This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3131584, IEEE Transactions on Knowledge and Data Engineering

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

Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, 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 existing SSL techniques for graph data. Specifically, we divide existing graph SSL methods into three categories: contrastive, generative, and predictive. More importantly, unlike other surveys that only provide a high-level description of published research, we present an additional mathematical summary of 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, open-source implementations, and experimental study 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, due to its ability to capture both graph structures and node/edge features. However, most of the 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 methods are hardly applicable to real-world scenarios, especially those requiring expert knowledge for annotation, such as medicine, meteorology, transportation, etc. More importantly, these methods are prone to suffer from problems of over-fitting, poor generalization, and weak robustness [1].

Recent advances in SSL [2–7] have provided novel insights into reducing the dependency on annotated labels and enable the training on massive unlabeled data. The primary goal of SSL is to learn transferable 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) [2–4] and natural language processing (NLP) [5–7]. For example, Moco [2] and SimCLR [3] augment image data by various means and then train Convolutional Neural Networks (CNNs) to capture dependencies between different augmentations. Besides, BERT [5] 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 domains, it is a promising direction to apply SSL to the graph domain to fully exploit graph structure information and rich unlabeled data.

Compared with image and language sequence data, applying SSL to graph data is very important and has promising research prospects. Firstly, Firstly, along with node features, graph data also contains structure, where extensive pretext tasks can be designed to capture the intrinsic relations of nodes. Secondly, real-world graphs are usually formed by 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 data generally supports transductive learning [8], such as node classification, where 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.

Though some works have been proposed recently to apply SSL to graph data and have achieved remarkable success [9–19], 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 for image and text data, nodes in the graph are correlated with each other rather than completely independent. This requires 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 ob-

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

Manuscript received April 19, 2005; revised August 26, 2015.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3131584, IEEE Transactions on Knowledge and Data Engineering

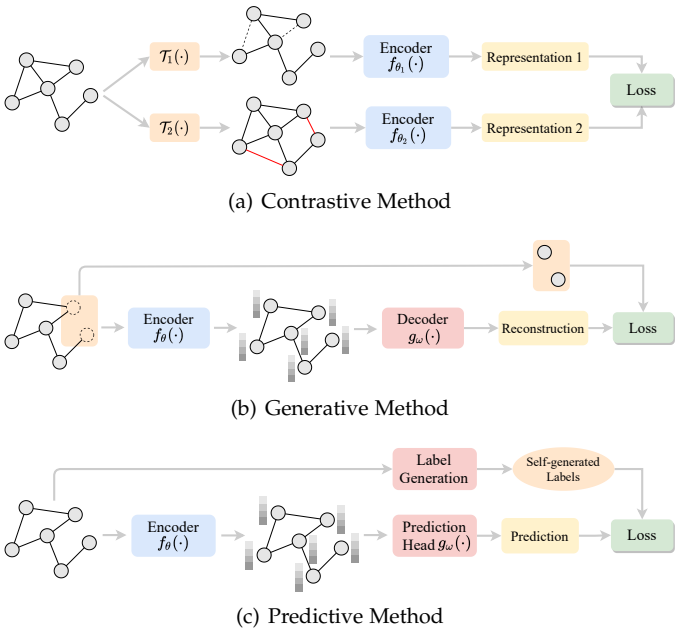


Fig. 1. A comparison among the contrastive, generative, and predictive method. **(a)**: the contrastive method contrasts the views generated from different augmentation $\mathcal{T}_1(\cdot)$ and $\mathcal{T}_2(\cdot)$. The information about the differences and sameness between (inter-data) data-data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

jectives 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. 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 their clarity and accessibility.
- Summarize the core mathematical ideas of recent research in graph SSL within a unified framework.
- Summarize the commonly used datasets, evaluation metrics, downstream tasks, open-source codes, and experimental study of surveyed 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 [1], 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, [20] 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 [21] 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 [20, 21], our advantages are as follow: (1) more mathematical details, striving to summarize each method with one equation; (2) more implementation details, including 41 datasets statistics (*vs* 20 datasets in [20]), evaluation metrics, and open-source code; (3) more thorough experimental study for fair comparison; (4) more fine-grained, clarified and rational taxonomy; (5) more surveyed works, 71 methods (*vs* 47 methods in [21] *vs* 18 methods in [20]); (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 $\mathbf{A}_{i,j} = 1$ if $e_{i,j} \in \mathcal{E}$ and $\mathbf{A}_{i,j} = 0$ 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 a 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 $\mathbf{x}_{i,j}^e \in \mathbb{R}^{b_0}$ 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 can be 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 the time step t and $\mathbf{A}_{i,j}^{(t)} = 1$ denotes an interaction between node v_i and v_j at the 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 $f_v : \mathcal{V} \to \mathcal{Y}^v$ and an edge type mapping function $f_e : \mathcal{E} \to \mathcal{Y}^e$, 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 define 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 $g = (\mathbf{A}, \mathbf{X}^{(t)})$, where $\mathbf{X}^{(t)} \in \mathbb{R}^{N \times d_0}$ is the node feature matrix at the time step t ($1 \leq t \leq T$).

2.2 Downstream Tasks

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

Node-level tasks. Node-level tasks focus on the properties of nodes, and 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 $f_\theta : \mathcal{V} \to \mathcal{Y}$ 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 $\mathcal{L}_{node}$, as follows

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

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

Link-level tasks. Link-level tasks focus on the representation of node paris 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,

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

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

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 $\mathcal{P}_L$ 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 $\mathcal{L}_{graph}$,

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

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, and $\ell(\cdot, \cdot)$ is the mean absolute error.

2.3 Graph Neural Networks

Graph neural networks (GNN) [8, 22, 23] are a family of neural networks that have been widely used as the backbone encoder in most of the reviewed works. A general GNN framework 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

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

where $1 \leq l \leq L$ 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

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

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 [24].

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 $v_i \in \mathcal{V}$
$\mathcal{E}$	The set of edges in graph g
$e_{i,j}$	An edge $e_{i,j} \in \mathcal{E}$ 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
$\mathbf{D}$	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 matrix at the time step t
$\mathbf{x}_{i,j}^e \in \mathbb{R}^{b_0}$	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 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, $\mathbf{H} = \mathbf{H}^{(L)}$
$\mathbf{h}_g \in \mathbb{R}^{d_L}$	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
$f_\theta, f_{\theta_1}, f_{\theta_2}, \cdots$	Node-level encoder to output $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$
$f_\gamma, f_{\gamma_1}, f_{\gamma_2}, \cdots$	Graph-level encoder to output $\mathbf{h}_g = f_\gamma(\mathbf{A}, \mathbf{X})$
$g_\omega, g_{\omega_1}, g_{\omega_2}, \cdots$	The prediction head
$\mathcal{T}, \mathcal{T}_1, \mathcal{T}_2, \cdots$	The data augmentation transformation
$\mathbf{W}, \Theta, \theta, \gamma, \omega$	Learnable model parameters

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3131584, IEEE Transactions on Knowledge and Data Engineering

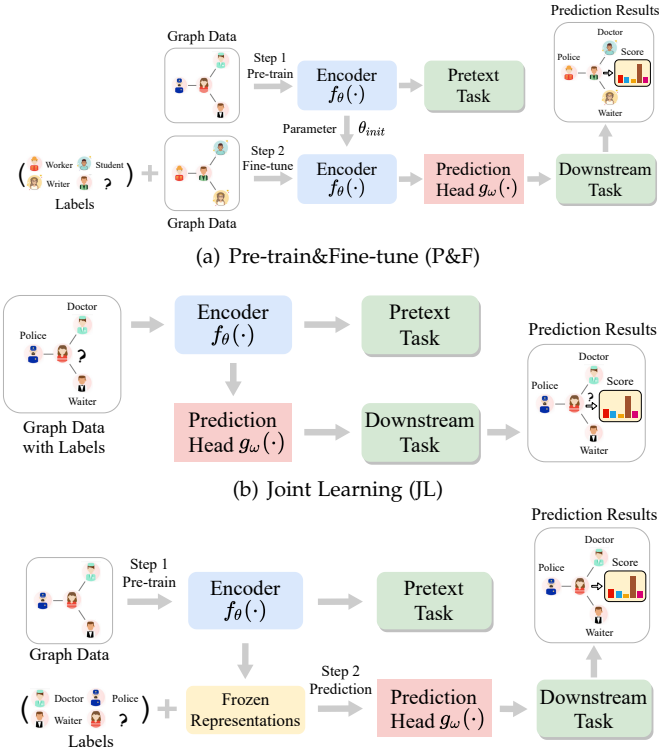


Fig. 2. An overview of training strategies for graph SSL. The training strategies can be divided into three categories. **(a)**: for the Pre-train&Fine-tune strategy, it first pre-trains the encoder $f_\theta(\cdot)$ with unlabeled nodes by the self-supervised pretext tasks. The pre-trained encoder's 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.

2.4 Training Strategy

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.

Pre-training and Fine-tuning (P&F). In this strategy, the model is trained in a two-stage paradigm [10]. 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 head $g_\omega(\cdot)$ under the supervision of specific downstream tasks. The learning objective is formulated as

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

with initialization $\theta_{init} = \arg\min_\theta \mathcal{L}_{ssl}(f_\theta)$, where $\mathcal{L}_{task}$ and $\mathcal{L}_{ssl}$ is the loss function of downstream tasks and self-supervised pretext tasks, respectively.

