READOUT function are applied to encode these subgraphs into subgraph-level representations $\{\mathbf{h}_1, \mathbf{h}_2, \cdots\}$ and $\{\widetilde{\mathbf{h}}_1, \widetilde{\mathbf{h}}_2, \cdots\}$. Next, K striking subgraphs are selected from candidate pool by a reinforcement learning module and pooled into a sketched graph where each node v_k $(1 \leq k \leq K)$ corresponds to a subgraph with embedding $\mathbf{h}_k$ for graph g ($\widetilde{\mathbf{h}}_k$ for graph $\widetilde{g}$).

Finally, the learning objective is defined as follows

$$\max_{\theta} \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \sum_{k=1}^{K} \mathcal{MI}\left(\mathbf{h}_g, \mathbf{h}_k\right) \tag{64}$$

where $\mathbf{h}_g = \text{READOUT}(\{\mathbf{h}_k\}_{k=1}^K)$ and the negative samples to contrast with $\mathbf{h}_g$ is $Neg(\mathbf{h}_g) = \{\widetilde{\mathbf{h}}_j\}_{j=1}^K$.

BiGi [54] is specifically designe for bipartite graph, where the class label y_i of each node v_i is already known and $y_i \in \{0, 1\}$. For a given $g = (\mathbf{A}, \mathbf{X})$, it first applies an structure-based augmentation to obtain $\widetilde{g} = (\widetilde{\mathbf{A}}, \mathbf{X}) = \mathcal{T}(\mathbf{A}, \mathbf{X})$. Then a shared graph encoder $f_\theta(\cdot)$ is applied to obtain $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$ and $\widetilde{\mathbf{H}} = f_\theta(\widetilde{\mathbf{A}}, \mathbf{X})$. Beisde, it can obtain a graph-level representation from $\mathbf{H}$ as follows

$$\mathbf{h}_g = \sigma\Big(\frac{1}{|\mathcal{V}^{(1)}|} \sum_{v_i \in \mathcal{V}^{(1)}} \mathbf{h}\Big) \Big\| \sigma\Big(\frac{1}{|\mathcal{V}^{(2)}|} \sum_{v_i \in \mathcal{V}^{(2)}} \mathbf{h}_i\Big) \tag{65}$$

where $\mathcal{V}^{(1)} = \{v_i | v_i \in \mathcal{V}, y_i = 0\}$ and $\mathcal{V}^{(2)} = \{v_i | v_i \in \mathcal{V}, y_i = 1\}$. For a specific edge $(v_i, v_j) \in \mathcal{E}$, it first performs the ege-nets sampling to obtain two subgraph (centered at node v_i and v_j), and then gets their node feature matrix $\mathbf{H}^{(i)}$ and $\mathbf{H}^{(j)}$ from $\mathbf{H}$ directly. Then a subgraph-level attention mechanism (similar to GAT) is applied to obatain two subgraph-level representation $\mathbf{h}_i = Att_\gamma(\mathbf{H}^{(i)})$ and $\mathbf{h}_j = Att_\gamma(\mathbf{H}^{(j)})$. Finally, $\mathbf{h}_i$ and $\mathbf{h}_j$ are fused to obtain $\mathbf{h}_{i,j} = [\mathbf{h}_i \| \mathbf{h}_j]$. Similarity, it can obtain the fused representation $\widetilde{\mathbf{h}}_{i,j}$ from $\widetilde{\mathbf{H}}$. The learning objective is defined as follows:

$$\max_{\theta, \gamma} \frac{1}{|\mathcal{E}|} \sum_{(v_i, v_j) \in \mathcal{E}} \mathcal{MI}\left(\mathbf{h}_g, \mathbf{h}_{i,j}\right) \tag{66}$$

where the negative samples to contrast with $\mathbf{h}_g$ is $Neg(\mathbf{h}_g) = \widetilde{\mathbf{h}}_{i,j}$.

Head-Tail Contrastive (HTC) [73] is proposed to enhance the graph-level representations learned by GNNs. Given a graph $g_i = (\mathbf{A}_i, \mathbf{X}_i) \in \mathcal{G}$, it first applies an augmentation transformation $\mathcal{T}(\cdot)$ to obtain $\widetilde{g}_i = (\widetilde{\mathbf{A}}_i, \widetilde{\mathbf{X}}_i) = \mathcal{T}(\mathbf{A}_i, \mathbf{X}_i)$. Then a shared graph encoder $f_{\theta_1}(\cdot)$ are applied to g_i and $\widetilde{g}_i$ to obtain node embedding matirces $\mathbf{H}_i$ and $\widetilde{\mathbf{H}}_i$. Besides, a READOUT$(\cdot)$ function is used to obtain graph-level representations $\mathbf{h}_{g_i}$ and $\widetilde{\mathbf{h}}_{\widetilde{g}_i}$. Moreover, HTC samples S subsets from $\mathbf{H}_i$: $\{\mathbf{H}_i^{(1)}, \mathbf{H}_i^{(1)}, \cdots, \mathbf{H}_i^{(S)}\}$ which corresponds to subgraphs $\{g_i^{(1)}, g_i^{(1)}, \cdots, g_i^{(S)}\}$. Then it calculates a subgraph-level representation by

$$\overline{\mathbf{h}}_{g_i} = f_{\theta_2}([\mathbf{h}_{g_i^{(1)}} \|; \mathbf{h}_{g_i^{(2)}} |; \cdots \|; \mathbf{h}_{g_i^{(S)}}]) \tag{67}$$

where $\mathbf{h}_{g_i^{(s)}} = \text{READOUT}(\mathbf{H}_i^{(s)})(1 \le s \le S)$ and $f_{\theta_2}(\cdot)$ is a $(S, 1)$ size convolution kernel. Finally, the learning objective is defined as follows

$$\max_{\theta_1, \theta_2} \frac{1}{|\mathcal{G}|} \sum_{g_i \in \mathcal{G}} \mathcal{MI}\left(\mathbf{h}_{g_i}, \overline{\mathbf{h}}_{g_i}\right) \tag{68}$$

where the negative samples to contrast with $\mathbf{h}_{g_i}$ is $Neg(\mathbf{h}_{g_i}) = \{\mathbf{h}_{g_j}\}_{g_j \in \mathcal{G}, j \neq i} \cup \widetilde{\mathbf{h}}_{\widetilde{g}_i}$.

3.3.3 Additional Methods

In this subsection, we continue to introduce some contrastive methods for graph SSL, but due to space limitations, we will not provide their detailed mathematical formulas, even though they may be one of the above categories. These methods including: (1) methods that are essentially the same as the frameworks already presented above with only minor differences; (2) methods that have not been accepted and only publicly available on platforms such as arxiv, OpenReview, etc.; (3) application methods, where graph SSL is only the techniques or tricks they adopt and is not the focus of their works; (4) methods with only minor performance improvement on the benchmark datasets. In the following generative and contrastive sections, we will take the same manner to present those *relatively less important* works.

COAD [23] applies graph SSL to expert linking, which aims at linking any external information of persons to experts in AMiner. Specifically, it first pre-trains the model by *local-local* contrastive learning with the triplet margin loss and then fine-tunes the model by adversarial learning to improve the model transferability. **Contrast-Reg** [46] is a lightweight *local-local* contrastive regularization term that adopts the InfoNCE loss to contrasts the node representation similarities of semantically similar (positive) pairs against those of negative pairs. Extensive theoretical analysis demonstrates that Contrast-Reg avoids the high scales of node representations' norms and the high variance among them to improve the generalization performance. **DITNet** [74] propose an end–to–end model to predict drug-target interactions on heterogeneous graphs. Specifically, it learns high quality representations for downstream tasks by *local-gobal* and *context-global* contrasting. **C-SWM** [75] models the structured environments, such as multi-object systems, as graph neural networks, and then utilizes a *local-local* contrastive approach to perform representation learning from environment interactions without supervision.

GraphLog [76] is a unified graph SSL framework. Besides the *local-local* (similar to GRACE) contrasting and *global-global* contrasting (similar to GraphGL), GraphLoG leverages an RPCL[77]-based clustering , e.g. K-means clustering, to learn hierarchical prototypes of the data and then perform *local-context* contrasting (similar to GIC) for each hierarchical layer. **MHCN** [78] uses a DGI-like auxiliary task to enhance social recommendation by leveraging high-order user relations. However, considering that the DGI-like local-graph contrasting stays at a coarse level and there is no guarantee that the encoder can distill sufficient information from the input data, MHCN extends the DGI to a fine-grained *local-context* contrasting level by exploiting the hierarchical structure in hypergraphs. **EGI** [53] considers transfer learning on graphs, i.e., the pre-training of graph neural networks. Specifically, unlike DGI that models the local-global mutual information, EGI samples a set of ego-subgraph and then directly optimizes the *local-context* MI maximization between the structural input and output of GNN, with a particular focus on the structural information.

3.4 Contrasive Objectives

The contrastive learning treats two representations *(views)* $\mathbf{h}_i$ and $\mathbf{h}_j$ as random variables and maximize the mutual

information between them, given by

$$\mathcal{MI}(\mathbf{h}_i, \mathbf{h}_j) = \mathbb{E}_{p(\mathbf{h}_i,\mathbf{h}_j)} \left[\log \frac{p(\mathbf{h}_i, \mathbf{h}_j)}{p(\mathbf{h}_i)p(\mathbf{h}_j)} \right] \tag{69}$$

To computationally estimate the mutual information in contrastive learning, three lower-bound forms of the mutual information are derived, and then the mutual information is maximized indirectly by maximizing their lower-bounds.

Donsker-Varadhan Estimator. The Donsker-Varadhan (DV) estimator [79, 80], also known as the DV representation of the KL divergence, is a *lower-bound* to the mutual information, defined as

$$\begin{aligned} \mathcal{MI}_{DV}(\mathbf{h}_i, \mathbf{h}_j) =& \mathbb{E}_{p(\mathbf{h}_i,\mathbf{h}_j)} \left[\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) \right] \\ & - \log \mathbb{E}_{p(\mathbf{h}_i)p(\mathbf{h}_j)} \left[e^{\mathcal{D}(\mathbf{h}_i,\mathbf{h}_j)} \right] \end{aligned} \tag{70}$$

where $p(\mathbf{h}_i, \mathbf{h}_j)$ denotes the joint distribution of two representations $\mathbf{h}_i$, $\mathbf{h}_j$, and $p(\mathbf{h}_i)p(\mathbf{h}_j)$ denotes the product of marginals. $\mathcal{D} : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ is a discriminator that maps the representations of two views $\mathbf{h}_i$, $\mathbf{h}_j$ to an agreement score. Generally, the discriminator $\mathcal{D}$ can optionally apply an additional prediction head $g_\omega(\cdot)$ to map $\mathbf{h}_i$ to $\mathbf{z}_i = g_\omega(\mathbf{h}_i)$ before computing agreement scores, where $g_\omega(\cdot)$ can be a linear mapping, a nonlinear mapping (e.g., MLP), or even a non-parametric identical mapping ($\mathbf{z}_i = \mathbf{h}_i$). The discriminator $\mathcal{D}$ can be taken in various forms, i.e., the standard inner product $\mathcal{D}(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T \mathbf{z}_j$, the inner product with temperature parameter τ $\mathcal{D}(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T \mathbf{z}_j / \tau$, the cosine similarity $\mathcal{D}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^T \mathbf{z}_j}{||\mathbf{z}_i|| ||\mathbf{z}_j||}$, or the gaussian similarity $\mathcal{D}(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\frac{||\mathbf{z}_i - \mathbf{z}_j||_2^2}{2\sigma^2}\right)$.

Jensen-Shannon Estimator. Replacing the KL-divergence with the JS-divergence, we derive another Jensen-Shannon (JS) estimator [81] which can estimate and optimize the mutual information more efficiently. The Jensen-Shannon (JS) estimator is also a *lower-bound* estimation to the mutual information, defined as

$$\begin{aligned} \mathcal{MI}_{JS}(\mathbf{h}_i, \mathbf{h}_j) =& \mathbb{E}_{p(\mathbf{h}_i,\mathbf{h}_j)} \left[\log \left(\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) \right) \right] \\ & - \log \mathbb{E}_{p(\mathbf{h}_i)p(\mathbf{h}_j)} \left[log\Big(1 - \mathcal{D}(\mathbf{h}_i, \mathbf{h}_j)\Big) \right] \end{aligned} \tag{71}$$

Let $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) = \text{sigmod}(\mathcal{D}'(\mathbf{h}_i, \mathbf{h}_j))$, the Equ.71 can be re-writed as a softplus (SP) version [37, 71]:

$$\begin{aligned} \mathcal{MI}_{SP}(\mathbf{h}_i, \mathbf{h}_j) =& \mathbb{E}_{p(\mathbf{h}_i,\mathbf{h}_j)} \Big[-sp\Big(-\mathcal{D}'(\mathbf{h}_i, \mathbf{h}_j) \Big) \Big] \\ & - \log \mathbb{E}_{p(\mathbf{h}_i)p(\mathbf{h}_j)} \Big[sp\Big(\mathcal{D}'(\mathbf{h}_i, \mathbf{h}_j) \Big) \Big] \end{aligned} \tag{72}$$

where $sp(x) = \log(1 + e^x)$.

Take the graph classification as an example, $f_\gamma(\cdot)$ is a graph encoder that maps a graph $g = (\mathbf{A}, \mathbf{X}) \in \mathcal{G}$ to a graph-level representation $\mathbf{h}_g = f_\gamma(\mathbf{A}, \mathbf{X})$. Then the Equ. 71 can be re-writed as

$$\begin{aligned} \mathcal{MI}_{JS}(\mathbf{h}_i, \mathbf{h}_j) =& \mathbb{E}_{(\mathbf{A},\mathbf{X})\sim\mathcal{G}} \Big[log\Big(\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) \Big) \Big] \\ & - \log \mathbb{E}_{[(\mathbf{A},\mathbf{X}),(\mathbf{A}',\mathbf{X}')]\sim\mathcal{G}\times\mathcal{G}} \Big[log\Big(1 - \mathcal{D}(\mathbf{h}_i, \mathbf{h}_j')\Big) \Big] \end{aligned} \tag{73}$$

where $\mathbf{h}_i = f_{\gamma_i}(\mathcal{T}_i(\mathbf{A}, \mathbf{X}))$, $\mathbf{h}_j = f_{\gamma_j}(\mathcal{T}_j(\mathbf{A}, \mathbf{X}))$ are positive pair of views that comes from the sample graph $g = (\mathbf{A}, \mathbf{X})$, and $\mathbf{h}_i = f_{\gamma_i}(\mathcal{T}_i(\mathbf{A}, \mathbf{X}))$, $\mathbf{h}_j' = f_{\gamma_j}(\mathcal{T}_j(\mathbf{A}', \mathbf{X}'))$ are negative pair of views that computed from $g = (\mathbf{A}, \mathbf{X})$ and $g' = (\mathbf{A}', \mathbf{X}')$ identically and independently distributed over $\mathcal{G}$. $\mathcal{T}_i(\cdot)$ and $\mathcal{T}_j(\cdot)$ are two data augmentation strategies.

InfoNCE Estimator. InfoNCE [82, 83] is one of the most popular lower-bound to the mutual information, defined as

$$\begin{aligned} \mathcal{MI}_{NCE}(\mathbf{h}_i, \mathbf{h}_j) = \mathbb{E}_{p(\mathbf{h}_i,\mathbf{h}_j)} \Bigg[& \mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) \\ & - \mathbb{E}_{K\sim\mathcal{P}^N} \Big[\log \frac{1}{N} \sum_{\mathbf{h}_j' \in K} e^{\mathcal{D}(\mathbf{h}_i,\mathbf{h}_j')} \Big] \Bigg] \end{aligned} \tag{74}$$

where K consists of N random variables sampled from a n identical and independent distribution . For the graph classification problem, the InfoNCE is in practice computed on a mini-batch $\boldsymbol{B}$ of size $N + 1$, then the Equ. 74 can be re-writed (with $\log N$ discarded) as

$$\begin{aligned} \mathcal{MI}_{NCE} = -\frac{1}{N+1} \sum_{(\mathbf{A},\mathbf{X})\in\boldsymbol{B}} \\ \left[\log \frac{e^{\mathcal{D}(\mathbf{h}_i,\mathbf{h}_j)}}{\sum_{(\mathbf{A}',\mathbf{X}')\in\boldsymbol{B}\setminus\{(\mathbf{A},\mathbf{X})\}} e^{\mathcal{D}\left(\mathbf{h}_i,\mathbf{h}_j'\right)}} \right] \end{aligned} \tag{75}$$

where $\mathbf{h}_i$, $\mathbf{h}_j$ are positive pair of views from the sample graph $g = (\mathbf{A}, \mathbf{X})$, and $\mathbf{h}_i$ and $\mathbf{h}_j'$ are negative views that are computed from $g = (\mathbf{A}, \mathbf{X})$ and $g' = (\mathbf{A}', \mathbf{X}')$ identically and independently. NT-Xent loss [84] is a special version of the InfoNCE loss, which defines the discriminator $\mathcal{D}$ as $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{h}_i^T \mathbf{h}_j / \tau$ with temperature parameter τ.

Triplet Margin Loss. The above-mentioned three estimators and their variants estimate the *lower bound* of the mutual information. Triplet margin loss [85], on the other hand, is an *non-bounded mutual information estimator* that is not provable lower-bound to the mutual information, and optimizing it does not guarantee the maximization of the mutual information. The triplet margin loss is formalized as

$$\begin{aligned} \mathcal{MI}_{triplet}(\mathbf{h}_i, \mathbf{h}_j) = \mathbb{E}_{[(\mathbf{A},\mathbf{X}),(\mathbf{A}',\mathbf{X}')]\sim\mathcal{G}\times\mathcal{G}} \Big[\\ \max\{\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) - \mathcal{D}(\mathbf{h}_i, \mathbf{h}_j') + \epsilon, 0\} \Big] \end{aligned} \tag{76}$$

where ϵ is the margin value.

4 Generative Learning

Compared with the contrastive methods, the generative methods shwon in Fig. 1(b) are usually based on generative models and treat the rich information embedded in the data itself as a natural self-supervision. In generative methods, the prediction head $g_\omega(\cdot)$ is usually called the graph decoder, which is used to perform the pretext task of graph reconstruction. Categorized by how the reconstruction is performed, we summarize generative methods into two categories: (1) graph autoencoding that performs graph reconstruction in a once-for-all manner; (2) graph autoregressive that iteratively performs graph reconstruction. The autoencoding methods focus on reconstructing certain parts of a given graph, such as masked features, or corrupted adjacency matrices. The autoregressive methods, on the other hand, emphasize the reconstruction of the remaining parts in an iterative manner, given partial graph information, e.g. attributes or structures.

4.1 Graph Autoencoding

Since the autoencoder [86] was proposed, it has been widely used as a basic architecture for a variety of image and text data. Given restricted access to the graph, the graph autoencoder is trained to reconstruct certain parts of the input data. Depending on which parts of the input graph are given or restricted, various pretext tasks have been proposed, which will be reviewed one by one next.

Graph Completion [31]. Motivated by the success of image inpainting, graph completion is proposed as a pretext task for graph data. It first masks one node by removing part of its features, and then aims to reconstruc masked features by feeding *unmasked* node features in the neighborhood. For a given node v_i, it randomly masks its features $\mathbf{x}_i$ with $\widehat{\mathbf{x}}_i = \mathbf{x}_i \odot \mathbf{m}_i$ to obtain a new node feature matrix $\widehat{\mathbf{X}}$, and then aim to reconstruct masked features. More formally,

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \widehat{\mathbf{X}}\right) = \left\| f_\theta(\mathbf{A}, \widehat{\mathbf{X}})_{v_i} - \mathbf{x}_i \right\|^2 \tag{77}$$

Here we just take one node as an example, and the reconstruction of multiple target nodes can also be considered in practice. For the graph completion task, note that only those *unmasked* neighborhood nodes can be used to reconstruct the target node features. Intuitively, this task teaches the model to extract features from the *context*.

Node Attribute Masking [29]. This task is similar to Graph Completion, but in this case it masks and reconstructs the features of multiple nodes *simultaneously*, and it no longer requires that the neighboring node features used for message passing must be unmasked features. It first randomly masks (i.e., set equal to zero) the features of a node set $\mathcal{M}_v$. Specifically, it obtains a masked node feature matrix $\widehat{\mathbf{X}}$ where $\widehat{\mathbf{x}}_i = \mathbf{x}_i \odot \mathbf{m}_i$ for $v_i \in \mathcal{M}_v$, and then ask the model to reconstruct these masked features. More formally,

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \widehat{\mathbf{X}}\right) = \frac{1}{|\mathcal{M}_v|} \sum_{v_i \in \mathcal{M}_v} \left\| f_\theta(\mathbf{A}, \widehat{\mathbf{X}})_{v_i} - \mathbf{x}_i \right\|^2 \tag{78}$$

Since the features in most real-world datasets are high-dimensional and sparse. Hence, Principle Component Analysis (PCA) [87] can be applied in practice to reduced dense features before applying Node Attribute Masking.

Edge Attribute Masking [30]. This pretext task is specifically designed for graph data with known edge features, and it enables GNN to learn more edge relation information. Similarly, it first randomly masks the features of a edge set $\mathcal{M}_e$. Specifically, it obtains a masked edge feature matrix $\widehat{\mathbf{X}}^e$ where $\widehat{\mathbf{x}}^e_{i,j} = \mathbf{x}^e_{i,j} \odot \mathbf{m}_{i,j}$ for $(v_i, v_j) \in \mathcal{M}_e$. More formally,

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \mathbf{X}, \widehat{\mathbf{X}}^e\right) = \frac{1}{|\mathcal{M}_e|} \sum_{(v_i, v_j) \in \mathcal{M}_e} \left\| \overline{\mathbf{x}}^e_{i,j} - \mathbf{x}^e_{i,j} \right\|^2 \tag{79}$$

where $\overline{\mathbf{x}}^e_{i,j} = (\overline{\mathbf{X}}^e)_{i,j}$ and $\overline{\mathbf{X}}^e = f_\theta(\mathbf{A}, \mathbf{X}, \widehat{\mathbf{X}}^e)$.

Node Attribute and Embedding Denoising [32]. Different from Node Attribute Masking, this pretext task aims to add noise to the node features to obtain a noisy node feature matrix $\widehat{\mathbf{X}} = \mathbf{X} + N(\mathbf{0}, \mathbf{\Sigma})$, and then ask the model to reconstruct the clean node features $\mathbf{X}$. More formally,

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \widehat{\mathbf{X}}\right) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left\| f_\theta(\mathbf{A}, \widehat{\mathbf{X}})_{v_i} - \mathbf{x}_i \right\|^2 \tag{80}$$

Different from Node Attribute Denoising, which reconstructs raw features from noisy inputs, Node Embedding Denoising aims to reconstructs embeddings $\mathbf{X}$ from noisy embeddings $\widehat{\mathbf{H}} = \mathbf{H} + N(\mathbf{0}, \mathbf{\Sigma})$. More formally,

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \widehat{\mathbf{H}}\right) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left\| f_\theta(\mathbf{A}, \widehat{\mathbf{H}})_{v_i} - \mathbf{x}_i \right\|^2 \tag{81}$$

Noise is only one means of corrupting the image, in addition to blurring, graying, etc. Inspired by this, it can use *arbitrary* corruption operations $\mathcal{C}(\cdot)$ to obtain the corrupted features and then force the model to reconstruct it. For Node Attribute Denoising, it has derived one special variant $\mathcal{C}(\mathbf{X}) = \mathbf{X} + N(\mathbf{0}, \mathbf{\Sigma})$. More formally,

$$\mathcal{L}_{self}\Big(\theta, \mathbf{A}, \mathcal{C}(\mathbf{X})\Big) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left\| f_\theta\big(\mathbf{A}, \mathcal{C}(\mathbf{X})\big)_{v_i} - \mathbf{x}_i \right\|^2 \tag{82}$$

Adjacency Matrix Reconstruction [33]. The graph adjacency matrix is one of the most important information in graph data, which stores the graph structure information and the relationships between nodes. This pretext task randomly perturbs parts of the edges in a graph $\mathbf{A}$ to obtain $\widehat{\mathbf{A}}$, then requires the model to reconstruct the adjacency matrix of the input graph. More formally,

$$\mathcal{L}_{self}\left(\theta, \widehat{\mathbf{A}}, \mathbf{X}\right) = \frac{1}{|\mathcal{V}|^2} \sum_{i,j} \left\| \overline{\mathbf{A}}_{i,j} - \mathbf{A}_{i,j} \right\|^2 \tag{83}$$

where $\overline{\mathbf{A}} = f_\theta\left(\widehat{\mathbf{A}}, \mathbf{X}\right)$. During the training process, since the adjacency matrix $\mathbf{A}$ is usually a sparse matrix, we can use cross-entropy instead of MAE as loss, defined as

$$\begin{aligned}\mathcal{L}_{self}\left(\theta, \widehat{\mathbf{A}}, \mathbf{X}\right) = -\sum_{i,j} \Big[& \mathbf{A}_{i,j} \log(\overline{\mathbf{A}}_{i,j}) \\ & + (1 - \mathbf{A}_{i,j}) \log(1 - \overline{\mathbf{A}}_{i,j}) \Big]\end{aligned} \tag{84}$$

4.2 Graph Autoregressive

The autoregressive model is a linear regression model that uses a combination of random variables from previous moments to represent random variables at a later moment.

GPT-GNN [49]. In recent years, the idea of GPT [19] has also been introduced into the GNN domain. For example, GPT-GNN proposes an autoregressive framework to perform node and edge reconstruction on given graph *iteratively*. Given a graph $g_t = (\mathbf{A}_t, \mathbf{X}_t)$ with its nodes and edges randomly masked in iteration t, GPT-GNN generates one masked node X_i and its connected edges E_i to obtain a updated graph $g_{t+1} = (\mathbf{A}_{t+1}, \mathbf{X}_{t+1})$ and optimizes the likelihood of the node and edges generation in the current iteration $t+1$, with learning objective defined as

$$\begin{aligned} & p_\theta\left(\mathbf{X}_{t+1}, \mathbf{A}_{t+1} \mid \mathbf{X}_t, \mathbf{A}_t\right) \\ = & \sum_o p_\theta\left(X_i, E_i^{\neg o} \mid E_i^o, \mathbf{X}_t, \mathbf{A}_t\right) \cdot p_\theta\left(E_i^o \mid \mathbf{X}_t, \mathbf{A}_t\right) \\ = & \mathbb{E}_o\Big[p_\theta\left(X_i, E_i^{\neg o} \mid E_i^o, \mathbf{X}_t, \mathbf{A}_t\right)\Big] \\ = & \mathbb{E}_o\Big[p_\theta\left(\mathbf{X}_{t+1} \mid E_i^o, \mathbf{X}_t, \mathbf{A}_t\right) p_\theta(E_i^{\neg o} \mid E_i^o, \mathbf{X}_{t+1}, \mathbf{A}_t)\Big] \end{aligned} \tag{85}$$

where o is a variable to denote the index vector of all the observed edges within E_t in the iteration t. Thus, E_t^o denotes the observed edges in the iteration t, and $E_i^{\neg o}$ denotes the the masked edges (to be generated) in the iteration $t+1$. Finally, the graph generation process is factorized into a node *attribute generation* step $p_\theta\left(\mathbf{X}_{t+1} \mid E_i^o, \mathbf{X}_t, \mathbf{A}_t\right)$ and an *edge generation* step $p_\theta(E_i^{\neg o} \mid E_i^o, \mathbf{X}_{t+1}, \mathbf{A}_t)$. In practice, we perform node and edge generation iteratively.

4.3 Additional Methods

In this subsection, we will continue to introduce some generative methods for graph SSL, but due to space limitations, we will not provide their detailed mathematical formulas.

To tackle the cold-start problem for recommendation tasks, **Pretrain-Recsys** [88] pre-trains the model by simulating the cold-start scenarios. Specifically, it applies an attention-based meta aggregator to reduce the impact from the cold-start neighbors and takes the target *embedding reconstruction* as the pretext task. **SLAPS** [89] is a graph learning framework that tasks the *node attribute reconstruction* as the self-supervised pretext task to infer a task-specific latent structure and then apply a GNN on the inferred graph.

G-BERT [90] combines the power of GNNs and BERT to performs representation learning for medication recommendation. Specifically, G-BERT uses two BERT-like self-supervised pretext tasks, *self-prediction and dual-prediction*. The self-prediction takes the learned embedding as input to reconstruct the masked medical codes with the same type, while dual-prediction takes one type of embedding as input and tries to reconstruct the other type. **Graph-Bert** [34] is a novel graph neural network that pre-trains the model with the self-supervised *node attribute reconstruction* and *structure recovery* tasks, and then transfers the model to other downstream tasks directly or with necessary fine-tuning if any supervised label information is available.