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

$$\theta^*, \omega^* = \arg\min_{(\theta,\omega)} \mathcal{L}_{task}(f_\theta, g_\omega) + \alpha \arg\min_\theta \mathcal{L}_{ssl}(f_\theta) \tag{7}$$

where α is a trade-off hyperparameter.

Unsupervised Representation Learning. This strategy can also be considered as a two-stage paradigm, with the first stage similar to 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 downstream tasks only. The learning objective is formulated as

$$\omega^* = \arg\min_\omega \mathcal{L}_{task}(f_{\theta_{init}}, g_\omega) \tag{8}$$

with initialization $\theta_{init} = \arg\min_\theta \mathcal{L}_{ssl}(f_\theta)$. Compared to other schemes, unsupervised representation learning is more challenging since the learning of the encoder $f_\theta(\cdot)$ depends only on the pretext task and is frozen in the second stage. In contrast, in the P&F strategy, the encoder $f_\theta(\cdot)$ can be further optimized under the supervision of the downstream task during the fine-tuning stage.

3 Contrastive Learning

3.1 A Unified Perspective

Inspired by the recent advances of contrastive learning in CV and NLP domains, some works have been proposed to apply contrastive learning for graph data. However, most 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 *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 through various 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 against the agreement of two independently sampled negative pairs. The agreement between views is usually measured through Mutual Information (MI) estimation. Given a graph $g = (\mathbf{A}, \mathbf{X})$, 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

$$\mathbf{A}_k, \mathbf{X}_k = \mathcal{T}_k(\mathbf{A}, \mathbf{X}); k = 1, 2, \cdots, K \tag{9}$$

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

$$\mathbf{h}_k = f_{\theta_k}(\mathbf{A}_i, \mathbf{X}_i); k = 1, 2, \cdots, K \tag{10}$$

The contrastive learning aims to maximize the mutual information of two views from the same instance as

$$\max_{\theta_1,\theta_2,\cdots,\theta_K} \sum_i \sum_{j\neq i} \alpha_{i,j} \mathcal{MI}(\mathbf{h}_i, \mathbf{h}_j) \tag{11}$$

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. $\mathcal{MI}(\mathbf{h}_i, \mathbf{h}_j)$ 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 negative samples to contrast with $\{\mathbf{h}_i\}_{i=1}^K$ can be taken as 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 concrete values vary in different schemes.

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 [24].

3.2 Data Augmentation

The recent works in the CV domain show that the success of contrastive 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 graph data, it is difficult to directly apply data augmentations designed for images to graphs. Here, we divide the data augmentation strategy for graph data into 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

$$\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}} = \mathcal{T}(\mathbf{A}, \mathbf{X}) = \mathbf{A}, \mathcal{T}_{\mathbf{X}}(\mathbf{X}) \tag{12}$$

Attribute Masking. The attribute masking [10, 11, 25–27] randomly masks a small portion of attributes. We specify $\mathcal{T}_{\mathbf{X}}(\mathbf{X})$ for the attribute masking as

$$\mathcal{T}_{\mathbf{X}}(\mathbf{X}) = \mathbf{X} \odot (1 - \mathbf{L}) + \mathbf{M} \odot \mathbf{L} \tag{13}$$

where $\mathbf{L}$ is a masking location matrix where $\mathbf{L}_{i,j} = 1$ if the j-th element of node v_i is masked, otherwise $\mathbf{L}_{i,j} = 0$. $\mathbf{M}$ denotes a masking value matrix. The matrix $\mathbf{L}$ is usually sampled by Bernoulli distribution or assigned manually. Besides, different schemes of $\mathbf{M}$ result in different augmentations. 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 [9, 28–31] 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 receive different contextual information. We specify $\mathcal{T}_{\mathbf{X}}(\mathbf{X})$ for the attribute shuffling as

$$\mathcal{T}_{\mathbf{X}}(\mathbf{X}) = \mathbf{X}[idx, :] \tag{14}$$

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 adjacent matrix $\mathbf{A}$, as follows

$$\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}} = \mathcal{T}(\mathbf{A}, \mathbf{X}) = \mathcal{T}_{\mathbf{A}}(\mathbf{A}), \mathbf{X} \tag{15}$$

Edge Perturbation. The edge perturbation [14, 25, 32–34] perturbs structural connectivity through randomly adding or removing a certain ratio of edges. We specify $\mathcal{T}_{\mathbf{A}}(\mathbf{A})$ for the edge perturbation as

$$\mathcal{T}_{\mathbf{A}}(\mathbf{A}) = \mathbf{A} \odot (1 - \mathbf{L}) + (1 - \mathbf{A}) \odot \mathbf{L} \tag{16}$$

where $\mathbf{L}$ is a perturbation location matrix where $\mathbf{L}_{i,j} = \mathbf{L}_{j,i} = 1$ if the edge between node v_i and v_j will be perturbed, otherwise $\mathbf{L}_{i,j} = \mathbf{L}_{j,i} = 0$. 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 [34] 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

$$p(\widetilde{\mathbf{A}}_{i,j} = \widetilde{\mathbf{A}}_{j,i} = 1) = r, p(\widetilde{\mathbf{A}}_{i,j} = \widetilde{\mathbf{A}}_{j,i} = 0) = 1 - r \tag{17}$$

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

Edge Diffusion. The edge diffusion [18, 35] generates a different topological view of the original graph structure, with the general edge diffusion process defined as

$$\mathcal{T}_{\mathbf{A}}(\mathbf{A}) = \sum_{k=0}^{\infty} \Theta_k \mathbf{S}^k \tag{18}$$

where $\mathbf{S} \in \mathbb{R}^{N\times N}$ is the generalized transition matrix and Θ is the weighting coefficient which satisfies $\sum_{k=0}^{\infty} \Theta_k = 1, \Theta_k \in [0, 1]$. Two instantiations of Equ. 18 are: (1) Personalized PageRank (PPR) with $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\Theta_k = \alpha(1 - \alpha)^k$, and (2) Heat Kernel (HK) with $\mathbf{S} = \mathbf{A}\mathbf{D}^{-1}$ and $\Theta_k = e^{-t}t^k/k!$, where α denotes teleport probability in a random walk and t is the diffusion time. The closed-form solutions of PPR and HK diffusion are formulated as

$$\begin{aligned} \mathcal{T}_{\mathbf{A}}^{PPR}(\mathbf{A}) &= \alpha \left(\mathbf{I}_n - (1 - \alpha)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^{-1} \\ \mathcal{T}_{\mathbf{A}}^{HK}(\mathbf{A}) &= \exp\left(t\mathbf{A}\mathbf{D}^{-1} - t\right) \end{aligned} \tag{19}$$

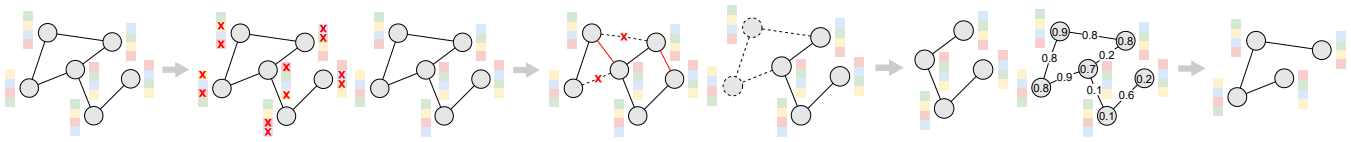
3.2.3 Sampling-based Augmentation

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

$$\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}} = \mathcal{T}(\mathbf{A}, \mathbf{X}) = \mathbf{A}[\mathcal{S}, \mathcal{S}], \mathbf{X}[\mathcal{S}, :] \tag{20}$$

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3131584, IEEE Transactions on Knowledge and Data Engineering



(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.

Uniform Sampling. The uniform sampling [34] (a.k.a Node Dropping) uniformly samples a given number of nodes $\mathcal{S}$ from $\mathcal{V}$ and remove the remaining nodes directly.

Ego-nets Sampling [11, 36, 37]. 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, the transformation $\mathcal{T}(\cdot)$ samples the L-ego net surrounding node v_i, with $\mathcal{S}$ defined as

$$\mathcal{S} = \{v_j \mid d(v_i, v_j) \leq L\} \tag{21}$$

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 [18, 25, 38]. 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, the walk returns back to the starting node v_i with a positive probability α. Finally, the visited nodes are collected into a node subset $\mathcal{S}$.

Importance Sampling [19]. Given a node v_i, we can sample a subgraph based on the importance of its neighboring nodes, with an importance score matrix $\mathbf{M}$ defined as

$$\mathbf{M} = \alpha \cdot (\mathbf{I}_n - (1-\alpha) \cdot \mathbf{A}\mathbf{D}^{-1}) \tag{22}$$

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

Knowledge Sampling [39]. The knowledge-based sampling incorporates domain knowledge into subgraph sampling. For example, the 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 [40] 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

$$p_{i,j} = \min\left(\frac{s_{\max} - s_{i,j}}{s_{\max} - \mu_s} \cdot p_e, p_\tau\right) \tag{23}$$

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 cut-off probability, used to truncate the probabilities since extremely high removal probabilities will overly corrupt graph structures. The node centrality can be defined as degree centrality, Eigenvector centrality [41], or PageRank centrality [42], which results in three variants. The attribute masking based on node importance is the same as above and will not be repeated.