5 Predictive Learning

The contrastive methods deal with the *inter-data* information (data-data pairs), the generative methods focus on the *intra-data* information, while the predictive methods aim to *self-generate informative labels* from the data as supervision and handle the *data-label* relationships. Categorized by how labels are obtained, we summarize predictive methods into four categories: (1) Node Property Prediction. The properties of nodes, such as node degree, are pre-calculated and used as self-supervised labels to perform prediction tasks. (2) Context-based Prediction. Local or global contextual information in the graph can be extracted as labels to aid self-supervised learning, e.g., by predicting the shortest path length between nodes, the model can capture long-distance dependencies, which is beneficial for downstream tasks such as link prediction. (3) Self-Training. Learning with the pseudo-labels obtained from the prediction or clustering in a previous stage or even randomly assigned. (4) Domain Knowledge-based Prediction. Expert knowledge or specialized tools is used in advance to analyze graph data (e.g., biological or chemical data) to obtain informative labels. A comparison of four predictive methods is shown in Fig. 5.

5.1 Node-Property Prediction (NP)

An effective way to perform predictive learning is to take advantage of the extensive implicit numerical properties within the graph, e.g. the commonly used node properties, such as node degree and local clustering coefficient [29]. Specifically, it first defines a mapping $\Omega: \mathcal{V} \rightarrow \mathcal{Y}$ to denote the extraction of *statistical labels* $y_i = \Omega(\mathbf{A}, \mathbf{X})_{v_i}$ for each node v_i from graph $g = (\mathbf{A}, \mathbf{X})$. The learning objective of this pretext task is formulated as

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \Big(f_\theta(\mathbf{A}, \mathbf{X})_{v_i} - y_i\Big)^2 \tag{86}$$

where $f_\theta(\mathbf{A}, \mathbf{X})$ is the predicted label matrix, and $f_\theta(\mathbf{A}, \mathbf{X})_{v_i}$ is the predicted label of node v_i. With different node properties, the mapping function $\Omega(\cdot)$ can have different designs. If we use node degree as a representative local node property for self-supervision, we have

$$y_i = \Omega(\mathbf{A}, \mathbf{X})_{v_i} = \sum_{j=1}^{N} \mathbf{A}_{i,j} \tag{87}$$

For the local clustering coefficients, we have

$$y_i = \Omega(\mathbf{A}, \mathbf{X})_{v_i} = \frac{2|\{(v_m, v_n) | v_m \in \mathcal{N}_i, v_n \in \mathcal{N}_i\}|}{|\mathcal{N}_i(\mathcal{N}_i - 1)|} \tag{88}$$

where the local clustering coefficient is a local coefficient describing the level of node aggregation in a graph. Beyond the above two properties, any other node property (or even a combination of them) can be used as statistical labels to perform the pretext task of Node Property Prediction.

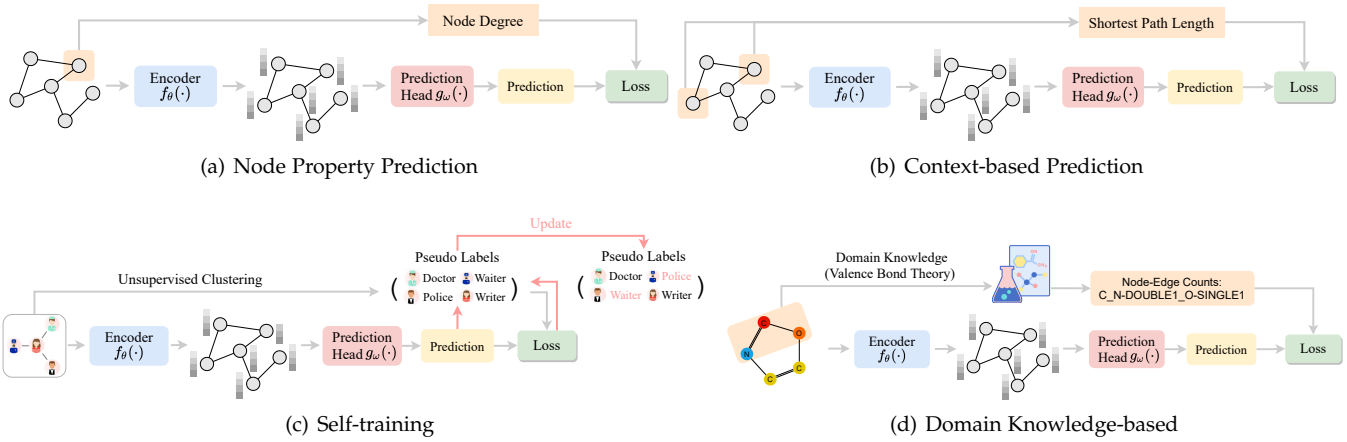
5.2 Context-based Prediction (CP)

Apart from Node Property Prediction, the underlying graph structure information can be further explored to construct a variety of regression-based or classification-based pretext tasks and thus provide self-supervised signals. We refer to this branch of methods as context-based predictive learning because it generally explores contextual information, such as adjacent geometric positions in the graph structure.

S^{2}GRL [91]. Motivated by the observation that two arbitrary nodes in a graph can interact with each other through paths of different lengths, it treats the contextual position of one node relative to the other as a source of free and effective supervisory signals. Specifically, it defines the k-hop context of node v_i as $\mathcal{C}_i^k = \{v_j | d(v_i, v_j) = k\} (k = 1, 2, \cdots, K)$, where $d(v_i, v_j)$ is the shortest path length between node v_i and node v_j. In this way, for each target node v_i, if a node $v_j \in \mathcal{C}_i^k$, then the hop count k (relative contextual position) will be assigned to node v_j as pseudo-label $y_{i,j} = k$. The learning objective is defined as predicting the hop count between pairs of nodes, as follows

$$\begin{aligned}\mathcal{L}_{self}\left(\theta, \omega, \mathbf{A}, \mathbf{X}\right) = \frac{1}{|\mathcal{V}|K} \sum_{v_i \in \mathcal{V}} \sum_{k=1}^{K} \sum_{v_j \in \mathcal{C}_i^k} \ell\Big(f_w\big(\\ f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, f_\theta(\mathbf{A}, \mathbf{X})_{v_j}\big), k\Big)\end{aligned} \tag{89}$$

where $\ell(\cdot)$ deontes the cross entropy loss and $f_\omega(\cdot)$ linearly maps the input to a 1-dimension value.



(a) Node Property Prediction (b) Context-based Prediction

(c) Self-training (d) Domain Knowledge-based

Fig. 5. A comparison of the predictive learning methods. The predictive method generally self-generates labels from graph data through some simple statistical analysis, or domain knowledge, and then designs prediction-based pretext tasks based on the self-generated labels. Categorized by how the labels areobtained, we summarize predictive learning methods forgraph data into four categories: node property prediction, context-based prediction, self-training, and domain knowledge-based prediction. **Fig. (a)**: the node property prediction pre-calculated the node properties, such as node degree and used them as self-supervised labels. **Fig. (b)**: for the context-based prediction, the local or global contextual information in the graph, such as the shortest path length between nodes can be extracted as labels to help with self-supervised learning. **Fig. (c)**: The self-learning method applies algorithms such as unsupervised clustering to obtain pseudo-labels and then updates the pseudo-label set of the previous stage based on the prediction results or losses. **Fig. (d)**: for the domain knowledge-based prediction, the domain knowledge, such as expert knowledge or specialized tools, can be used in advance to obtain informative labels.

PairwiseDistance [29]. The pretext task of PairwiseDistance aims to guide the model to preserve global topology information by predicting the shortest path length between different node pairs. More specifically, it first randomly samples a certain amount of node pairs $\mathcal{S}$ from all node pairs $\{(v_i, v_j)|v_i, v_j \in \mathcal{V}\}$ and calculates the pairwise node shortest path length $d_{i,j} = d(v_i, v_j)$ for node pairs $(v_i, v_j) \in \mathcal{S}$. Furthermore, it groups the shortest path lengths into four categories: $C_{i,j} = 0, C_{i,j} = 1, C_{i,j} = 2$, and $C_{i,j} = 3$ corresponding to $d_{i,j} = 1, d_{i,j} = 2, d_{i,j} = 3$, and $d_{i,j} \geq 3$, respectively. The learning objective can then be formulated as a multi-class classification problem, as follows

$$\mathcal{L}_{self}(\theta, \omega, \mathbf{A}, \mathbf{X}) = \frac{1}{|\mathcal{S}|} \sum_{(v_i, v_i) \in \mathcal{S}} \ell\Big(f_w(|f_\theta(\mathbf{A}, \mathbf{X})_{v_i} - f_\theta(\mathbf{A}, \mathbf{X})_{v_j}|), C_{i,j}\Big) \tag{90}$$

where $\ell(\cdot)$ denotes the cross-entropy loss and $f_\omega(\cdot)$ linearly maps the input to a 1-dimension value. Compared with the task of S^{2}GRL, the PairwiseDistance has truncated the shortest path longer than 4, mainly to avoid the excessive computational burden and to prevent very noisy ultra-long pairwise distances from dominating the optimization.

PairwiseAttrSim [29]. Due to the neighborhood-based message passing mechanism, the learned representations of two similar nodes in the graph are not necessarily similar, as opposed to two identical images that will yield the same representations in the image domain. Though we would like to utilize local neighborhoods in GNNs to enhance node feature transformation, we still wish to preserve the node pairwise similarity to some extent, rather than allowing a node's neighborhood to drastically change it. Thus, a context-based pretext task of PairwiseAttrSim is established to achieve node similarity preservation. Moreover, due to the majority of the pairwise similarity being near zero, it develops the following pair sampling strategy. Specifically, it first samples node pairs with the K highest and lowest similarities $\mathcal{S}_{i,h}$ and $\mathcal{S}_{i,s}$ for node v_i, given by

$$\begin{aligned} \mathcal{S}_{i,h} &= \Big\{(v_i, v_j) \mid s_{ij} \text{ in top-K of } \{s_{ik}\}_{k=1,k\neq i}^{N}\Big\} \\ \mathcal{S}_{i,l} &= \Big\{(v_i, v_j) \mid s_{ij} \text{ in bottom-K of } \{s_{ik}\}_{k=1,k\neq i}^{N}\Big\} \end{aligned} \tag{91}$$

where $s_{i,j}$ measures the node feature similarity between node v_i and node v_j (according to cosine similarity). Let $\mathcal{S}_i = \mathcal{S}_{i,h} \cup \mathcal{S}_{i,h}$, the learning objective can then be formulated as a regression problem, as follows

$$\mathcal{L}_{self}(\theta, \omega, \mathbf{A}, \mathbf{X}) = \frac{1}{2|\mathcal{V}|K} \sum_{v_i \in \mathcal{V}} \sum_{(v_m, v_n) \in \mathcal{S}_i} \Big\| f_w\big(f_\theta(\mathbf{A}, \mathbf{X})_{v_m}, f_\theta(\mathbf{A}, \mathbf{X})_{v_n}\big) - s_{m,n} \Big\|^2 \tag{92}$$

where $f_\omega(\cdot)$ linearly maps the input to a 1-dimension value.

Distance2Clusters [29]. The PairwiseAttrSim applies a sampling strategy to reduce the time complexity, but still involves sorting the node similarities, which is a very time-consuming operation. Inspired by various unsupervised clustering algorithms [92–100], if a set of clusters can be pre-obtained, the PairwiseAttrSim can be further simplified to predict the shortest path from each node to the anchor nodes associated with cluster centers, resulting in a novel pretext task - Distance2Clusters. Specifically, it first partitions the graph into K clusters $\{C_1, C_2, \cdots, C_K\}$ by applying some classical unsupervised clustering algorithms. Inside each cluster C_k , the node with the highest degree will be taken as the corresponding cluster center, denoted as c_k $(1 \leq k \leq K)$. Then it can calculate the distance $\mathbf{d}_i \in \mathbb{R}^K$ from node v_i to cluster centers $\{c_k\}_{k=1}^K$. The learning objective of Distance2Clusters is defined as

$$\mathcal{L}_{self}(\theta, \mathbf{A}, \mathbf{X}) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \Big\| f_\theta(\mathbf{A}, \mathbf{X})_{v_i} - \mathbf{d}_i \Big\|^2 \tag{93}$$

EdgeMask [29]. Different from masking edge features, it can also mask edge connections, but instead of reconstructing the entire adjacency matrix directly, EdgeMask takes the link prediction as a pretext task. More specifically, it first masks m_e edges $\mathcal{M}_e \in \mathcal{E}$ where $|\mathcal{M}_e| = m_e$ and also samples m_e edges $\overline{\mathcal{M}}_e \in \{(v_i, v_j)|v_i, v_j \in \mathcal{V} \text{ and } (v_i, v_j) \notin \mathcal{E}\}$. Then, learning objective of this pretext task is to predict whether there exists a link between a given node pair. More formally,

$$\begin{aligned}\mathcal{L}_{self}(\theta, \omega, \mathbf{A}, \mathbf{X}) = \frac{1}{2m_e}\Big(\\ \sum_{(v_i,v_j)\in\mathcal{M}_e} \ell\big(f_w(|f_\theta(\mathbf{A},\mathbf{X})_{v_i} - f_\theta(\mathbf{A},\mathbf{X})_{v_j}|), 1\big) + \\ \sum_{(v_i,v_j)\in\overline{\mathcal{M}}_e} \ell\big(f_w(|f_\theta(\mathbf{A},\mathbf{X})_{v_i} - f_\theta(\mathbf{A},\mathbf{X})_{v_j}|), 0\big)\Big)\end{aligned} \tag{94}$$

where $\ell(\cdot)$ denotes the cross-entropy loss and $f_\omega(\cdot)$ linearly maps the input to a 1-dimension value. The EdgeMask task aims to help GNN learn more local structural information.

TopoTER [101] amis to maximize the mutual information between topology transformations and node representations before and after the transformations, which can be relaxed to minimizing the cross entropy between the applied topology transformation types and its estimation from node representations. Given a graph $g = (\mathbf{A}, \mathbf{X})$, it first randomly samples a subset $\mathcal{S}_1$ of M connected edges and a subset $\mathcal{S}_2$ of M disconnected edges. Then it randomly removes $r \cdot M$ edges from $\mathcal{S}_1$ and adds $r \cdot M$ edges to $\mathcal{S}_2$, where r is the edge perturbation rate, to obatain a topology-perturbation graph $\widetilde{g} = (\widetilde{\mathbf{A}}, \mathbf{X})$. Next, it applies a shared graph encoder $f_\theta(\cdot)$ to obtain their node embedding matrices $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$ and $\widetilde{\mathbf{H}} = f_\theta(\widetilde{\mathbf{A}}, \mathbf{X})$. Finally, a mapping function $g_\omega(\cdot)$ is applied to the difference $\Delta\mathbf{H} = \mathbf{H} - \widetilde{\mathbf{H}}$ to predict transformation types $\widehat{\mathbf{Y}}_{i,j} = g_\omega(\Delta\mathbf{H}_i, \Delta\mathbf{H}_j) \in \mathbb{R}^4$ for any $e_{i,j} \in \mathcal{S}_1 \bigcup \mathcal{S}_2$. Then, the learning objective of this pretext task is defined as

$$\mathcal{L}_{self}(\theta, \omega, \mathbf{A}, \mathbf{X}) = -\sum_{e_{i,j}\in\mathcal{S}_1\bigcup\mathcal{S}_2}\sum_{c=0}^{3} \mathbf{Y}_{i,j}^{(c)} \log(\widehat{\mathbf{Y}}_{i,j}^{(c)}) \tag{95}$$

where c denotes four transformation types, $c = 0$: add an edge to a disconnected node pair in $\mathcal{S}_2$; $c = 1$: reomove an edge from a connected node pair in $\mathcal{S}_1$; $c = 2$: keep the connection between node pairs in $\mathcal{S}_1$; $c = 3$: keep the disconnection between node pairs in $\mathcal{S}_2$. $\mathbf{Y}_{i,j}^{(c)}$ is the ground-truth binary indicator (0 or 1) for each transformation type.

Centrality Score Ranking [36]. Node centrality is an important metric for graphs, which measures the importance of nodes based on their structural roles in the whole graph. However, in contrast to the pretext task of node property prediction, centrality scores are not comparable among different graphs with different scales. Therefore, it resorts to rank the *relative orders* between nodes and consider them as pseudo labels. Specifically, four different centrality scores $\mathcal{S} = \{$eigencentrality, betweenness, closeness, Subgraph Centrality$\}$ are used. For a centrality score $s \in \mathcal{S}$ and a node pair (v_i, v_j) with relative order $\mathbf{Y}_{i,j}^s = \mathbb{I}(s_i > s_j)$, a mapping function $f_\theta^s(\cdot)$ is applied to estimate its rank score by $\mathbf{r}^s = f_\theta^s(\mathbf{A}, \mathbf{X})$, where $\mathbf{r}_i^s$ denotes the rank score of node v_i. The probability of estimated rank order is defined as

$$\widehat{\mathbf{Y}}_{i,j}^s = \frac{\exp(\mathbf{r}_i^s - \mathbf{r}_j^s)}{1 + \exp(\mathbf{r}_i^s - \mathbf{r}_j^s)} \tag{96}$$

The learning objective of this pretext task is defined as

$$\begin{aligned}\mathcal{L}_{self}(\theta, \mathbf{A}, \mathbf{X}) = -\frac{1}{|\mathcal{S}|\cdot|\mathcal{V}|^2}\sum_{s\in\mathcal{S}}\sum_{i,j\in\mathcal{V}}\Big[\mathbf{Y}_{i,j}^s \log(\widehat{\mathbf{Y}}_{i,j}^s) \\ + (1 - \mathbf{Y}_{i,j}^s)\log(1 - \widehat{\mathbf{Y}}_{i,j}^s)\Big]\end{aligned} \tag{97}$$

Meta-path Prediction [102]. Due to the unique properties of heterogeneous graphs, some existing graph SSL methods may not be directly applied to heterogeneous graphs, which inspires us to design more effective pretext tasks. A meta-path of length l is a sequence of nodes connected with heterogeneous edges, i.e., $v_1 \xrightarrow{t_1} v_2 \xrightarrow{t_2} \dots \xrightarrow{t_l} v_l$, where $t_l \in \mathcal{T}^e$ denote the type of l-th edge in the meta-path. Given a set of node pair $\mathcal{S}$ sampled from the heterogeneous graph and K *pre-defined* meta-path type $\mathcal{M}$ ($K = |\mathcal{M}|$), this pretext task aims to predict if the two nodes $(v_i, v_j) \in \mathcal{S}$ are connected by one of the meta-paths type $m \in \mathcal{M}$. Finally, the predictions of the K meta-paths are formulated as K binary classification tasks, as follows

$$\begin{aligned}\mathcal{L}_{self}(\theta, \mathbf{A}, \mathbf{X}) = \frac{1}{|\mathcal{M}|\cdot|\mathcal{S}|}\sum_{m\in\mathcal{M}}\sum_{(v_i,v_j)\in\mathcal{S}} \\ \ell\Big(f_w\big(f_\theta(\mathbf{A},\mathbf{X})_{v_i}, f_\theta(\mathbf{A},\mathbf{X})_{v_j}\big), \mathbf{Y}_{i,j}^m\Big)\end{aligned} \tag{98}$$

where $\ell(\cdot)$ deontes the cross entropy loss, and $\mathbf{Y}_{i,j}^m$ is the ground-truth label where $\mathbf{Y}_{i,j}^m = 1$ if there exits a meta-path m between node v_i and node v_j, otherwise $\mathbf{Y}_{i,j}^m = 0$.

SLiCE [103]. Instead of the pretext task of Meta-path Prediction that requires pre-defined mate-paths, SLiCE automatically learns the composition of different meta-paths for a specific task. Specifically, it first samples a set of nodes $\mathcal{S}$ from the node set $\mathcal{V}$. Given a node in $v_i \in \mathcal{S}$, it generates a context subgraph $g_i = (\mathbf{A}_i, \mathbf{X}_i)$ around v_i and encodes the context as a low-dimensional embedding matrix $\mathbf{H}_i$. Then it randomly masks a node v_i^m in graph g_i for prediction. Therefore, the pretext task aims to maximize the probability of observing this masked node v_i^m based on the context g_i,

$$\mathcal{L}_{self}(\theta, \mathbf{A}, \mathbf{X}) = \prod_{v_i\in\mathcal{S}}\prod_{v_i^m\in g_i} p(v_i^m \mid \mathbf{H}_i, \theta) \tag{99}$$

where $p(\cdot \mid \theta)$ can in practice be approximated by a GNN model $f_\theta(\cdot)$ parameterized by θ.

Distance2Labeled [29]. Some recent work provides deep insight into existing self-supervised pretext tasks that utilize only attribute and structure information and finds that they are not always beneficial in improving the performance of downstream tasks, possibly because the information mined by the pretext tasks may have been fully exploited during the message passing by the GNN model. Thus, given partial information about downstream tasks, such as a small set of labeled nodes, we can directly explore task (label)-specific self-supervised pretext tasks. For example, we can directly

modify the pretext task of Distance2Cluster to take into consideration information from the labeled nodes. Specifically, it first calculates the average, minimum, and maximum shortest path length from node v_i to all labeled nodes in class $\{C_k\}_{k=1}^K$, resulting in a distance vector $\mathbf{d}_i \in \mathbb{R}^{3K}$. Finally, the learning objective of this pretext task can be formulated as a distance regression problem, as follows

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left\| f_\theta(\mathbf{A}, \mathbf{X})_{v_i} - \mathbf{d}_i \right\|^2 \tag{100}$$

Compared with Distance2Cluster, the pretext task of Distance2Labeled leverages the task-specific label information instead of additional unsupervised clustering algorithms to find the cluster centers, demonstrating advantages in both efficiency and performance.

ContextLabel [29]. Different from Distance2Labeled, which uses relative position as a self-supervised signal, the pretext task of ContextLabel works by constructing a *local label distribution* for each node and then asking the model to regress these distributions. It first assigns labels for unlabeled node with the Label Propagation (LP) algorithm [104], and then defines the local label distribution $\mathbf{y}_i$ for node v_i within its k-hop neighborhood where the c-th element of the label distribution vector $\mathbf{y}_{i,c}$ can be defined as,

$$\mathbf{y}_{i,c} = \frac{\left|\mathcal{N}_i^{\mathcal{V}_L}(c)\right| + \left|\mathcal{N}_i^{\mathcal{V}_U}(c)\right|}{\left|\mathcal{N}_i^{\mathcal{V}_L}\right| + \left|\mathcal{N}_i^{\mathcal{V}_U}\right|}, c = 1, 2, \cdots, C \tag{101}$$

where $\mathcal{N}_i^{\mathcal{V}_L}$ and $\mathcal{N}_i^{\mathcal{V}_U}$ are the labeled nodes and unlabeled nodes within k-hop neighborhood of node v_i, respectively. $\mathcal{N}_i^{\mathcal{V}_L}(c)$ denotes only those in the neighborhood set with pre-assigned label c, and $\mathcal{N}_i^{\mathcal{V}_U}(c)$ denotes those in the neighborhood set that are assigned label c by Label Propagation (LP) algorithm. Finally, the learning objective is defined as,

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left\| f_\theta(\mathbf{A}, \mathbf{X})_{v_i} - \mathbf{y}_i \right\|^2 \tag{102}$$

In addition to Label Propagation, there are numerous algorithms that can be used to compute pseudo-labels, such as the Iterative Classification Algorithm (ICA) [105], and even a combination of LP and ICA for better performance.

5.3 Self-Training (ST)

The labels in the self-training method are generally the prediction results from the previous stage. In other words, the prediction results from the previous stage can be used as labels to guide the training in the next stage, thus achieving self-training in such an iterative process.

Multi-stage Self-training [106]. This pretext task is proposed to leverage the abundant unlabeled nodes to help training. Given both the labeled set $\mathcal{D}_L^t$ and unlabeled set $\mathcal{D}_U^t$ in the iteration step t, the graph encoder $f_\theta(\cdot)$ is first trained on the labeled set $\mathcal{D}_L^t$, as follows

$$\mathcal{L}_{node}\left(\theta, \mathbf{A}, \mathbf{X}, \mathcal{D}_L^t\right) = \sum_{\left(v_i, y_i^t\right) \in \mathcal{D}_L^t} \ell\Big(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, y_i^t\Big) \tag{103}$$

and then applied to make predictions $\widehat{\mathcal{Y}}^t = \{\widehat{y}_i^t \mid v_i \in \mathcal{V}_U^t\}$ on the unlabeled set $\mathcal{V}_U^t$. Then the predicted labels (as well as corresponding nodes) with K-top high confidence

$$\mathcal{D}_N^t = \Big\{(v_i, \widehat{y}_i^t) | \mid \widehat{y}_i^t \text{ in top-K confidence of } \widehat{\mathcal{Y}}^t\Big\} \tag{104}$$

are considered as the pseudo-labels and moved to the labeled node set $\mathcal{D}_L^t$ to obtain a updated labeled set $\mathcal{D}_L^{t+1} = \mathcal{D}_L^t \bigcup \mathcal{D}_N^t$ and a updated unlabeled set $\mathcal{D}_U^{t+1} = \mathcal{D}_U^t / \mathcal{D}_N^t$. Finally, a fresh graph encoder is trained on the updated labeled set $\mathcal{D}_L^{t+1}$, and the above operations are performed multiple times in an iterative manner.

Node Clustering or Partitioning [31]. Compared to Multi-stage Self-training, the pretext task of Node Clustering pre-assigns a pseudo-label y_i, e.g., the cluster index, to each node v_i by some unsupervised clustering algorithms. The learning objective of this pretext task is then formulated as a classification problem as follows

$$\mathcal{L}_{self}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \ell\Big(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, y_i\Big) \tag{105}$$

When node attributes are not available, another choice to obtain node pseudo-labels is based on the topology of a given graph structure or adjacency matrix. Specifically, graph partitioning [107, 108] is to partition the nodes of a graph into roughly equal subsets, such that the number of edges connecting nodes across subsets is minimized.

To absorb the advantages of both attributive- and structural-based clustering, **CAGNN** [109] combines the node clustering and node partitioning to proposed a new pretext task. Concretely, it first assigns cluster indices as pseudo labels but follow with a topology refining process that refines the clusters by minimizing the inter-cluster edges. Specifically, for an edge $e_{i,j}$, we remove it if the probability that v_i and v_j fall into the same cluster is less than a threshold r , i.e. $\widehat{\mathbf{y}}_i \widehat{\mathbf{y}}_j < r$ (where $\widehat{\mathbf{y}}_i = f_\theta(\mathbf{A}, \mathbf{X})_{v_i}$), otherwise the edge $e_{i,j}$ will be preserved in the graph.

M3S [110]. Combining Multi-stage Self-training with Node Clustering, M3S applies the DeepCluster [100] and the aligning mechanism as a self-checking mechanism and hence provides stronger self-supervision. Specifically, a K-mean clustering algorithm is performed on node representations $\mathbf{H}_t$ learned in the iteration step t (rather than $\mathbf{X}$) and the clustered pseudo-label $\mathcal{D}_N^t$ that matches the prediction of the classifier in the last iteration step $t-1$ will added to the labeled set to obtain an updated labeled set $\mathcal{D}_L^{t+1} = \mathcal{D}_L^t \bigcup \mathcal{D}_N^t$. Finally, a fresh model is trained on the labeled set $\mathcal{D}_L^{t+1}$ with the objective defined as Equ. 103.