Gradient-based. Unlike the simple uniform edge removal and insertion as in GRACE [26], GROC [43] 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 graph $g = (\mathbf{A}, \mathbf{X})$ to obtain two views, masking node attributes independently with probability r_1 and r_2 and then computing the contrastive loss $\mathcal{L}_{ssl}$ between these two views. For a given node v_i, an edge removal candidate set is defined as

$$\mathcal{S}^- = \left\{(v_i, v_k) \middle| v_k \in \mathcal{N}_i^{(l)}\right\} \tag{24}$$

, and an edge insertation candidate set is defined as

$$\mathcal{S}^+ = \left\{(v_i, v_k) \middle| v_k \in \left(\cup_{v_m \in \mathcal{B}} \mathcal{N}_m^{(l)} \backslash \mathcal{N}_i^{(l)}\right)\right\} \tag{25}$$

where $\mathcal{B} \subset \mathcal{V}$ is a node batch. $\mathcal{S}^+$ 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 $\mathcal{L}_{ssl}$ to obtain gradient intensity values for each edge in $\mathcal{S}^-$ and $\mathcal{S}^+$. A further gradient-based adaptive augmentations are applied on the views by removing a subset of edges with *minimal* edge gradient magnitude values in $\mathcal{S}^-$ and inserting a subset of edges with the *maximal* edge gradient magnitude 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3131584, IEEE Transactions on Knowledge and Data Engineering

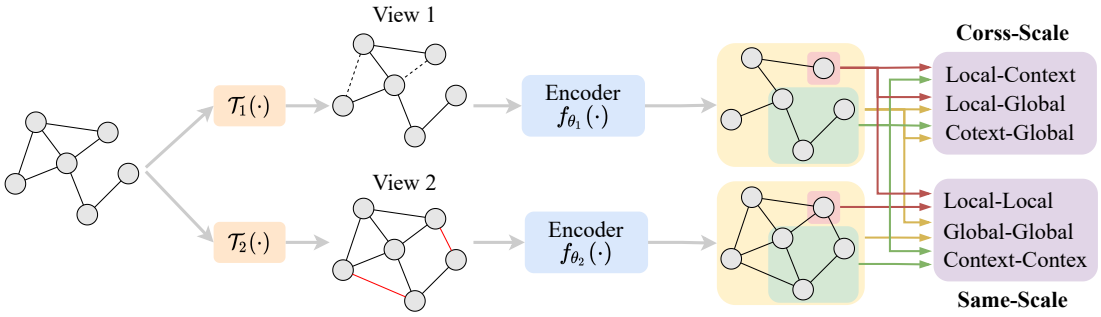


Fig. 4. A general framework for contrastive learning methods with three main modules: data augmentation strategies, pretext tasks, and contrastive objectives. Different views can be generated by a single or a combination of augmentations $\mathcal{T}_1(\cdot)$ and $\mathcal{T}_2(\cdot)$. For graph encoder $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$, the commonly used graph neural networks include GAE [44], VGAE [44], 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 views 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. 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 and graph-graph pairs, while the two views in the cross-scale contrasting have different scales, such as node-subgraph or node-graph contrasting. We categorize existing methods from these two perspectives and present them in a unified framework as shown Fig. 4. In this section, due to space limitations, we present only some representative contrastive methods and place those relatively less important works in **Appendix A**.

3.3.1 Contrasting with the same-scale

The same-scale contrastive learning is further refined into three 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 [25]. 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-level encoder $f_\gamma(\cdot)$ is applied to obtain graph-level representations $\mathbf{h}_{g_i} = f_\gamma(\mathbf{A}_i, \mathbf{X}_i)$ and $\widetilde{\mathbf{h}}_{\widetilde{g}_i} = f_\gamma(\widetilde{\mathbf{A}}_i, \widetilde{\mathbf{X}}_i)$, respectively. Finally, the learning objective is defined as follows

$$\max_{\theta} \frac{1}{|\mathcal{G}|} \sum_{g_i \in \mathcal{G}} \mathcal{MI}\left(\mathbf{h}_{g_i}, \widetilde{\mathbf{h}}_{\widetilde{g}_i}\right) \tag{26}$$

Contrastive Self-supervised Learning (CSSL) [34] follows a very similar 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, removes all edges in S, adds a new node v_i, and adds an edge between v_i and each node in S.

Label Contrastive Coding (LCC) [45] is proposed to encourage intra-class compactness and inter-class separability. To power contrastive learning, LLC 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-level 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

$$\max_{\gamma_q} \log \frac{\sum_{i=1}^m \mathbb{I}_{y_i=y_q} \cdot \exp\left(\mathbf{h}_{g_q} \cdot \mathbf{h}_{g_k}^{(i)}/\tau\right)}{\sum_{i=1}^m \exp\left(\mathbf{h}_{g_q} \cdot \mathbf{h}_{g_k}^{(i)}/\tau\right)} \tag{27}$$

where m is the size of memory bank, τ is the temperature hyperparameter, and $\mathbb{I}_{y_i=y_q}$ 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 y_q. The parameter γ_k of $f_{\gamma_k}(\cdot)$ follows a momentum-based update mechanism as Moco [2], given by

$$\gamma_k \longleftarrow \alpha\gamma_k + (1-\alpha)\gamma_q \tag{28}$$

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) [38] 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 and collect them in to a memory bank $\mathcal{S}$. Then the query subgraph $g_q \in \mathcal{S}$ and key subgraph $g_k \in \mathcal{S}$ are encoded by two graph-level encoders $f_{\gamma_q(\cdot)}$ and $f_{\gamma_k}(\cdot)$ to obtain graph-level representations $\mathbf{h}_{g_q}$ and $\mathbf{h}_{g_k}$, respectively. If g_q and 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3131584, IEEE Transactions on Knowledge and Data Engineering

(g_q, y_q) where y_q is the index of graph it sampled from, its graph contrastive loss is calculated by

$$\max_{\gamma_q} \log \frac{\sum_{i=1}^{|\mathcal{S}|} \mathbb{I}_{y_i=y_q} \cdot \exp\left(\mathbf{h}_{g_q} \cdot \mathbf{h}_{g_k^{(i)}}/\tau\right)}{\sum_{i=1}^{|\mathcal{S}|} \exp\left(\mathbf{h}_{g_q} \cdot \mathbf{h}_{g_k^{(i)}}/\tau\right)} \tag{29}$$

where $\mathbb{I}_{y_i=y_q}$ 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. 28.

3.3.1.3 Local-Local Contrasting

GRACE [26]. Rather than contrasting global-global views as GraphCL [25] and CSSL [34], GRACE focuses on contrasting views at the node level. Given a graph $g = (\mathbf{A}, \mathbf{X})$, it first generates two augmentatd graphs $g^{(1)} = (\mathbf{A}^{(1)}, \mathbf{X}^{(1)}) = \mathcal{T}_1(\mathbf{A}, \mathbf{X})$ and $g^{(2)} = (\mathbf{A}^{(1)}, \mathbf{X}^{(2)}) = \mathcal{T}_2(\mathbf{A}, \mathbf{X})$. Then it applies a shared encoder $f_\theta(\cdot)$ to generate their node embedding matrices $\mathbf{H}^{(1)} = f_\theta(\mathbf{A}^{(1)}, \mathbf{X}^{(1)})$ and $\mathbf{H}^{(2)} = f_\theta(\mathbf{A}^{(2)}, \mathbf{X}^{(2)})$. Finally, the pairwise objective for each positive pair $(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)})$ is defined as follows

$$\mathcal{L}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)}) = \log \frac{e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)})/\tau}}{e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)})/\tau} + Neg} \tag{30}$$

where Neg is defined as

$$Neg = \sum_{k=1}^{N} \mathbf{1}_{k \neq i} \left[e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(1)})/\tau} + e^{\mathcal{D}(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(2)})/\tau} \right] \tag{31}$$

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,

$$\max_{\theta} \frac{1}{2N} \sum_{i=1}^{N} \left[\mathcal{L}(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)}) + \mathcal{L}(\mathbf{h}_i^{(2)}, \mathbf{h}_i^{(1)}) \right] \tag{32}$$

GCA [40] and **GROC** [43] adopt the same framework and objective as GRACE but with more flexible and *adaptive* data augmentation strategies. The framework proposed by SEPT [46] is similar to GRACE, but it is specifically designed for the specific downstream task (recommendation) by combining cross-view contrastive learning with semi-supervised tri-training. Technically, SEPT first augments the user data with the user social information, and then it builds three graph encoders upon the augmented views, with one for recommendation and the other two used to predict unlabeled users. Given a certain user, SEPT takes those nodes whose predicted labels are highly consistent with the target user as positive samples and then encourages the consistency between the target user and positive samples.

Cross-layer Contrasting (GMI) [16]. 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

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{MI}(\mathbf{h}_i, \mathbf{x}_i) \tag{33}$$

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

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{v_j \in \mathcal{N}_i} \mathcal{MI}(\mathbf{w}_{i,j}, \mathbf{A}_{i,j}) \tag{34}$$

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 $Neg(\mathbf{w}_{i,j}) = \{\mathbf{A}_{i,k} \mid v_k \in \mathcal{N}_i \quad \text{and} \quad k \neq j\}$.