Cluster Preserving [36]. An important characteristic of real-world graphs is the cluster structure, so we can consider the cluster preservation as a self-supvised pretext task. The unsupervised clustering algorithms are first applied to group nodes in a graph into K non-overlapping clusters $\{C_k\}_{k=1}^K$, then the cluster prototypes can be obtained by $c_k = \text{AGGRATE}(\{f_\theta(\mathbf{A}, \mathbf{X})_{v_i} \mid v_i \in C_k\})$. The mapping function $g_\omega(\cdot)$ is used to estimate the similarity of node v_i

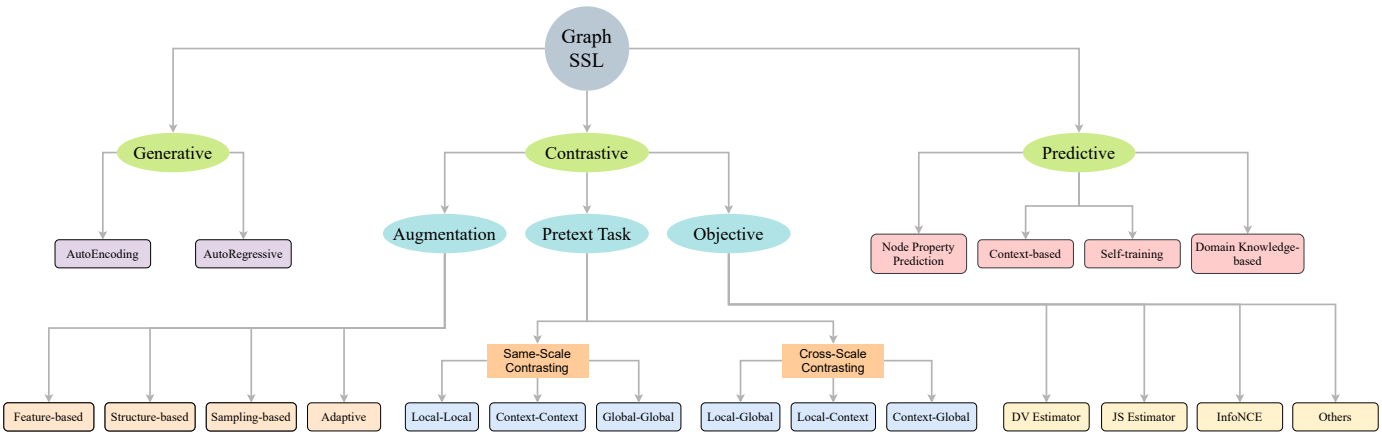


Fig. 6. An summary of graph self-supervised learning (SSL) methods. We categorize them into three branches: contrastive, generative, and predictive. For contrastive methods, they contrast different views and deal with data-data pairs (inter-data) information, and we further categorize it from three aspects: augmentation strategy, pretext task, and objective. In terms of augmentation strategy, it can be divided into four major categories: feature-based augmentation, structure-based augmentation, sampling-based augmentation, and adaptive augmentation. From the perspective of pretext tasks, it can be divided into same-scale contrasting and cross-scale contrasting. The same-scale contrasting includes Local-Local (L-L), Context-Cotext (C-C), and Global-Global (G-G) methods, while the cross-scale contrasting includes the Local-Global (L-G), Local-Cotext (L-C), and Context-Global (C-G) methods. For generative methods, it focuses on the intra-data information and can be divided into Autoencoding and Autoregressive methods. For predictive methods, it handles the data-label relationship, which can be further divided into four major categories: Node Property Prediction (NP), Content-based Prediction (CP), Self-Training (ST), and Domain Knowledge-based Prediction (DK).

with the cluster prototype c_k, e.g., the probability $\widehat{y}_{i,k}$ that node v_i belongs to cluster C_k is defined as follows,

$$\widehat{y}_{i,k} = \frac{\exp\Big(g_\omega(f_\theta(\mathbf{A},\mathbf{X})_{v_i}, c_k)\Big)}{\sum_{k=1}^{K}\exp(g_\omega\Big(f_\theta(\mathbf{A},\mathbf{X})_{v_i}, c_k)\Big)} \tag{106}$$

Finally, we optimize this pretext task as follow

$$\mathcal{L}_{self}\left(\theta,\mathbf{A},\mathbf{X}\right) = -\frac{1}{|\mathcal{V}|}\sum_{v_i\in\mathcal{V}}\sum_{k=1}^{K} y_{i,k}\log(\widehat{y}_{i,k}) \tag{107}$$

where the ground-truth label $y_{i,k} = 1$ if node v_i belongs to cluster C_k, otherwise $y_{i,k} = 0$.

5.4 Domain Knowledge-based Prediction (DK)

The formation of real-world graphs usually obeys specific rules, e.g., the links between atoms in molecular graphs are bounded by valence bonding theory, while cross-cited papers in citation networks generally have the same topic or authors. Therefore, extensive expert knowledge can be incorporated as a *prior* into the design of pretext tasks. Specifically, we can use expert knowledge or specialized tools to analyze graph data to obtain informative labels that can directly be used as self-supervised signals.

Contextual Molecular Property Prediction [111] incorporates domain knowledge about biological macromolecules to design domain (molecule)-specific pretext tasks. For a given node v_i, it samples its k-hop neighborhood nodes and edges as a local subgraph and then extracts statistical properties of this subgraph. Specifically, it counts the number of occurrence of (node, edge) pairs around the center node v_i and then list all the node-edge count terms in alphabetical order, which constitutes the final property: e.g., p_k = C_N-DOUBLE1_O-SINGLE1 in Fig. 5(d). With a large number of context-aware properties $\mathcal{P} = \{p_k\}_{k=1}^{K}$ pre-defined, the contextual property prediction task can be defined as a multi-class prediction problem with one class (label) corresponds to one contextual property, as follow

$$\mathcal{L}_{self}\left(\theta,\mathbf{A},\mathbf{X}\right) = \frac{1}{|\mathcal{V}|}\sum_{v_i\in\mathcal{V}}\ell\Big(f_\theta(\mathbf{A},\mathbf{X})_{v_i}, y_i\Big) \tag{108}$$

where $\ell(\cdot)$ deontes the cross entropy loss, and $y_i = k$ if the molecular property of node v_i is p_k.

Graph-level Motif Prediction [111]. Motifs are recurrent sub-graphs among the input graph data, which are prevalent in molecular graph data. One important class of motifs in molecules are functional groups, which encodes the rich domain knowledge of molecules and can be easily detected by professional softwares, such as RDKit. Suppose we are considering the presence of K motifs $\mathcal{M} = \{m_k\}_{k=1}^{K}$ in the molecular data. For one specific molecule graph $g_i \in \mathcal{G}$, it detects whether each of the motif shows up in g_i, then use it as the labels $\mathbf{y}_i \in \mathbb{R}^K$ where if m_k shows up in g_i, the k-th elements $\mathbf{y}_{i,k}$ will be set to 1, otherwise 0. Formally, the learning objective of the motif prediction task is formulated as a multi-label classification problem, as follows

$$\mathcal{L}_{self}\left(\gamma,\mathcal{G}\right) = \frac{1}{|\mathcal{G}|}\sum_{g_i\in\mathcal{G}}\ell\Big(f_\gamma(\mathbf{A}_i,\mathbf{X}_i), \mathbf{y}_i\Big) \tag{109}$$

where $\ell(\cdot)$ deontes the binary cross entropy loss.

5.5 Additional Methods

In this subsection, we will continue to introduce some predictive methods for graph SSL, but due to space limitations, we will not provide their detailed mathematical formulas.

DrRepair [113] consider the problem of learning to repair programs from diagnostic feedback by building a program feedback graph, which connects symbols relevant to program repair in source code and diagnostic feedback.

TABLE 2
A summary of the surveyed papers. The classification is organized based on the underlying motivation behind pretext task design.

Methods	Graph Property	Pretext Task	Data Augmentation	Objective Function	Training Strategy	Year
Graph Completion [31]	Attributed	Generative/AE	Attribute Masking	MAE	P&F/JL	2020
Node Attribute Masking [29]	Attributed	Generative/AE	Attribute Masking	MAE	P&F/JL	2020
Edge Attribute Masking [30]	Attributed	Generative/AE	Attribute Masking	MAE	P&F	2019
Node Attribute and Embedding Denoising [32]	Attributed	Generative/AE	Attribute Masking	MAE	JL	2020
Adjacency Matrix Reconstruction [33]	Attributed	Generative/AE	Attribute Masking Edge Perturbation	MAE/CE	JL	2020
Graph Bert [34]	Attributed	Generative/AE	Attribute Masking Edge Perturbation	MAE	P&F	2020
Pretrain-Recsys [88]	Attributed	Generative/AE	Edge Perturbation	MAE	P&F	2021
GPT-GNN [49]	Heterogeneous	Generative/AR	Attribute Masking Edge Perturbation	MAE/InfoNCE	P&F	2020
GraphCL [42]	Attributed	Contrastive/G-G	Attribute Masking Edge Perturbation Random Walk Sampling	InfoNCE	URL	2020
IGSD [50]	Attributed	Contrastive/G-G	Edge Perturbation Edge Doffisopm	InfoNCE	JL/URL	2020
DACL [64]	Attributed	Contrastive/G-G	Mixup	InfoNCE	URL	2020
LCC [65]	Attributed	Contrastive/G-G	None	InfoNCE	JL	2021
CSSL [51]	Attributed	Contrastive/G-G	Node Insertion Edge Perturbation Uniform Sampling	InfoNCE	P&F/JL/URL	2020
GCC [55]	Unattributed	Contrastive/C-C	Random Walk Sampling	InfoNCE	P&F/URL	2020
GRACE [43]	Attributed	Contrastive/L-L	Attribute Masking Edge Perturbation	InfoNCE	URL	2020
GCA [57]	Attributed	Contrastive/L-L	Attention-based	InfoNCE	URL	2020
GROC [60]	Attributed	Contrastive/L-L	Gradient-based	InfoNCE	URL	2021
STDGI [45]	Spatial-Temporal	Contrastive/L-L	Attribute Shuffling	JS Estimator	URL	2019
GMI [35]	Attributed	Contrastive/L-L	None	SP Estimator	URL	2020
KS2L [66]	Attributed	Contrastive/L-L	None	InfoNCE	URL	2020
CG^3 [67]	Attributed	Contrastive/L-L	None	InfoNCE	JL	2020
BGRL [44]	Attributed	Contrastive/L-L	Attribute Masking Edge Perturbation	Inner Product	URL	2021
SelfGNN [52]	Attributed	Contrastive/L-L	Attribute Masking Edge Diffusion	MSE	URL	2021
PT-DGNN [68]	Dynamic	Contrastive/L-L	Attribute Masking Edge Perturbation	InforNCE	P&F	2021
COAD [23]	Attributed	Contrastive/L-L	None	Triplet Loss	P&F	2020
Contrst-Reg [46]	Attributed	Contrastive/L-L	Attribute Shuffling	InfoNCE	JL	2021
DGI [28]	Attributed	Contrastive/L-G	Arbitrary	JS Estimator	URL	2019
HDMI [47]	Attributed	Contrastive/L-G	Attribute Shuffling	JS Estimator	URL	2021
DMGI [69]	Heterogeneous	Contrastive/L-G	Attribute Shuffling	JS Estimator MAE	URL	2020
MVGRL [37]	Attributed	Contrastive/L-G	Attribute Masking Edge Perturbation Edge Diffusion Random Walk Sampling	DV Estimator JS Estimator NT-Xent InfoNCE	URL	2020
HDGI [48]	Heterogeneous	Contrastive/L-G	Attribute Shuffling	JS Estimator	URL	2019
Subg-Con [38]	Attributed	Contrastive/L-C	Importance Sampling	Triplet Margin	URL	2020
Cotext Prediction [30]	Attributed	Contrastive/L-C	Ego-nets Sampling	CE	P&F	2019
GIC [70]	Attributed	Contrastive/L-C	Arbitrary	JS Estimator	URL	2020
GraphLoG [76]	Attributed	Contrastive/L-L Contrastive/G-G Contrastive/L-C	Attribute Masking	InfoNCE	URL	2021
MHCN [78]	Heterogeneous	Contrastive/L-C	Attribute Shuffling	InfoNCE	JL	2021
EGI [53]	Attributed	Contrastive/L-C	Ego-nets Sampling	SP Estimator	P&F	2020
MICRO-Graph [56]	Attributed	Contrastive/C-G	Knowledge Sampling	InfoNCE	URL	2020
InfoGraph [71]	Attributed	Contrastive/C-G	None	SP Estimator	URL	2019
SUGAR [72]	Attributed	Contrastive/C-G	BFS Sampling	JS Estimator	JL	2021
BiGI [54]	Heterogeneous	Contrastive/C-G	Edge Perturbation Ego-nets Sampling	JS Estimator	JL	2021
HTC [73]	Attributed	Contrastive/C-G	Attribute Shuffling	SP Estimator DV Estimator	URL	2021
Node Property Prediction [29]	Attributed	Predictive/NP	None	MAE	P&F/JL	2020
S^2GRL [91]	Attributed	Predictive/CP	None	CE	URL	2020
PairwiseDistance [29]	Attributed	Predictive/CP	None	CE	P&F/JL	2020

TABLE 3
A summary of the surveyed papers. The classification is organized based on the underlying motivation behind pretext task design.

Methods	Graph Property	Pretext Task	Data Augmentation	Objective Function	Training Strategy	Year
PairwiseAttrSim [29]	Attributed	Predictive/CP	None	MAE	P&F/JL	2020
Distance2Cluster [29]	Attributed	Predictive/CP	None	MAE	P&F/JL	2020
EdgeMask [29]	Attributed	Predictive/CP	None	CE	P&F/JL	2020
TopoTER [101]	Attributed	Predictive/CP	Edge Perturbation	CE	URL	2021
Centrality Score Ranking [36]	Attributed	Predictive/CP	None	CE	P&F	2019
Meta-path prediction [102]	Heterogeneous	Predictive/CP	None	CE	JL	2020
SLiCE [103]	Heterogeneous	Predictive/CP	None	CE	P&F	2020
Distance2Labeled [29]	Attributed	Predictive/CP	None	MAE	P&F/JL	2020
ContextLabel [29]	Attributed	Predictive/CP	None	MAE	P&F/JL	2020
HCM [112]	Attributed	Predictive/CP	Edge Perturbation	Bayesian inference	URL	2021
Contextual Molecular Property Prediction [111]	Attributed	Predictive/DK	None	CE	P&F	2020
Graph-level Motif Prediction [111]	Attributed	Predictive/DK	None	CE	P&F	2020
Multi-stage Self-training [106]	Attributed	Predictive/ST	None	None	JL	2018
Node Clustering [31]	Attributed	Predictive/ST	None	Clustering	P&F/JL	2020
Graph Partitioning [31]	Attributed	Predictive/ST	None	Partitioning	P&F/JL	2020
CAGAN [109]	Attributed	Predictive/ST	None	Clustering	URL	2020
M3S [110]	Attributed	Predictive/ST	None	Clustering	JL	2020
Cluster Preserving [36]	Attributed	Predictive/ST	None	Clustering/CE	P&F	2019

Besides, it leverages unlabeled programs available online to create a large amount of *extra program repair examples*, which are used to pre-train the model. Motivated by the observation that anomalous nodes differ from normal nodes in structures and attributes, **Hop-Count based Model (HCM)** propose to use *global context prediction* as a self-supervised task for modeling both local and global contextual information, and then achieve better anomaly detection on attributed networks. **SEF** [114] presents a framework that first trains a GAT model with *pseudo labels* obtained from unsupervised Louvain clustering [107] and then uses the learned edge attention coefficients as self-supervised *edge features*. Besides, it encodes the learned edge features via a Set Transformer and combines them with node features for node classification in an end-to-end training manner.

6 Summary of the Implementation

A summary of all the surveyed works is presented in Fig. 6, and Table 2 and Table 3 list their properties, including graph property, pretext task type, data augmentation strategy, objective function, training strategy, and the year of publication. Furthermore, we show in Table 4 and Table 5 implementation details of the surveyed works, such as the (node/link/graph) level of downstream tasks, evaluation metrics with specific tasks, and the commonly used datasets.

6.1 Downstream Tasks

The graph SSL methods are generally evaluated on three levels of graph tasks: node-level, link-level, and graph-level. Among them, the **graph-level tasks** are usually performed on multiple graphs in the form of inductive learning, where the attributes and structures of those test samples are not available during the training process. Commonly used graph-level tasks include graph classification and graph regression. The **link-level tasks** mainly focus on link prediction, that is, given two nodes, the objective is to predict whether a link (edge) exists between them. On the other hand, the **node-level tasks** are generally performed on a large graph in the form of transductive learning. Depending on whether labels are provided, it can be divided into three categories: node regression, node classification, and node clustering. The node regression tasks are usually adopted as downstream tasks for evaluation on the spatial-temporal graph, and the node classification tasks are performed with partial known labels. Instead, the node clustering tasks are performed in a more challenging unsupervised manner and adopted as downstream tasks when the performance of node classification is not distinguishable enough.

6.2 Evaluation Metrics

For **graph classification** tasks, the commonly used evaluation metrics include ROC-AUC and Accuracy (Acc); while for **graph regression** tasks, the Mean Absolute Error (MAE) is used. In terms of **link prediction** tasks, the ROC-AP, ROC-PR, and ROC-AUC are usually used as the evaluation metrics, of which ROC-AUC is the most commonly used and most fairly evaluate the effectiveness of the compared methods. Besides, the **node regression** tasks are usually used for spatial-temporal graphs, and the adopted evaluation metrics include MAE, Mean Square Error (MSE), and Mean Absolute Percentage Error (MAPE). In addition to Accuracy, the **node classification** tasks often adopt F1-score for the single-label classification problem and Micro-F1 (or Macro-F1) for the multi-label classification problem. Moreover, the **node clustering** tasks adopt some metrics used to evaluate the performance of unsupervised clusterings, such as Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), Accuracy, Micro-F1, and Macro-F1, etc.

6.3 Datasets

The statistics of common datasets are shown in Table 6, including category, graph number, node number *per graph*,

TABLE 4
A summary of implementation details about the surveyed papers, including downstream task level, evaluation metric, and dataset.

Methods	Task Level	Evaluation Metric	Dataset
Graph Completion [31]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed
Node Attribute Masking [29]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Reddit
Edge Attribute Masking [30]	Graph	Graph Classification (ROC-AUC)	MUTAG, PTC, PPI, BBBP, Tox21, ToxCast, ClinTox, MUV, HIV, SIDER, BACE
Node Attribute and Embedding Denoising [32]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed
Adjacency Matrix Reconstruction [33]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed
Graph Bert [34]	Node	Node Classification (Acc), Node Clustering (NMI)	Cora, Citeseer, Pubmed
Pretrain-Recsys [88]	Node/Link	-	ML-1M, MOOCs and Last-FM
GPT-GNN [49]	Node/Link	Node Classification (F1-score), Link Prediction (ROC-AUC)	OAG, Amazon, Reddit
GraphCL [42]	Graph	Graph Classification (Acc, ROC-AUC)	NCI1, PROTEINS, D&D, COLLAB, RDT-B, RDT-M5K, GITHUB, MNIST, CIFAR10, MUTAG, IMDB-B, BBBP, Tox21, ToxCast, SIDER, ClinTox, MUV, HIV, BACE, PPI
IGSD [50]	Graph	Graph Classification (Acc)	MUTAG, PTC_MR, NCI1, IMDB-B, QM9, COLLAB, IMDB-M
DACL [64]	Graph	Graph Classification (Acc)	MUTAG, PTC_MR, IMDB-B, IMDB-M, RDT-B, RDT-M5K
LCC [65]	Graph	Graph Classification (Acc)	IMDB-B, IMDB-M, COLLAB, MUTAG, PROTEINS, PTC, NCI1, D&D
CSSL [51]	Graph	Graph Classification (Acc)	PROTEINS, D&D, NCI1, NCI109, Mutagenicity
GCC [55]	Node/Graph	Node Classification (Acc), Graph Classification (Acc)	US-Airport, H-index, COLLAB, IMDB-B, IMDB-M, RDT-B, RDT-M5K
GRACE [43]	Node	Node Classification (Acc, Micro-F1)	Cora, Citeseer, Pubmed, DBLP, Reddit, PPI
GCA [57]	Node	Node Classification (Acc)	Wiki-CS, Amazon-Computers, Amazon-Photo, Coauthor-CS, Coauthor-Physics
GROC [60]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Amazon-Photo, Wiki-CS
STDGI [45]	Node	Node Regression (MAE, RMSE, MAPE)	METR-LA
GMI [35]	Node/Link	Node Classification (Acc, Micro-F1), Link Prediction (ROC-AUC)	Cora, Citeseer, PubMed, Reddit, PPI, BlogCatalog, Flickr
KS2L [66]	Node/Link	Node Classification (Acc), Link Prediction (ROC-AUC)	Cora, Citeseer, Pubmed, Amazon-Computers, Amazon-Photo, Coauthor-CS
CG^3 [67]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Amazon-Computers, Amazon-Photo, Coauthor-CS
BGRL [44]	Node	Node Classification (Acc, Micro-F1)	Wiki-CS, Amazon-Computers, Amazon-Photo, PPI, Coauthor-CS, Coauthor-Physics, ogbn-arxiv
SelfGNN [52]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Amazon-Computers, Amazon-Photo, Coauthor-CS, Coauthor-Physics
PT-DGNN [68]	Link	Link Prediction (ROC-AUC)	HepPh, Math Overflow, Super User
COAD [23]	Node/Link	Node Clustering (Precision, Recall, F1-score), Link Prediction (HitRatio@K, MRR)	AMiner, News, LinkedIn
Contrast-Reg [46]	Node/Link	Node Classification (Acc), Node Clustering (NMI, Acc, Macro-F1), Link Prediction (ROC-AUC)	Cora, Citeseer, Pubmed, Reddit, ogbn-arxiv, Wikipedia, ogbn-products, Amazo-Computers, Amazo-Photo
DGI [28]	Node	Node Classification (Acc, Micro-F1)	Cora, Citeseer, Pubmed, Reddit, PPI
HDMI [47]	Node	Node Classification (Micro-F1, Macro-F1), Node Clustering (NMI)	ACM, IMDB, DBLP, Amazon
DMGI [69]	Node	Node Clustering (NMI), Node Classification (Acc)	ACM, IMDB, DBLP, Amazon
MVGRL [37]	Node/Graph	Node Classification (Acc), Node Clustering (NMI, ARI), Graph Classification (Acc)	Cora, Citeseer, Pubmed, MUTAG, PTC_MR, IMDB-B, IMDB-M, RDT-B
HDGI [48]	Node	Node Classification (Micro-F1, Macro-F1), Node Clustering (NMI, ARI)	ACM, DBLP, IMDB
Subg-Con [38]	Node	Node Classification (Acc, Micro-F1)	Cora, Citeseer, Pubmed, PPI, Flickr, Reddit
Cotext Prediction [30]	Graph	Graph Classification (ROC-AUC)	MUTAG, PTC, PPI, BBBP, Tox21, ToxCast, ClinTox, MUV, HIV, SIDER, BACE

TABLE 5
A summary of implementation details about the surveyed papers, including downstream task level, evaluation metric, and dataset.

Methods	Task Level	Evaluation Metric	Dataset
GIC [70]	Node/Link	Node Classification (Acc), Node Clustering (Acc, NMI, ARI), Link Prediction (ROC-AUC, ROC-AP)	Cora, Citeseer, Pubmed, Amazon-Computers, Amazon-Photo, Coauthor-CS, Coauthor-Physics
GraphLoG [76]	Graph	Graph Classification (ROC-AUC)	BBBP, Tox21, ToxCast, ClinTox, MUV, HIV, SIDER, BACE
MHCN [78]	Node/Link	-	Last-FM, Douban, Yelp
EGI [53]	Node/Link	Node Classification (Acc), Link Prediction (ROC-AUC, MRR)	YAGO, Airport
MICRO-Graph [56]	Graph	Graph Classification (ROC-AUC)	BBBP, Tox21, ToxCast, ClinTox, HIV, SIDER, BACE
InfoGraph [71]	Graph	Graph Classification (Acc)	MUTAG, PTC_MR, RDT-B, RDT-M5K, IMDB-B, QM9, IMDB-M
SUGAR [72]	Graph	Graph Classification (Acc)	MUTAG, PTC, PROTEINS, D&D, NCI1, NCI109
BiGI [54]	Link	Link Prediction (AUC-ROC, AUC-PR)	DBLP, ML-100K, ML-1M, Wikipedia
HTC [73]	Graph	Graph Classification (Acc)	MUTAG, PTC_MR, IMDB-B, IMDB-M, RDT-B, QM9, RDT-M5K
Node Property Prediction [29]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Reddit
S^2GRL [91]	Node/Link	Node Classification (Acc, Micro-F1), Node Clustering (NMI), Link Prediction (ROC-AUC)	Cora, Citeseer, Pubmed, PPI, Flickr, BlogCatalog, Reddit
PairwiseDistance [29]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Reddit
PairwiseAttrSim [29]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Reddit
Distance2Cluster [29]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Reddit
EdgeMask [29]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Reddit
TopoTER [101]	Node/Graph	Node Classification (Acc), Graph Classification (Acc)	Cora, Citeseer, Pubmed, MUTAG, PTC-MR, RDT-B, RDT-M5K, IMDB-B, IMDB-M
Centrality Score Ranking [36]	Node/Link/Graph	Node Classification (Micro-F1), Link Prediction (Micro-F1), Graph Classification (Micro-F1)	Cora, Pubmed, ML-100K, ML-1M, IMDB-M, IMDB-B
Meta-path prediction [102]	Node/Link	Node Classification (F1-score), Link Prediction (ROC-AUC)	ACM, IMDB, Last-FM, Book-Crossing
SLiCE [103]	Link	Link Prediction (ROC-AUC, Micro-F1)	Amazon, DBLP, Freebase, Twitter, Healthcare
Distance2Labeled [29]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Reddit
ContextLabel [29]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed, Reddit
HCM [112]	Node	Node Classification (ROC-AUC)	ACM, Amazon, Enron, BlogCatalog, Flickr
Contextual Molecular Property Prediction [111]	Graph	Graph Classification (Acc), Graph Regression (MAE)	BBBP, SIDER, ClinTox, BACE, Tox21, ToxCast, ESOL, FreeSolv, Lipo, QM7, QM8
Graph-level Motif Prediction [111]	Graph	Graph Classification (Acc), Graph Regression (MAE)	BBBP, SIDER, ClinTox, BACE, Tox21, ToxCast, ESOL, FreeSolv, Lipo, QM7, QM8
Multi-stage Self-training [106]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed
Node Clustering [31]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed
Graph Partitioning [31]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed
CAGAN [109]	Node	Node Classfication (Micro-F1, Macro-F1), Node Clustering (Micro-F1, Macro-F1, NMI)	Cora, Citeseer, Pubmed
M3S [110]	Node	Node Classification (Acc)	Cora, Citeseer, Pubmed
Cluster Preserving [36]	Node/Link/Graph	Node Classification (Micro-F1), Link Prediction (Micro-F1), Graph Classification (Micro-F1)	Cora, Pubmed, ML-100K, ML-1M, IMDB-M, IMDB-B

edge number *per graph*, dimensionality of node attributes, class number and citation papers. Commonly used datasets for graph self-supervised learning tasks can be divided into five categories: citation networks, social networks, protein networks, molecule graphs, and others.

Citation Networks. In citation networks, nodes usually denote papers, node attributes are some keywords in papers, edges denote cross-citation connections, and categories are topics of papers. Note that nodes in the citation networks may also sometimes indicate authors, institutions, etc. The commonly used citation network datasets include Cora [105], Citeseer [115], Pubmed [116], Wiki-CS [117], Coauthor-CS [118], etc.

Social Networks. The social network graph dataset considers entities (e.g. users or authors) as nodes, their interests and hobbies as attributes, and their social interactions as edges. The widely used social network datasets for self-supervised learning are mainly some classical graph

TABLE 6
A summary and statistics of common graph datasets for graph self-supervised learning.