STDGI [28] extents the idea of mutual information maximization 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 the 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

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{MI}\left(\mathbf{h}_i^{(t)}, \mathbf{x}_i^{(t+k)}\right) \tag{35}$$

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

BGRL [27]. 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 matrices $\mathbf{H}^{(1)} = f_{\theta_1}(\mathbf{A}^{(1)}, \mathbf{X}^{(1)})$ 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

$$\max_{\theta_1, \omega} \frac{1}{N} \sum_{i=1}^{N} \frac{\mathbf{z}_i^{(1)} (\mathbf{h}_i^{(2)})^T}{\|\mathbf{z}_i^{(1)}\| \|\mathbf{h}_i^{(2)}\|} \tag{36}$$

where the parameter θ_2 are updated as an exponential moving average (EMA) of parameters θ_1, as done in Equ. 28.

SelfGNN [35] differs from BGRL only in the definition of the objective function. Unlike Equ. 36, SelfGNN defines the implicit contrastive term directly in the form of MSE,

$$\min_{\theta_1, \omega} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{z}_i^{(1)} - \mathbf{h}_i^{(2)}\|^2 \tag{37}$$

HeCo [47]. Consider a meta-path Φ_k form the meta-path set $\{\Phi_k\}_{k=1}^K$, if there exist a meta-path Φ_k between node v_i and node v_j, then v_j can be considered as in the meta-path neighborhood $\mathcal{N}_i^{\Phi_k}$ of node v_i, which yields a meta-path based adjacent matrix $\mathbf{A}^{\Phi_k}$. The HeCo first applies two graph encoder $f_{\theta_1}^{sc}(\cdot)$ and $f_{\theta_2}^{mp}(\cdot)$ to obtain node embedding matrices $\mathbf{H}^{sc} = f_{\theta_1}^{sc}(\mathbf{A}, \mathbf{X})$ and $\mathbf{H}^{mp} = f_{\theta_2}^{ml}(\{\mathbf{A}^{\Phi_k}\}_{k=1}^K, \mathbf{X})$. To define positive and negative samples, HeCo first defines a function $\mathbb{C}_i(j) = \sum_{k=1}^K \mathbb{I}\left(j \in \mathcal{N}_i^{\Phi_k}\right)$ to count the number of meta-paths connecting nodes v_i and v_j. Then it constructs a set $\mathcal{S}_i = \{j \mid j \in \mathcal{V} \text{ and } \mathbb{C}_i(j) \neq 0\}$ and sort it in the descending order based on the value of $\mathbb{C}_i(j)$. Next it selects the top T_{pos} nodes from $\mathcal{S}_i$ as positive samples $\mathbb{P}_i$ and treat

the rest as negative samples $\mathbb{N}_i$ directly. Finally, the learning objective can be defined as follows

$$\max_{\theta_1,\theta_2} \frac{1}{N}\sum_{i=1}^{N} \log \frac{\sum_{v_j\in\mathbb{P}_i} e^{\mathcal{D}(\mathbf{h}_i^{sc},\mathbf{h}_j^{mp})/\tau}}{\sum_{v_k\in\{\mathbb{P}_i\cup\mathbb{N}_i\}} e^{\mathcal{D}(\mathbf{h}_i^{sc},\mathbf{h}_k^{mp})/\tau}} \tag{38}$$

3.3.2 Contrasting with the cross-scale

Based on 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) [9] 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 an augmented graph $\widetilde{g} = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}) = \mathcal{T}(\mathbf{A}, \mathbf{X})$. Then it passes these two graphs 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 $\mathbf{H} = f_{\theta_2}(\mathbf{A}, \mathbf{X})$, respectively. Beside, a READOUT function is applied to obtain the graph-level representaion $\widetilde{\mathbf{h}}_{\widetilde{g}} = \text{READOUT}(\widetilde{\mathbf{H}})$. Finally, the learning objective is defined as follows

$$\max_{\theta_1,\theta_2} \frac{1}{N}\sum_{v_i\in\mathcal{V}} \mathcal{MI}\left(\widetilde{\mathbf{h}}_{\widetilde{g}}, \mathbf{h}_i\right) \tag{39}$$

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 $Neg(\widetilde{\mathbf{h}}_{\widetilde{g}}) = \{\mathbf{h}_j\}_{v_j\in\mathcal{V}, j\neq i}$.

MVGRL [18] maximize the the mutual information between the cross-view representations of nodes and graphs. Given a $g = (\mathbf{A}, \mathbf{X}) \in \mathcal{G}$, it first applies an 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 $g^{(2)} = (\mathbf{A}^{(2)}, \mathbf{X}^{(2)}) = \mathcal{T}_2(\mathbf{A}, \mathbf{X})$ from it. Then two graph encoder $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$ and a 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 $\mathbf{H}^{(2)} = g_{\omega_1}(f_{\theta_2}(\mathbf{A}^{(2)}, \mathbf{X}^{(2)}))$. In addition, a READOUT function and another prejection head $g_{\omega_2}(\cdot)$ are use to obtain graph-level representations $\mathbf{h}_g^{(1)} = f_{\omega_2}(\text{READOUT}(\mathbf{H}^{(1)}))$ and $\mathbf{h}_g^{(2)} = f_{\omega_2}(\text{READOUT}(\mathbf{H}^{(2)}))$. The learning objective is defined as follows:

$$\max_{\theta_1,\theta_2,\omega_1,\omega_2} \frac{1}{N}\sum_{v_i\in\mathcal{V}} \left[\mathcal{MI}(\mathbf{h}_g^{(1)}, \mathbf{h}_i^{(2)}) + \mathcal{MI}(\mathbf{h}_g^{(2)}, \mathbf{h}_i^{(1)})\right] \tag{40}$$

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 $Neg(\mathbf{h}_g^{(2)}) = \{\mathbf{h}_j^{(1)}\}_{v_j\in\mathcal{V}, j\neq i}$.

3.3.2.2 Local-Context Contrasting

SUBG-CON [19] 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 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

$$\max_{\theta} \frac{1}{|\mathcal{S}|}\sum_{v_i\in\mathcal{S}} \mathcal{MI}\left(\mathbf{h}_i^{(i)}, \mathbf{h}_{g_i}\right) \tag{41}$$

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 $Neg(\mathbf{h}_i^{(i)}) = \{\mathbf{h}_{g_j}\}_{v_j\in\mathcal{S}, j\neq i}$.

Graph InfoClust (GIC) [48] relies on a framework similar to DGI [9]. However, in addition to contrast local-global views, GIC also maximize the MI between node representations and their corresponding cluster embeddings. Given 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 graph encoder $f_\theta(\cdot)$ is applied to obtain node embedding matrices $\mathbf{H} = f_\theta(\mathbf{A}, \mathbf{X})$ 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 it obtains the cluster centers by $\boldsymbol{\mu}_k = \frac{1}{|C_k|}\sum_{v_i\in C_k} \mathbf{h}_i$ where $1 \leq k \leq K$. To compute the cluster embedding $\mathbf{z}_i$ for each node v_i, it applies a weighted average of the summaries of the cluster centers to which node v_i belongs $\mathbf{z}_i = \sigma\left(\sum_{k=1}^{K} r_{ik}\boldsymbol{\mu}_k\right)$, 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

$$\max_{\theta} \frac{1}{N}\sum_{v_i\in\mathcal{V}} \mathcal{MI}\left(\mathbf{h}_i, \mathbf{z}_i\right) \tag{42}$$

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 [39]. 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 matrices $\{\mathbf{H}_1^{(1)}, \mathbf{H}_2^{(1)}, \cdots, \mathbf{H}_K^{(1)}\}$ 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

$$\max_{\theta} \frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}}\sum_{k=1}^{K} \left[\mathcal{MI}(\mathbf{h}_g^{(1)}, \mathbf{h}_k^{(1)}) + \mathcal{MI}(\mathbf{h}_g^{(2)}, \mathbf{h}_k^{(2)})\right] \tag{43}$$

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 [49] aims to obtain embeddings *at the whole graph level* for self-supervised learning. Given 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 learn 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 the graph-level representation $\mathbf{h}_g = \text{READOUT}(\{\mathbf{h}_i\}_{i=1}^N)$. Finally, the learning objective is defined as follows

$$\max_\theta \sum_{g \in \mathcal{G}} \frac{1}{|g|} \sum_{v_i \in g} \mathcal{MI}\left(\mathbf{h}_g, \mathbf{h}_i\right) \tag{44}$$

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

BiGi [37] is specifically designed for bipartite graph, where the class label $y_i \in \{0, 1\}$ of each node v_i is already known. For a given $g = (\mathbf{A}, \mathbf{X})$, it first applies a 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 the graph-level representation from $\mathbf{H}$ directly as follows

$$\mathbf{h}_g = \Big[\sigma\Big(\frac{1}{|\mathcal{V}^{(1)}|} \sum_{v_i \in \mathcal{V}^{(1)}} \mathbf{h}\Big) \Big\| \sigma\Big(\frac{1}{|\mathcal{V}^{(2)}|} \sum_{v_i \in \mathcal{V}^{(2)}} \mathbf{h}_i\Big)\Big] \tag{45}$$