Dataset	Category	#Graph	#Node (Avg.)	#Edge (Avg.)	#Feature	#Class	Citation
Cora [105]	Citation Network	1	2708	5429	1433	7	[31], [29], [32], [33], [34], [43], [60], [35], [67], [28], [37], [38], [70], [91], [36], [106], [109], [110], [46], [52], [66], [101]
Citeseer [115]	Citation Network	1	3327	4732	3703	6	[31], [29], [32], [33], [34], [43], [60], [35], [67], [28], [37], [38], [70], [91], [106], [109], [110], [46], [52], [66], [101]
Pubmed [116]	Citation Network	1	19717	44338	500	3	[31], [29], [32], [33], [34], [43], [60], [35], [67], [28], [37], [38], [70], [91], [36], [106], [109], [110], [46], [52], [66], [101]
Wiki-CS [117]	Citation Network	1	11701	216123	300	10	[57], [60], [44]
Coauthor-CS [118]	Citation Network	1	18333	81894	6805	15	[57], [67], [70], [44], [52], [66]
Coauthor-Physics [118]	Citation Network	1	34493	247962	8415	5	[57], [70], [44], [52]
DBLP (v12)	Citation Network	1	4894081	45564149	-	-	[43], [69], [103], [54], [48], [47]
ogbn-arxiv [119]	Citation Network	1	169343	1166243	128	40	[44], [46]
Reddit [3]	Social Network	1	232965	11606919	602	41	[29], [49], [43], [35], [28], [38], [91], [46]
BlogCatalog [120]	Social Network	1	5196	171743	8189	6	[35], [91], [112]
Flickr [120]	Social Network	1	7575	239738	12047	9	[35], [38], [91], [112]
COLLAB [121]	Social Networks	5000	74.49	2457.78	-	2	[42], [50], [65], [55]
RDT-B [121]	Social Networks	2000	429.63	497.75	-	2	[42], [55], [37], [71], [73], [101], [64]
RDT-M5K [121]	Social Networks	4999	508.52	594.87	-	5	[42], [55], [71], [73], [101], [64]
IMDB-B [121]	Social Networks	1000	19.77	96.53	-	2	[42], [50], [65], [55], [37], [71], [36], [69], [102], [48], [73], [47], [101], [64]
IMDB-M [121]	Social Networks	1500	13.00	65.94	-	3	[50], [65], [55], [37], [71], [36], [69], [102], [48], [73], [47], [101], [64]
ML-100K [122]	Social Networks	1	2625	100000	-	5	[36], [54]
ML-1M [122]	Social Networks	1	9940	1000209	-	5	[36], [54], [88]
PPI [123]	Protein Networks	24	56944	818716	50	121	[30], [42], [43], [35], [35], [28], [38], [91], [44]
D&D [124, 125]	Protein Networks	1178	284.32	715.65	82	2	[42], [65], [51], [72]
PROTEINS [124, 126]	Protein Networks	1113	39.06	72.81	4	2	[42], [65], [51], [72]
NCI1 [121, 127]	Molecule Graphs	4110	29.87	32.30	37	2	[42], [50], [65], [51], [72]
MUTAG [128, 129]	Molecule Graphs	188	17.93	19.79	7	2	[30], [42], [50], [65], [37], [71], [72], [73], [101], [64]
PTC [125]	Molecule Graphs	344	25.50	-	19	2	[30], [65], [72]
QM9 (QM7, QM8) [130]	Molecule Graphs	133885	-	-	-	-	[50], [71], [29], [73]
BBBP [119, 131]	Molecule Graphs	2039	24.05	25.94	-	2	[30], [42], [56], [111], [76]
Tox21 [119, 132, 133]	Molecule Graphs	7831	18.51	25.94	-	12	[30], [42], [56], [111], [76]
ToxCast [119, 134]	Molecule Graphs	8575	18.78	19.26	-	167	[30], [42], [56], [111], [76]
ClinTox [135]	Molecule Graphs	1478	26.13	27.86	-	2	[30], [42], [56], [111], [76]
MUV [136]	Molecule Graphs	93087	24.23	26.28	-	17	[30], [42], [76]
HIV [119]	Molecule Graphs	41127	25.53	27.48	-	2	[30], [42], [56], [76]
SIDER [137]	Molecule Graphs	1427	33.64	35.36	-	27	[30], [42], [56], [111], [76]
BACE [138]	Molecule Graphs	1513	34.12	36.89	-	2	[30], [42], [56], [111], [76]
PTC_MR [125]	Molecule Graphs	344	14.29	14.69	-	2	[50], [37], [71], [73], [101], [64]
NCI109 [121, 127]	Molecule Graphs	4127	29.68	32.13	-	2	[51], [72]
Mutagenicity [139, 140]	Molecule Graphs	4337	30.32	30.77	-	2	[51]
MNIST [141]	Others (Image)	-	70000	-	784	10	[42]
CIFAR10 [142]	Others (Image)	-	60000	-	1024	10	[42]
METR-LA [143]	Others (Traffic)	1	207	1515	2	-	[45]
Amazon-Computers [118]	Others (Purchase)	1	13752	245861	767	10	[57], [67], [70], [49], [103], [44], [112], [47], [46], [52], [66]
Amazon-Photo [118]	Others (Purchase)	1	7650	119081	745	8	[57], [60], [67], [70], [49], [103], [44], [112], [47], [46], [52], [66]
ogbn-products [119]	Others (Purchase)	1	2449029	61859140	100	47	[46]

datasets, such as Reddit [3], COLLAB [121], RDT-B [121], and IMDB-B [121], etc.

Molecule Graphs. In molecular graphs, nodes represent atoms in the molecule, the atom index is indicated by the node attributes and edges represent bonds. Molecular graph datasets typically contain multiple graphs and are commonly used for tasks such as graph classification and graph regression, e.g. predicting molecular properties. The commonly used molecule graph datasets include NC1 [121, 127], MUTAG [128, 129], BBBP [119, 131], MUV [136],

and HIV [119], etc.

Protein Networks. The protein networks can be divided into two main categories - Protein Molecule Graph and Protein Interaction Network - based on the way they are modeled. The Protein Molecule Graph is a particular type of molecule graph, where nodes represent amino acids and an edge indicates the two connected nodes are less than 6 angstroms apart. The commonly used datasets include PROTEINS [124, 126] and D&D [124, 125], used for chemical molecular property prediction. The other branch is Protein Interaction Networks, where nodes denote protein molecules and edges indicate their interactions. The commonly used dataset is PPI [123], used for graph biological function prediction.

Other Graphs. In addition to the four types of datasets mentioned above, some other datasets are less common or difficult to categorize, such as image datasets (MNIST [141], CIFAR10 [142]), traffic datasets (METR-LA [143]), and co-purchase datasets (Amazon-Computers [118], Amazon-Photo [118]).

6.4 Codes in Github

The open-source codes are beneficial to the development of the deep learning community. A summary of the open-source codes of the surveyed works is presented in Table 7, where we provide hyperlinks to their open-source codes, and those for which no open-source code is found are indicated by "N.A.". Most of these methods are implemented on GPUs based on Pytorch or Tensorflow libraries. Moreover, we have created a GitHub repository https://github.com/LirongWu/awesome-graph-self-supervised-learning to summarize the latest advances in graph SSL, which will be updated in real-time as more papers and their codes become available.

7 Discussion

Despite recent successes in CV and NLP domains, applying SSL to graph data is still very challenging. In this section, we analyze the existing technical challenges for graph SSL and point out some promising directions for future works.

7.1 Pretext Tasks for Complex Types of Graph

Most current works on graph SSL focus on attribute graphs, with little effort to other more complex and challenging graph types, such as spatial-temporal and heterogeneous graphs. Most of the designed pretext tasks utilize only node-level or graph-level features, limiting their ability to exploit the richer information embedded in the graph, such as temporal information in spatial-temporal graphs and relation information in heterogeneous graphs. A promising direction is to design *graph type-specific* pretext tasks that adaptively pick the most suitable tasks depending on the type of graph.

7.2 Lack of Theoretical Foundation

Despite the great success of graph SSL on various tasks, they mostly draw on the successful experience of SSL on CV and NLP domains. In other words, most existing graph SSL methods are designed with intuition, and their performance gains are evaluated by empirical experiments. The lack of sufficient theoretical foundations behind the design has led to both performance bottlenecks and poor explainability. Therefore, we believe that building a solid theoretical foundation for graph SSL from a graph theory perspective and minimizing the gap between the theoretical foundation and empirical design is also a promising future direction.

7.3 Insufficient Augmentation Strategy

Recent advances in the field of visual representation learning largely owing to a variety of data augmentation strategies [12, 13], such as resize, rotation, coloring, etc. However, due to the inherent Non-Euclidean nature of graph data, it is difficult to directly apply existing CNN-based data augmentation to graph data. Moreover, most current data augmentation strategies on graphs are limited to adding/removing nodes and edges or their combination to achieve the asserted SOTA. To further improve the performance of SSL on graphs, it is a promising direction to design more efficient augmentation strategies or to perform augmentation adaptively for specific downstream tasks. Furthermore, the selection of high-quality negative samples from the augmentation is also a crucial issue. For example, [144] has developed a family of unsupervised sampling strategies for user-controllable negative sample selection.

7.4 Lack of Explanability

Though existing graph SSL methods have achieved excellent results on various downstream tasks, we still do not know exactly what has been learned from self-supervised tasks. Which of the feature patterns, significant structures, or feature-structure relationships has been learned by self-supervision? Is this learning explicit or implicit? Is it possible to find interpretable correspondences on the input data? These are important issues for understanding and interpreting model behavior but are missing in current graph SSL works. Therefore, we need to explore the interpretability of graph SSL and perform a deep analysis of model behavior to improve the generalization and robustness of existing methods for security or privacy-related downstream tasks.

7.5 Margin from Pre-training to Downstream Tasks

Pre-training with self-supervised pretext tasks and then using the pre-trained model for a specific downstream task, either by fine-tuning or freezing the weights, is a common training strategy in graph SSL [36, 68, 88, 90]. However, how shall we transfer the pre-trained knowledge to downstream tasks? Though a large number of strategies have been proposed to address this problem in the field of CV and NLP [145–147], they are difficult to be directly applied to the GNN domains due to the inherent non-Euclidean structure of graph data. Therefore, it is an important issue to design graph-specific techniques to minimize the margin between pre-training and downstream tasks.

8 Conclusion

In this survey, we provide a comprehensive overview of SSL techniques for graph data. We divide existing graph

TABLE 7
A summary of open-source codes of the surveyed papers.

Methods	Github
Graph Completion [31]	https://github.com/Shen-Lab/SS-GCNs
Node Attribute Masking [29]	https://github.com/ChandlerBang/SelfTask-GNN
Edge Attribute Masking [30]	http://snap.stanford.edu/gnn-pretrain
Node Attribute and Embedding Denoising [32]	N.A.
Adjacency Matrix Reconstruction [33]	N.A.
Graph Bert [34]	https://github.com/anonymous-sourcecode/Graph-Bert
Pretrain-Recsys [88]	https://github.com/jerryhao66/Pretrain-Recsys
GPT-GNN [49]	https://github.com/acbull/GPT-GNN
GraphCL [42]	https://github.com/Shen-Lab/GraphCL
IGSD [50]	N.A.
DACL [64]	N.A.
LCC [65]	https://github.com/YuxiangRen/Label-Contrastive-Coding-based-Graph-Neural-Network-for-Graph-Classification-
CSSL [51]	N.A.
GCC [55]	https://github.com/THUDM/GCC
GRACE [43]	https://github.com/CRIPAC-DIG/GRACE
GCA [57]	https://github.com/CRIPAC-DIG/GCA
GROC [60]	N.A.
STDGI [45]	N.A.
GMI [35]	https://github.com/zpeng27/GMI
KS2L [66]	N.A.
CG^3 [67]	N.A.
BGRL [44]	N.A.
SelfGNN [52]	https://github.com/zekarias-tilahun/SelfGNN
PT-DGNN [68]	https://github.com/Mobzhang/PT-DGNN
COAD [23]	https://github.com/allanchen95/Expert-Linking
Contrast-Reg [46]	N.A.
DGI [28]	https://github.com/PetarV-/DGI
HDMI [47]	N.A.
DMGI [69]	https://github.com/pcy1302/DMGI
MVGRL [37]	https://github.com/kavehhassani/mvgrl
HIGI [48]	https://github.com/YuxiangRen/Heterogeneous-Deep-Graph-Infomax
Subg-Con [38]	https://github.com/yzjiao/Subg-Con
Cotext Prediction [30]	http://snap.stanford.edu/gnn-pretrain
GIC [70]	https://github.com/cmavro/Graph-InfoClust-GIC
GraphLoG [76]	https://openreview.net/forum?id=DAaaaqPv9-q
MHCN [78]	https://github.com/Coder-Yu/RecQ
EGI [53]	https://openreview.net/forum?id=J_pvI6ap5Mn
MICRO-Graph [56]	https://drive.google.com/file/d/1b751rpnV-SDmUJvKZZI-AvpfEa9eHxo9/
InfoGraph [71]	https://github.com/fanyun-sun/InfoGraph
SUGAR [72]	https://github.com/RingBDStack/SUGAR
BiGI [54]	https://github.com/clhchtcjj/BiNE
HTC [73]	N.A.
Node Property Prediction [29]	https://github.com/ChandlerBang/SelfTask-GNN
S^2GRL [91]	N.A.
PairwiseDistance [29]	https://github.com/ChandlerBang/SelfTask-GNN
PairwiseAttrSim [29]	https://github.com/ChandlerBang/SelfTask-GNN
Distance2Cluster [29]	https://github.com/ChandlerBang/SelfTask-GNN
EdgeMask [29]	https://github.com/ChandlerBang/SelfTask-GNN
TopoTER [101]	N.A.
Centrality Score Ranking [36]	N.A.
Meta-path prediction [102]	https://github.com/mlvlab/SELAR
SLiCE [103]	https://github.com/pnnl/SLICE
Distance2Labeled [29]	https://github.com/ChandlerBang/SelfTask-GNN
ContextLabel [29]	https://github.com/ChandlerBang/SelfTask-GNN
HCM [112]	N.A.
Contextual Molecular Property Prediction [111]	https://github.com/tencent-ailab/grover
Graph-level Motif Prediction [111]	https://github.com/tencent-ailab/grover
Multi-stage Self-training [106]	https://github.com/Davidham3/deeper_insights_into_GCNs
Node Clustering [31]	https://github.com/Shen-Lab/SS-GCNs
Graph Partitioning [31]	https://github.com/Shen-Lab/SS-GCNs
CAGAN [109]	N.A.
M3S [110]	https://github.com/datake/M3S
Cluster Preserving [36]	N.A.

SSL methods into three categories: contrastive, generative, and predictive. Specifically, we investigate and summarize the core mathematical ideas of recent research in graph SSL within a unified framework. Moreover, we summarize the commonly used datasets, evaluation metrics, downstream tasks, and open-source implementations of the surveyed works, setting the stage for the development of future methods. Finally, we point out the technical limitations of current research and provide promising directions for future works on graph self-supervised learning.

ACKNOWLEDGMENT

We acknowledge the support received from the National Natural Science Foundation of China. In addition, Lirong would especially like to thank DaHa, whose company has given him powerful spiritual support over the past years.

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[3] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017.

[4] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[5] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[7] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[8] M. Liu, Z. Wang, and S. Ji, "Non-local graph neural networks," *arXiv preprint arXiv:2005.14612*, 2020.

[9] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 338–348.

[10] D. Kim and A. Oh, "How to find your friendly neighborhood: Graph attention design with self-supervision," in *International Conference on Learning Representations*, 2021.

[11] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *arXiv preprint arXiv:2006.08218*, vol. 1, no. 2, 2020.

[12] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[13] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[14] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *arXiv preprint arXiv:2006.09882*, 2020.

[15] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," *arXiv preprint arXiv:2006.07733*, 2020.

[16] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," *arXiv preprint arXiv:2103.03230*, 2021.

[17] X. Chen and K. He, "Exploring simple siamese representation learning," *arXiv preprint arXiv:2011.10566*, 2020.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] 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, p. 9, 2019.

[20] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[21] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[22] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[23] B. Chen, J. Zhang, X. Zhang, X. Tang, L. Cai, H. Chen, C. Li, P. Zhang, and J. Tang, "Coad: Contrastive pre-training with adversarial fine-tuning for zero-shot expert linking," *arXiv preprint arXiv:2012.11336*, 2020.

[24] Y. Xie, Z. Wang, and S. Ji, "Noise2same: Optimizing a self-supervised bound for image denoising," *arXiv preprint arXiv:2010.11971*, 2020.

[25] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void-learning denoising from single noisy images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2129–2137.

[26] J. Batson and L. Royer, "Noise2self: Blind denoising by self-supervision," in *International Conference on Machine Learning*. PMLR, 2019, pp. 524–533.

[27] L. Galke, I. Vagliano, and A. Scherp, "Can graph neural networks go" online"? an analysis of pretraining and inference," *arXiv preprint arXiv:1905.06018*, 2019.