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 given 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 module (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 $\widetilde{\mathbf{H}}$. Finally, the learning objective is defined as follows:

$$\max_{\theta,\gamma} \frac{1}{|\mathcal{E}|} \sum_{(v_i,v_j) \in \mathcal{E}} \mathcal{MI}\left(\mathbf{h}_g, \mathbf{h}_{i,j}\right) \tag{46}$$

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

3.4 Contrasive Objectives

The main way to optimize the contrastive learning is to treat two representations *(views)* $\mathbf{h}_i$ and $\mathbf{h}_j$ as random variables and maximize their mutual information, given by

$$\mathcal{MI}(\mathbf{h}_i, \mathbf{h}_j) = \mathbb{E}_{p(\mathbf{h}_i, \mathbf{h}_j)} \left[\log \frac{p(\mathbf{h}_i, \mathbf{h}_j)}{p(\mathbf{h}_i) p(\mathbf{h}_j)}\right] \tag{47}$$

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 [50] is one of the *lower-bound* to the mutual information, defined as

$$\begin{aligned} \mathcal{MI}_{DV}\left(\mathbf{h}_i, \mathbf{h}_j\right) =& \mathbb{E}_{p(\mathbf{h}_i, \mathbf{h}_j)} \left[\mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j\right)\right] \\ & - \log \mathbb{E}_{p(\mathbf{h}_i) p(\mathbf{h}_j)} \left[e^{\mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j\right)}\right] \end{aligned} \tag{48}$$

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 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 $\mathcal{D}(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T \mathbf{z}_j / \tau$ with temperature parameter τ, 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 in Equ. 47 with the JS-divergence, it derives another Jensen-Shannon (JS) estimator [51] which can estimate and optimize the mutual information more efficiently. The Jensen-Shannon (JS) estimator is defined as

$$\begin{aligned} \mathcal{MI}_{JS}\left(\mathbf{h}_i, \mathbf{h}_j\right) = & \mathbb{E}_{p(\mathbf{h}_i, \mathbf{h}_j)} \Big[\log\left(\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j)\right)\Big] \\ & - \log \mathbb{E}_{p(\mathbf{h}_i) p(\mathbf{h}_j)} \Big[log\Big(1 - \mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j\right)\Big)\Big] \end{aligned} \tag{49}$$

Let $\mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j\right) = \text{sigmod}(\mathcal{D}'\left(\mathbf{h}_i, \mathbf{h}_j\right))$, the Equ.49 can be re-writed as a softplus (SP) version [18, 49], as follows

$$\begin{aligned} \mathcal{MI}_{SP}\left(\mathbf{h}_i, \mathbf{h}_j\right) = & \mathbb{E}_{p(\mathbf{h}_i, \mathbf{h}_j)} \Big[-sp\Big(-\mathcal{D}'\left(\mathbf{h}_i, \mathbf{h}_j\right)\Big)\Big] \\ & - \log \mathbb{E}_{p(\mathbf{h}_i) p(\mathbf{h}_j)} \Big[sp\Big(\mathcal{D}'\left(\mathbf{h}_i, \mathbf{h}_j\right)\Big)\Big] \end{aligned} \tag{50}$$

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

InfoNCE Estimator. InfoNCE [52] is one of the most popular lower-bound to the mutual information, defined as

$$\begin{aligned} \mathcal{MI}_{NCE}\left(\mathbf{h}_i, \mathbf{h}_j\right) = & \mathbb{E}_{p(\mathbf{h}_i, \mathbf{h}_j)} \Bigg[\mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j\right) \\ & - \mathbb{E}_{K \sim \mathcal{P}^N} \Big[\log \frac{1}{N} \sum_{\mathbf{h}'_j \in K} e^{\mathcal{D}\left(\mathbf{h}_i, \mathbf{h}'_j\right)}\Big]\Bigg] \end{aligned} \tag{51}$$

where K consists of N random variables sampled from a n identical and independent distribution. NT-Xent loss [53] 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. Taking 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 $\mathbf{h}_g = f_\gamma(\mathbf{A}, \mathbf{X})$. The InfoNCE is in practice computed on a mini-batch $\boldsymbol{B}$ of size $N + 1$, then the Equ. 51 can be re-writed (with $\log N$ discarded) as

$$\mathcal{MI}_{NCE} = -\frac{1}{N+1} \sum_{(\mathbf{A}, \mathbf{X}) \in \boldsymbol{B}} \log \frac{e^{\mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j\right)}}{\sum_{(\mathbf{A}', \mathbf{X}') \in \boldsymbol{B}} e^{\mathcal{D}\left(\mathbf{h}_i, \mathbf{h}'_j\right)}} \tag{52}$$

where $\mathbf{h}_i$, $\mathbf{h}_j$ are the positive pair of views that comes from the same graph $g = (\mathbf{A}, \mathbf{X})$, and $\mathbf{h}_i$ and $\mathbf{h}'_j$ are the negative

pair of views that are computed from $g = (\mathbf{A}, \mathbf{X})$ and $g' = (\mathbf{A}', \mathbf{X}')$ identically and independently.

Triplet Margin Loss. While all three MI estimators above estimate the lower bound on mutual information, mutual information maximization has been shown not to be a must for contrastive learning [54]. For example, the triplet margin loss [55] can also be used to optimize the contrastive learning, but it is not an MI-based contrastive objective, and optimizing it does not guarantee the maximization of mutual information. The triplet margin loss is defined as

$$\mathcal{L}_{triplet}\left(\mathbf{h}_i, \mathbf{h}_j\right) = \mathbb{E}_{[(\mathbf{A},\mathbf{X}),(\mathbf{A}',\mathbf{X}')]\sim\mathcal{G}\times\mathcal{G}}\Big[\max\{\mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j'\right) - \mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j\right) + \epsilon, 0\}\Big] \tag{53}$$

where the triplet margin loss does not directly minimize the agreement of the negative sample pair $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j')$, but only ensures that the agreement of the negative sample pair is smaller than that of the positive sample pair by a margin value ϵ. The idea behind is that when the negative samples are sufficiently far apart, i.e., the agreement between them is small enough, there is no need to further reduce their agreement, which helps to focus the training more on those hard samples that are hard to distinguish. The quadruplet loss [56] further considers imposing constraints on inter-class samples on top of the triplet margin loss, defined as:

$$\mathcal{L}_{Quadruplet}\left(\mathbf{h}_i, \mathbf{h}_j\right) = \mathcal{MI}_{triplet} + \mathbb{E}_{[(\mathbf{A},\mathbf{X}),(\mathbf{A}',\mathbf{X}')]\sim\mathcal{G}\times\mathcal{G}}\Big[\max\{\mathcal{D}\left(\mathbf{h}_i', \mathbf{h}_j'\right) - \mathcal{D}\left(\mathbf{h}_i, \mathbf{h}_j\right) + \epsilon', 0\}\Big] \tag{54}$$

where ϵ' is a smaller margin value than ϵ. The quadruplet loss differs from the triplet margin loss in that it not only uses an anchor-based sampling strategy but also samples negative samples in a more random way, which helps to learn more distinguishable inter-class boundaries.

RankMI Loss. While both triplet margin loss and quadruplet loss ignore the lower bound of the mutual information, the RankMI Loss [57] seamlessly incorporates information-theoretic approaches into the representation learning and maximizes the mutual information among samples belonging to the same category, defined as:

$$\mathcal{MI}_{RankMI}\left(\mathbf{h}_i, \mathbf{h}_j\right) = \mathbb{E}_{[(\mathbf{A},\mathbf{X}),(\mathbf{A}',\mathbf{X}')]\sim\mathcal{G}\times\mathcal{G}}\Big[\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j) + \log(2 - e^{\mathcal{D}(\mathbf{h}_i', \mathbf{h}_j')})\Big] \tag{55}$$

As RankMI can incorporate margins based on random positive and negative pairs, the quadruple loss can be considered as a special case of RankMI with a fixed margin.

4 Generative Learning

Compared with contrastive methods, the generative methods shown in Fig. 1(b) are based on generative models and treat rich information embedded in the data 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 graph reconstruction. Categorized by how the reconstruction is performed, we summarize generative methods into two categories: (1) graph autoencoding that performs reconstruction in a once-for-all manner; (2) graph autoregressive that iteratively performs reconstruction. In this section, due to space limitations, we present only some representative generative methods and place those relatively less important works in **Appendix B**.

4.1 Graph Autoencoding

Since the autoencoder [58] 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 [12]. 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,

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \widehat{\mathbf{X}}\right) = \left\| f_\theta(\mathbf{A}, \widehat{\mathbf{X}})_{v_i} - \mathbf{x}_i \right\|^2 \tag{56}$$

Here, it just takes one node as an example, and the reconstruction of multiple nodes can be considered in practice. Note that only those *unmasked* neighborhood nodes can be used to reconstruct the target node for graph completion.

Node Attribute Masking [10] is similar to Graph Completion, but it reconstructs the features of multiple nodes *simultaneously*, and it no longer requires that the neighboring node features used for reconstruction must be unmasked.

Edge Attribute Masking [11]. 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 $\widehat{\mathbf{x}}_{i,j}^e = \mathbf{x}_{i,j}^e \odot \mathbf{m}_{i,j}$ for $(v_i, v_j) \in \mathcal{M}_e$. More formally,

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \mathbf{X}, \widehat{\mathbf{X}}^e\right) = \frac{1}{|\mathcal{M}_e|} \sum_{(v_i,v_j)\in\mathcal{M}_e} \left\| \overline{\mathbf{x}}_{i,j}^e - \mathbf{x}_{i,j}^e \right\|^2 \tag{57}$$

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

Node Attribute Denoising [13]. 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,

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \widehat{\mathbf{X}}\right) = \frac{1}{N} \sum_{v_i\in\mathcal{V}} \left\| f_\theta(\mathbf{A}, \widehat{\mathbf{X}})_{v_i} - \mathbf{x}_i \right\|^2 \tag{58}$$

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

Adjacency Matrix Reconstruction [14]. The graph adjacency matrix is one of the most important information in

graph data, which stores the graph structure information and the relations 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,

$$\mathcal{L}_{ssl}\left(\theta, \widehat{\mathbf{A}}, \mathbf{X}\right) = \frac{1}{N^2}\sum_{i,j}\left(\overline{\mathbf{A}}_{i,j} - \mathbf{A}_{i,j}\right)^2 \tag{59}$$

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

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 [32]. In recent years, the idea of GPT [6] 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 next iteration $t+1$, with the learning objective defined as

$$\begin{aligned} & p_\theta\left(\mathbf{X}_{t+1}, \mathbf{A}_{t+1} \mid \mathbf{X}_t, \mathbf{A}_t\right) \\ =& \sum_o p_\theta\left(X_i, E_i^{\neg o} \mid E_i^o, \mathbf{X}_t, \mathbf{A}_t\right) \cdot p_\theta\left(E_i^o \mid \mathbf{X}_t, \mathbf{A}_t\right) \\ =& \mathbb{E}_o\left[p_\theta\left(X_i, E_i^{\neg o} \mid E_i^o, \mathbf{X}_t, \mathbf{A}_t\right)\right] \\ =& \mathbb{E}_o\left[p_\theta\left(\mathbf{X}_{t+1} \mid E_i^o, \mathbf{X}_t, \mathbf{A}_t\right) p_\theta(E_i^{\neg o} \mid E_i^o, \mathbf{X}_{t+1}, \mathbf{A}_t)\right] \end{aligned} \tag{60}$$

where o is a variable to denote the index vector of all 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, GPT-GNN performs node and edge generation iteratively.

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 are 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. In this section, due to space limitations, we present only some representative predictive methods and place those relatively less important works in **Appendix C**.

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. node properties, such as node degree and local clustering coefficient. Specifically, it first defines a mapping $\Omega : \mathcal{V} \to \mathcal{Y}$ 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 is then formulated as

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \frac{1}{N}\sum_{v_i \in \mathcal{V}}\Big(f_\theta(\mathbf{A}, \mathbf{X})_{v_i} - y_i\Big)^2 \tag{61}$$

where $f_\theta(\mathbf{A}, \mathbf{X})_{v_i}$ is the predicted label of node v_i. With different node properties, the mapping function $\Omega(\cdot)$ can have different designs. If it use node degree as a local node property for self-supervision, we have $y_i = \Omega(\mathbf{A}, \mathbf{X})_{v_i} = \sum_{j=1}^{N} \mathbf{A}_{i,j}$. For the local clustering coefficients, we have

$$y_i = \Omega(\mathbf{A}, \mathbf{X})_{v_i} = \frac{2\left|\{(v_m, v_n) | v_m \in \mathcal{N}_i, v_n \in \mathcal{N}_i\}\right|}{|\mathcal{N}_i|(|\mathcal{N}_i| - 1)} \tag{62}$$

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.

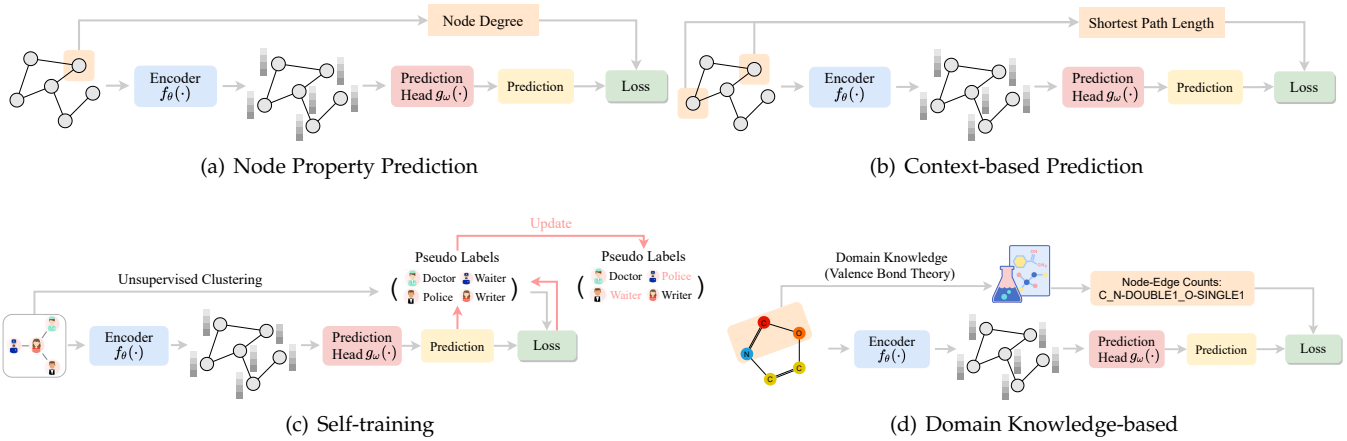
5.2 Context-based Prediction (CP)

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 methods because it generally explores contextual information.

S^{2}GRL [59]. Motivated by the observation that two arbitrary nodes in a graph can interact with each other through paths of different lengths, S^{2}GRL 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

$$\begin{aligned} \mathcal{L}_{ssl}\left(\theta, \omega, \mathbf{A}, \mathbf{X}\right) = \frac{1}{NK}\sum_{v_i \in \mathcal{V}}\sum_{k=1}^{K}\sum_{v_j \in \mathcal{C}_i^k} \ell\Big(f_w\big(\\ f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, f_\theta(\mathbf{A}, \mathbf{X})_{v_j}\big), k\Big) \end{aligned} \tag{63}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3131584, IEEE Transactions on Knowledge and Data Engineering



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

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

Fig. 5. A comparison of predictive learning methods. Categorized by how the labels are obtained, we summarize predictive methods for graph data into four categories: node property prediction, context-based prediction, self-training, and domain knowledge-based prediction. **Fig. (a)**: the node property prediction pre-calculates 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.

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** [10] 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 [10]. 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 wishes to preserve the node pairwise similarity to some extent, rather than allowing a node's neighborhood to drastically change it. Thus, the pretext task of PairwiseAttrSim can be established to achieve node similarity preservation. 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

$$\begin{aligned}\mathcal{S}_{i,h} &= \left\{(v_i, v_j) \mid s_{ij} \text{ in top-K of } \{s_{ik}\}_{k=1,k\neq i}^{N}\right\} \\ \mathcal{S}_{i,l} &= \left\{(v_i, v_j) \mid s_{ij} \text{ in bottom-K of } \{s_{ik}\}_{k=1,k\neq i}^{N}\right\}\end{aligned} \tag{64}$$

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

$$\begin{aligned}\mathcal{L}_{ssl}(\theta, \omega, \mathbf{A}, \mathbf{X}) = \frac{1}{2NK} \sum_{v_i \in \mathcal{V}} \sum_{(v_m, v_n) \in \mathcal{S}_i} \Big(f_w \big(\\ f_\theta(\mathbf{A}, \mathbf{X})_{v_m}, f_\theta(\mathbf{A}, \mathbf{X})_{v_n} \big) - s_{m,n} \Big)^2\end{aligned} \tag{65}$$

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

Distance2Clusters [10]. 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 [60–68], 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

$$\mathcal{L}_{ssl}(\theta, \mathbf{A}, \mathbf{X}) = \frac{1}{N} \sum_{v_i \in \mathcal{V}} \left\| f_\theta(\mathbf{A}, \mathbf{X})_{v_i} - \mathbf{d}_i \right\|^2 \tag{66}$$

Meta-path Prediction [69]. A meta-path of length l is a sequence of nodes connected with heterogeneous edges, i.e., $v_1 \xrightarrow{t_1} v_2 \xrightarrow{t_2} \ldots \xrightarrow{t_l} v_l$, where $t_l \in \mathcal{T}^e$ denote the type of l-th edge in the meta-path. Given a set of node pair $\mathcal{S}$ sampled from the heterogeneous graph and K *pre-defined* meta-path types $\mathcal{M}$, this pretext task aims to predict if the two nodes $(v_i, v_j) \in \mathcal{S}$ are connected by one of the meta-path type $m \in \mathcal{M}$. Finally, the predictions of the K meta-paths are formulated as K binary classification tasks, as follows

$$\begin{aligned}\mathcal{L}_{ssl}(\theta, \mathbf{A}, \mathbf{X}) = \frac{1}{K|\mathcal{S}|} \sum_{m \in \mathcal{M}} \sum_{(v_i, v_j) \in \mathcal{S}} \\ \ell\Big(f_w\big(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, f_\theta(\mathbf{A}, \mathbf{X})_{v_j} \big), \mathbf{Y}_{i,j}^m \Big)\end{aligned} \tag{67}$$

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 [70]. Different from the pretext task of Meta-path Prediction [69] 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,

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \prod_{v_i \in \mathcal{S}} \prod_{v_i^m \in g_i} p\left(v_i^m \mid \mathbf{H}_i, \theta\right) \tag{68}$$

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

Distance2Labeled [10]. 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 explore label-specific self-supervised tasks. For example, we directly modify the pretext task of Distance2Cluster by combining label information to create a new pretext task - Distance2Labeled. 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 Distance2Labeled can be formulated as a distance regression problem, as follows

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \frac{1}{N} \sum_{v_i \in \mathcal{V}} \left\| f_\theta(\mathbf{A}, \mathbf{X})_{v_i} - \mathbf{d}_i \right\|^2 \tag{69}$$

Compared with Distance2Cluster, Distance2Labeled utilizes task-specific label information rather than additional unsupervised clustering algorithms to find cluster centers, showing advantages in both efficiency and performance.