[28] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." in *ICLR*

(Poster), 2019.

[29] W. Jin, T. Derr, H. Liu, Y. Wang, S. Wang, Z. Liu, and J. Tang, "Self-supervised learning on graphs: Deep insights and new direction," *arXiv preprint arXiv:2006.10141*, 2020.

[30] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*, 2019.

[31] Y. You, T. Chen, Z. Wang, and Y. Shen, "When does self-supervision help graph convolutional networks?" in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 871–10 880.

[32] F. Manessi and A. Rozza, "Graph-based neural network models with multiple self-supervised auxiliary tasks," *arXiv preprint arXiv:2011.07267*, 2020.

[33] Q. Zhu, B. Du, and P. Yan, "Self-supervised training of graph convolutional networks," *arXiv preprint arXiv:2006.02380*, 2020.

[34] J. Zhang, H. Zhang, C. Xia, and L. Sun, "Graph-bert: Only attention is needed for learning graph representations," *arXiv preprint arXiv:2001.05140*, 2020.

[35] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *Proceedings of The Web Conference 2020*, 2020, pp. 259–270.

[36] Z. Hu, C. Fan, T. Chen, K.-W. Chang, and Y. Sun, "Pre-training graph neural networks for generic structural feature extraction," *arXiv preprint arXiv:1905.13728*, 2019.

[37] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4116–4126.

[38] Y. Jiao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu, "Sub-graph contrast for scalable self-supervised graph representation learning," *arXiv preprint arXiv:2009.10273*, 2020.

[39] Y. Xie, Z. Xu, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *arXiv preprint arXiv:2102.10757*, 2021.

[40] Y. Liu, S. Pan, M. Jin, C. Zhou, F. Xia, and P. S. Yu, "Graph self-supervised learning: A survey," *arXiv preprint arXiv:2103.00111*, 2021.

[41] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.

[42] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[43] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," *arXiv preprint arXiv:2006.04131*, 2020.

[44] S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Veličković, and M. Valko, "Bootstrapped representation learning on graphs," *arXiv preprint arXiv:2102.06514*, 2021.

[45] F. L. Opolka, A. Solomon, C. Cangea, P. Veličković, P. Liò, and R. D. Hjelm, "Spatio-temporal deep graph infomax," *arXiv preprint arXiv:1904.06316*, 2019.

[46] K. Ma, H. Yang, H. Yang, T. Jin, P. Chen, Y. Chen, B. F. Kamhoua, and J. Cheng, "Improving graph representation learning by contrastive regularization," *arXiv preprint arXiv:2101.11525*, 2021.

[47] B. Jing, C. Park, and H. Tong, "Hdmi: High-order deep multiplex infomax," *arXiv preprint arXiv:2102.07810*, 2021.

[48] Y. Ren, B. Liu, C. Huang, P. Dai, L. Bo, and J. Zhang, "Heterogeneous deep graph infomax," *arXiv preprint arXiv:1911.08538*, 2019.

[49] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "Gpt-gnn: Generative pre-training of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1857–1867.

[50] H. Zhang, S. Lin, W. Liu, P. Zhou, J. Tang, X. Liang, and E. P. Xing, "Iterative graph self-distillation," *arXiv preprint arXiv:2010.12609*, 2020.

[51] J. Zeng and P. Xie, "Contrastive self-supervised learning for graph classification," *arXiv preprint arXiv:2009.05923*, 2020.

[52] Z. T. Kefato and S. Girdzijauskas, "Self-supervised graph neural networks without explicit negative sampling," *arXiv preprint arXiv:2103.14958*, 2021.

[53] Q. Zhu, Y. Xu, H. Wang, C. Zhang, J. Han, and C. Yang, "Transfer learning of graph neural networks with ego-graph information maximization," *arXiv preprint arXiv:2009.05204*, 2020.

[54] J. Cao, X. Lin, S. Guo, L. Liu, T. Liu, and B. Wang, "Bipartite graph embedding via mutual information maximization," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 635–643.

[55] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1150–1160.

[56] S. Zhang, Z. Hu, A. Subramonian, and Y. Sun, "Motif-driven contrastive learning of graph representations," *arXiv preprint arXiv:2012.12533*, 2020.

[57] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," *arXiv preprint arXiv:2010.14945*, 2020.

[58] P. Bonacich, "Power and centrality: A family of measures," *American journal of sociology*, vol. 92, no. 5, pp. 1170–1182, 1987.

[59] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[60] N. Jovanović, Z. Meng, L. Faber, and R. Wattenhofer, "Towards robust graph contrastive learning," *arXiv preprint arXiv:2102.13085*, 2021.

[61] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[62] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *Proceedings of the 2017 ACM on Conference on*

Information and Knowledge Management, 2017, pp. 889–898.

[63] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6519–6528.

[64] V. Verma, M.-T. Luong, K. Kawaguchi, H. Pham, and Q. V. Le, "Towards domain-agnostic contrastive learning," *arXiv preprint arXiv:2011.04419*, 2020.

[65] Y. Ren, J. Bai, and J. Zhang, "Label contrastive coding based graph neural network for graph classification," *arXiv preprint arXiv:2101.05486*, 2021.

[66] L. Yu, S. Pei, C. Zhang, L. Ding, J. Zhou, L. Li, and X. Zhang, "Self-supervised smoothing graph neural networks," *arXiv preprint arXiv:2009.00934*, 2020.

[67] S. Wan, S. Pan, J. Yang, and C. Gong, "Contrastive and generative graph convolutional networks for graph-based semi-supervised learning," *arXiv preprint arXiv:2009.07111*, 2020.

[68] J. Zhang, K. Chen, and Y. Wang, "Pre-training on dynamic graph neural networks," *arXiv preprint arXiv:2102.12380*, 2021.

[69] C. Park, D. Kim, J. Han, and H. Yu, "Unsupervised attributed multiplex network embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5371–5378.

[70] C. Mavromatis and G. Karypis, "Graph infoclust: Leveraging cluster-level node information for unsupervised graph representation learning," *arXiv preprint arXiv:2009.06946*, 2020.

[71] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," *arXiv preprint arXiv:1908.01000*, 2019.

[72] Q. Sun, H. Peng, J. Li, J. Wu, Y. Ning, P. S. Yu, and L. He, "Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism," *arXiv preprint arXiv:2101.08170*, 2021.

[73] C. Wang and Z. Liu, "Graph representation learning by ensemble aggregating subgraphs via mutual information maximization," *arXiv preprint arXiv:2103.13125*, 2021.

[74] S. Cheng, L. Zhang, B. Jin, Q. Zhang, and X. Lu, "Drug target prediction using graph representation learning via substructures contrast," 2021.

[75] T. Kipf, E. van der Pol, and M. Welling, "Contrastive learning of structured world models," *arXiv preprint arXiv:1911.12247*, 2019.

[76] M. Xu, H. Wang, B. Ni, H. Guo, and J. Tang, "Self-supervised graph-level representation learning with local and global structure," 2021. [Online]. Available: https://openreview.net/forum?id=DAaaaqPv9-q

[77] L. Xu, A. Krzyzak, and E. Oja, "Rival penalized competitive learning for clustering analysis, rbf net, and curve detection," *IEEE Transactions on Neural networks*, vol. 4, no. 4, pp. 636–649, 1993.

[78] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang, "Self-supervised multi-channel hypergraph convolutional network for social recommendation," *arXiv preprint arXiv:2101.06448*, 2021.

[79] M. D. Donsker and S. S. Varadhan, "Asymptotic evaluation of certain markov process expectations for large time. iv," *Communications on Pure and Applied Mathematics*, vol. 36, no. 2, pp. 183–212, 1983.

[80] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *International Conference on Machine Learning*. PMLR, 2018, pp. 531–540.

[81] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," *arXiv preprint arXiv:1606.00709*, 2016.

[82] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.

[83] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[84] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 1857–1865.

[85] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[86] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[87] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[88] B. Hao, J. Zhang, H. Yin, C. Li, and H. Chen, "Pre-training graph neural networks for cold-start users and items representation," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 265–273.

[89] B. Fatemi, L. E. Asri, and S. M. Kazemi, "Slaps: Self-supervision improves structure learning for graph neural networks," *arXiv preprint arXiv:2102.05034*, 2021.

[90] J. Shang, T. Ma, C. Xiao, and J. Sun, "Pre-training of graph augmented transformers for medication recommendation," *arXiv preprint arXiv:1906.00346*, 2019.

[91] Z. Peng, Y. Dong, M. Luo, X.-M. Wu, and Q. Zheng, "Self-supervised graph representation learning via global context prediction," *arXiv preprint arXiv:2003.01604*, 2020.

[92] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[93] Z. Gao, H. Lin, S. Li *et al.*, "Clustering based on graph of density topology," *arXiv preprint arXiv:2009.11612*, 2020.

[94] L. Wu, Z. Liu, Z. Zang, J. Xia, S. Li, S. Li *et al.*, "Deep clustering and representation learning that preserves

geometric structures," *arXiv preprint arXiv:2009.09590*, 2020.

[95] S. Z. Li, L. Wu, and Z. Zang, "Consistent representation learning for high dimensional data analysis," *arXiv preprint arXiv:2012.00481*, 2020.

[96] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4066–4075.

[97] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

[98] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[99] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, and I. Craddock, "N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding," *arXiv preprint arXiv:1908.05968*, 2019.

[100] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 132–149.

[101] X. Gao, W. Hu, and G.-J. Qi, "Topo{ter}: Unsupervised learning of topology transformation equivariant representations," 2021. [Online]. Available: https://openreview.net/forum?id=9az9VKjOx00

[102] 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," *arXiv preprint arXiv:2007.08294*, 2020.

[103] P. Wang, K. Agarwal, C. Ham, S. Choudhury, and C. K. Reddy, "Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks," *arXiv preprint arXiv:2007.11192*, 2020.

[104] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[105] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[106] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[107] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[108] V. A. Traag, L. Waltman, and N. J. Van Eck, "From louvain to leiden: guaranteeing well-connected communities," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[109] Y. Zhu, Y. Xu, F. Yu, S. Wu, and L. Wang, "Cagnn: Cluster-aware graph neural networks for unsupervised graph representation learning," *arXiv preprint arXiv:2009.01674*, 2020.

[110] K. Sun, Z. Lin, and Z. Zhu, "Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5892–5899.

[111] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang, "Self-supervised graph transformer on large-scale molecular data," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[112] T. Huang, Y. Pei, V. Menkovski, and M. Pechenizkiy, "Hop-count based self-supervised anomaly detection on attributed networks," *arXiv preprint arXiv:2104.07917*, 2021.

[113] M. Yasunaga and P. Liang, "Graph-based, self-supervised program repair from diagnostic feedback," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 799–10 808.

[114] A. Sehanobish, N. G. Ravindra, and D. van Dijk, "Self-supervised edge features for improved graph neural network training," *arXiv preprint arXiv:2007.04777*, 2020.

[115] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries*, 1998, pp. 89–98.

[116] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[117] P. Mernyei and C. Cangea, "Wiki-cs: A wikipedia-based benchmark for graph neural networks," *arXiv preprint arXiv:2007.02901*, 2020.

[118] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.

[119] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[120] J. Li, X. Hu, J. Tang, and H. Liu, "Unsupervised streaming feature selection in social media," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1041–1050.

[121] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.

[122] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[123] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, 2017.

[124] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *Journal of molecular biology*, vol. 330, no. 4, pp. 771–783, 2003.

[125] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[126] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.

[127] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 347–375, 2008.

[128] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[129] N. Kriege and P. Mutzel, "Subgraph matching kernels for attributed graphs," *arXiv preprint arXiv:1206.6483*, 2012.

[130] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific data*, vol. 1, no. 1, pp. 1–7, 2014.

[131] I. F. Martins, A. L. Teixeira, L. Pinheiro, and A. O. Falcao, "A bayesian approach to in silico blood-brain barrier penetration modeling," *Journal of chemical information and modeling*, vol. 52, no. 6, pp. 1686–1697, 2012.

[132] A. Mayr, G. Klambauer, T. Unterthiner, and S. Hochreiter, "Deeptox: toxicity prediction using deep learning," *Frontiers in Environmental Science*, vol. 3, p. 80, 2016.

[133] R. Huang, M. Xia, D.-T. Nguyen, T. Zhao, S. Sakamuru, J. Zhao, S. A. Shahane, A. Rossoshek, and A. Simeonov, "Tox21challenge to build predictive models of nuclear receptor and stress response pathways as mediated by exposure to environmental chemicals and drugs," *Frontiers in Environmental Science*, vol. 3, p. 85, 2016.

[134] A. M. Richard, R. S. Judson, K. A. Houck, C. M. Grulke, P. Volarath, I. Thillainadarajah, C. Yang, J. Rathman, M. T. Martin, J. F. Wambaugh *et al.*, "Toxcast chemical landscape: paving the road to 21st century toxicology," *Chemical research in toxicology*, vol. 29, no. 8, pp. 1225–1251, 2016.

[135] P. A. Novick, O. F. Ortiz, J. Poelman, A. Y. Abdulhay, and V. S. Pande, "Sweetlead: an in silico database of approved drugs, regulated chemicals, and herbal isolates for computer-aided drug discovery," *PloS one*, vol. 8, no. 11, p. e79568, 2013.

[136] E. J. Gardiner, J. D. Holliday, C. O'Dowd, and P. Willett, "Effectiveness of 2d fingerprints for scaffold hopping," *Future medicinal chemistry*, vol. 3, no. 4, pp. 405–414, 2011.

[137] M. Kuhn, I. Letunic, L. J. Jensen, and P. Bork, "The sider database of drugs and side effects," *Nucleic acids research*, vol. 44, no. D1, pp. D1075–D1079, 2016.

[138] G. Subramanian, B. Ramsundar, V. Pande, and R. A. Denny, "Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches," *Journal of chemical information and modeling*, vol. 56, no. 10, pp. 1936–1949, 2016.

[139] K. Riesen and H. Bunke, "Iam graph database repository for graph based pattern recognition and machine learning," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2008, pp. 287–297.

[140] J. Kazius, R. McGuire, and R. Bursi, "Derivation and validation of toxicophores for mutagenicity prediction," *Journal of medicinal chemistry*, vol. 48, no. 1, pp. 312–320, 2005.

[141] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[142] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[143] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi, "Big data and its technical challenges," *Communications of the ACM*, vol. 57, no. 7, pp. 86–94, 2014.

[144] J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, "Contrastive learning with hard negative samples," *arXiv preprint arXiv:2010.04592*, 2020.

[145] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[146] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[147] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.