5.3 Self-Training (ST)

For self-training methods, the prediction results from the previous stage can be used as labels to guide the training of next stage, thus achieving self-training in an iterative way.

Multi-stage Self-training [71]. 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

$$\mathcal{L}_{node}\left(\theta, \mathbf{A}, \mathbf{X}, \mathcal{D}_L^t\right) = \sum_{\left(v_i, y_i^t\right) \in \mathcal{D}_L^t} \ell\Big(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, y_i^t\Big) \tag{70}$$

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

$$\mathcal{D}_N^t = \Big\{(v_i, \widehat{y}_i^t) | \mid \widehat{y}_i^t \text{ in top-K confidence of } \widehat{\mathcal{Y}}^t\Big\} \tag{71}$$

are considered as the pseudo-labels and moved to the labeled node set $\mathcal{D}_L^t$ to obtain an updated labeled set $\mathcal{D}_L^{t+1} = \mathcal{D}_L^t \bigcup \mathcal{D}_N^t$ and an 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 [12]. 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

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \frac{1}{N} \sum_{v_i \in \mathcal{V}} \ell\Big(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, y_i\Big) \tag{72}$$

When node attributes are not available, another choice to obtain pseudo-labels is based on the topology of a given graph structure or adjacency matrix. Specifically, graph partitioning [72, 73] 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** [74] combines the node clustering and node partitioning to proposed a new pretext task. Concretely, it first assigns cluster indices as pseudo labels but follows a topology refining process that refines the clusters by minimizing the inter-cluster edges.

M3S [75]. Combining Multi-stage Self-training with Node Clustering, M3S applies DeepCluster [68] and the alignment mechanism as a self-checking mechanism, thus providing 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 will be trained on the labeled set $\mathcal{D}_L^{t+1}$ with the objective defined as Equ. 70.

Cluster Preserving [17]. An important characteristic of real-world graphs is the cluster structure, so we can consider the cluster preservation as a self-supervised 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 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,

$$\widehat{y}_{i,k} = \frac{\exp\Big(g_\omega(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, c_k)\Big)}{\sum_{k=1}^K \exp\Big(g_\omega(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, c_k)\Big)} \tag{73}$$

Finally, the objective of Cluster Preserving is defined as

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \mathbf{X}\right) = -\frac{1}{N} \sum_{v_i \in \mathcal{V}} \sum_{k=1}^K y_{i,k} \log(\widehat{y}_{i,k}) \tag{74}$$

where the ground-truth label $y_{i,k} = 1$ if node v_i is grouped into 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.

Contextual Molecular Property Prediction [76] incorporates domain knowledge about biological macromolecules to design molecule-specific pretext tasks. Given a node v_i,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2021.3131584, IEEE Transactions on Knowledge and Data Engineering

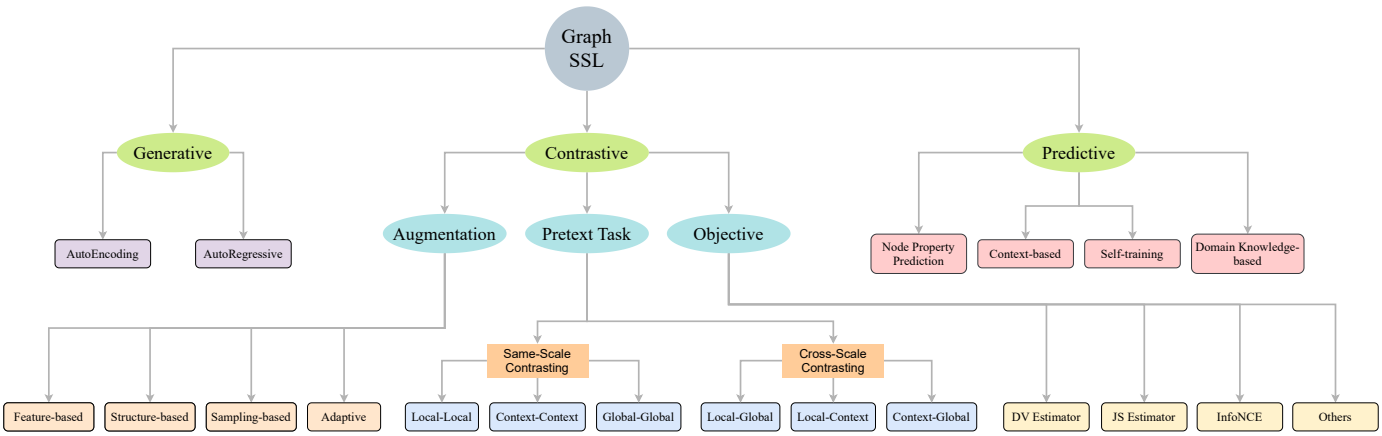


Fig. 6. An summary of graph self-supervised learning (SSL) methods. We categorize them into three branches: contrastive, generative, and predictive. For contrastive methods, they contrast different views and deal with data-data pairs (inter-data) information, and we further categorize it from three aspects: 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, they focus 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).

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., C_N-DOUBLE1_O-SINGLE1 in Fig. 5 (d). With plenty of context-aware properties $\mathcal{P} = \{p_k\}_{k=1}^{K}$ pre-defined, the contextual property prediction can be defined as a multi-class prediction problem with one class corresponds to one contextual property, as follows

$$\mathcal{L}_{ssl}\left(\theta, \mathbf{A}, \mathbf{X}\right) = \frac{1}{N} \sum_{v_i \in \mathcal{V}} \ell\Big(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, y_i\Big) \tag{75}$$

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

Graph-level Motif Prediction [76]. Motifs are recurrent sub-graphs among the input graph, which are prevalent in molecular graphs. 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 considering the presence of K motifs $\{m_k\}_{k=1}^{K}$ in the molecular data, then for one specific molecule graph $g_i = (\mathbf{A}_i, \mathbf{X}_i) \in \mathcal{G}$, it detects whether each motif shows up in g_i and use it as the label $\mathbf{y}_i \in \mathbb{R}^K$. Specifically, 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

$$\mathcal{L}_{ssl}\left(\gamma, \mathcal{G}\right) = \frac{1}{|\mathcal{G}|} \sum_{g_i \in \mathcal{G}} \ell\Big(f_\gamma(\mathbf{A}_i, \mathbf{X}_i), \mathbf{y}_i\Big) \tag{76}$$

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

6 Summary of the Implementation

A summary of the surveyed works is presented in Fig. 6, and **Appendix D** lists their properties, including graph property, pretext task, augmentation, objective function, training strategy, and publication year. Furthermore, we show in **Appendix E** the implementation details of surveyed works, such as downstream tasks, evaluation metrics, and 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. 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 classification and node regression are usually performed with partial known labels. Instead, the node clustering is performed in a more challenging unsupervised manner and adopted when the performance of the node classification is not sufficiently distinguishable.

6.2 Evaluation Metrics

For **graph classification** tasks, the commonly used evaluation metrics include ROC-AUC and Accuracy (Acc); while for **graph regression** tasks, Mean Absolute Error (MAE) is used. In terms of **link prediction** tasks, ROC-AP, ROC-PR, and ROC-AUC are usually used as evaluation metrics. Besides, **node regression** tasks are usually evaluated by metrics including MAE, Mean Square Error (MSE), and Mean Absolute Percentage Error (MAPE). In addition to Accuracy, **node classification** tasks also adopt F1-score for single-label classification and Micro-F1 (or Macro-F1) for multi-label classification. Moreover, **node clustering** tasks often adopt the same metrics used to evaluate the unsupervised clustering, such as Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), Accuracy, etc.

6.3 Datasets

The statistics of a total of 41 datasets are available in **Appendix F**. 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. (1) *Citation Networks.* In citation networks, nodes usually denote papers, node attributes are some keywords in papers, edges denote cross-citation, and categories are topics of papers. Note that nodes in the citation networks may also sometimes indicate authors, institutions, etc. (2) *Social Networks.* The social network datasets consider 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 datasets, such as Reddit [8], COLLAB [77]. (3) *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. (4) *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 [78] and D&D [78], 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 [79], used for graph biological function prediction. (5) *Other Graphs.* In addition to the four types of datasets mentioned above, there are some datasets that are less common or difficult to categorize, such as image, traffic, and co-purchase datasets.

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 71 surveyed works is presented in **Appendix G**, where we provide hyperlinks to their open-source codes. 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.

6.5 Experimental Study

To make a fair comparison, we select two important downstream tasks, node classification and graph classification, and provide the classification performance of various classical algorithms on 15 commonly used graph datasets in **Appendix H**. Due to space limitations, please refer to the appendix for more experimental results and analysis.

7 Discussion

We begin with some discussion and summary of the connections and developments between various methods. To present a clearer picture of the development lineage of various graph SSL methods, we provide a complete timeline in **Appendix I**, listing the publication dates of some key milestones. Besides, we provide the inheritance connections between methods to show how they are developed. Furthermore, we provide short descriptions of contributions to some seminal works to highlight their importance.

DGI [9] is a pioneering work for graph contrastive learning, originally designed specifically for node classification on attribute graphs, and later extended to other types of graphs, resulting in new variants such as HDGI [31] for heterogeneous graphs, STDGI [28] for spatial-temporal graphs, DMGI [80] for multiplexed graphs, and BiGI [37] for bipartite graphs. Besides, InfoGraph [49] extends DGI to global-context contrasting, achieving state-of-the-art performance on multiple graph classification datasets. With the focus shifted from local nodes and global graphs to subgraphs, GCC [38] proposes the first subgraph-level context-context contrasting framework, where subgraphs sampled from the same graph are considered as the positive pair.

Different from DGI, GRACE [26] focuses on contrasting views at the node-level by generating multiple augmented graphs through handcrafted augmentations and then encouraging consistency between the same nodes in different views. GCA [40] adopts a similar framework to GRACE, but focuses on designing the *adaptive* augmentation strategy. Similarly, GROC [43] claims that gradient information can be used to guide data augmentation and proposes a gradient-based graph topology augmentation that further improves the performance of GRACE and GCA. The same local-local contrasting as GRACE, but BGRL [27] is inspired by BYOL [81] and explores for the first time whether negative samples are a must for graph contrastive learning.

The focus on three different levels of nodes, edges, and structures has led to different generative methods such as Graph Completion [12], Edge Feature Masking [11], and Adjacency Matrix Reconstruction [14], respectively. Moreover, due to their simplicity and effectiveness, these methods have been widely used in algorithms such as GPT-GNN [32] and recommendation applications such as Pretrain-Recsys [82]. In terms of predictive methods, the basic difference between methods is how to obtain pseudo-labels, and there are three main means: (1) numerical statistics, such as Node Property Prediction and PairwiseAttrSim [10]; (2) prediction results from the previous training stage, such as CAGNN [74] and M3S [75]; (3) domain knowledge, such as Molecular Property Prediction and Global Motif Prediction [76].

Discussion on Pros and Cons. Next, we will discuss the advantages and disadvantages of some classical algorithms on four aspects: innovation, accessibility, effectiveness (performance), and efficiency, based on which we divide existing algorithms into four categories: (1) Pioneering work. Representative works include DGI [9], InfoGraph [49], HDGI [31], STDGI [28], etc. These methods, for the first time, apply contrastive learning to a novel downstream task or graph type, showing promising innovations and inspiring many follow-up researches. However, as early attempts, they often

perform relatively poorly on downstream tasks and with high computational complexity, compared to some subsequent works. (2) Knowledge-based work. There are some works that combine self-supervised learning techniques with prior knowledge to obtain excellent performance on downstream tasks. For example, Molecular Property Prediction [76] combines domain knowledge, i.e., molecular properties, while LCC [45] introduces label information into the computation of supervision signals. These knowledge-based methods usually achieve fairly good performance due to the introduction of additional information, but this also limits their applicability to other tasks and graph types. (3) There is some work aimed at building on existing work and pursuing state-of-the-art performance on a variety of datasets, among which representative works include K2SL [83], BGRL [27], and SUGAR [84]. While these works have achieved state-of-the-art performance on a variety of downstream tasks, they are largely incremental contributions to previous seminal work (e.g., DGI) and are relatively weak on innovation and accessibility. (4) Most generative and predictive methods are less effective than contrastive methods, but are generally very simple to implement, easy to combine with existing frameworks, have lower computational complexity, and exhibit better applicability and efficiency.

Pretext Tasks for Complex Types of Graph. Most of the existing work is focused on the design of pretext tasks, especially on attribute graphs, with little effort to other more complex graph types, such as spatial-temporal and heterogeneous graphs. Moreover, these pretext tasks usually utilize only node-level or graph-level features, limiting their ability to exploit richer information, such as temporal information in spatial-temporal graphs and relation information in heterogeneous graphs. As a result, how to design more suitable pretext tasks can be considered from three aspects: (1) designing *graph type-specific* pretext tasks that adaptively pick the most suitable tasks depending on the type of graph; (2) incorporating temporal or heterogeneous information (in the form of prior knowledge) into the pretext task design; (3) taking the automated design of pretext tasks as a new research topic from the perspective of automatic learning.

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. For example, an important problem for graph SSL is whether mutual information maximization is the only means to achieve graph contrastive learning? Such problems have been explored in [54] for image data, but how to extend them to the graph domain is not yet available. In Sec. 3.4, in addition to MI estimators such as InforNCE, we have introduced some contrastive objectives that are not based on mutual information, such as triplet margin and quadruplet loss. However, how to theoretically analyze the connection between these losses and mutual information needs to be further explored.

Insufficient Augmentation Strategy. Recent advances in the field of visual representation learning [2, 3] are mainly attributed to a variety of data augmentation strategies, such as resize, rotation, coloring, etc. However, due to the inherent non-Euclidean nature of graph data, it is difficult to directly apply existing image-based augmentation to graphs. Moreover, most 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. More importantly, the design of the augmentation strategy should follow some well-designed guidelines instead of relying entirely on subjective intuition. In summary, we argue that the design of graph augmentation should be based on the following four guidelines: (1) Applicability, graph augmentation should ideally be a plug-and-play module that can be easily combined with the existing self-supervised learning frameworks. (2) Adaptability, some work [10, 27] have pointed out that different datasets and task types may require different augmentations, so how to design the date-specific and task-specific augmentation strategy is a potential research topic. (3) Efficiency, data augmentation should be a lightweight module that does not bring a huge computational burden to the original implementation. (4) Dynamic, with the ability to dynamically update the augmentation strategy as the training proceeds.

Inefficient Negative Sampling Strategy. The selection of high-quality negative samples is a crucial issue. The most common sampling strategy is uniform sampling, but this has been shown to be very informative [85–87]. The problem of how to better obtain negative samples has been well explored in the field of computer vision. For example, [85] presents the debiased contrastive learning that directly corrects the sampling bias of negative samples. Besides, [86] takes advantage of mixup techniques to directly synthesize hard negative samples in the embedding space. Moreover, [87] develops a family of unsupervised sampling strategies for user-controllable negative sample selection. Despite the great success, these methods, specifically designed for image data, may be difficult to apply directly to non-Euclidean graph data. More importantly, accurate estimation of hard negative samples becomes more difficult when label information is not available. Therefore, how to reduce the gap between ideal and practical contrastive learning by a decent negative sampling strategy requires more exploration.

Lack of Explainability. 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 pretext 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.

Margin from Pre-training to Downstream Tasks. Pre-training with self-supervised tasks and then using the pre-trained model for specific downstream tasks, either by fine-tuning or freezing the weights, is a common training strategy in graph SSL [82, 88, 89]. However, how shall we transfer the pre-trained knowledge to downstream tasks? Though numerous strategies have been proposed to address this problem in the CV and NLP domains [90], they are difficult to apply directly to graphs due to the inherent non-Euclidean structure of graphs. Therefore, it is an important issue to design graph-specific techniques to minimize the margin between pre-training and downstream tasks.

8 Conclusion

A comprehensive survey of the literature on graph self-supervised learning techniques is conducted in this paper. We develop a unified mathematical framework for graph SSL. Moreover, we summarize the implementation details in each work and show their similarities and differences. More importantly, we are the first survey to provide a detailed experimental study on self-supervised learning, setting the stage for the future development of graph SSL. Finally, we point out the technical limitations of the current research and provide promising directions for future work on graph SSL. We hope this survey will inspire follow-up researchers to focus on other important but easy-to-miss details such as theoretical foundations, explainability, etc., in addition to model performance on downstream tasks.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." in *ICLR (Poster)*, 2019.

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

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

[31] Y. Ren, B. Liu, C. Huang, P. Dai, L. Bo, and J. Zhang,

"Heterogeneous deep graph infomax," *arXiv preprint arXiv:1911.08538*, 2019.

[32] 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.

[33] 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.

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

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

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

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

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

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

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

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

[46] J. Yu, H. Yin, M. Gao, X. Xia, X. Zhang, and N. Q. V. Hung, "Socially-aware self-supervised tri-training for recommendation," *arXiv preprint arXiv:2106.03569*, 2021.

[47] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," *arXiv preprint arXiv:2105.09111*, 2021.

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

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.

[54] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic, "On mutual information maximization for representation learning," *arXiv preprint arXiv:1907.13625*, 2019.

[55] 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.

[56] W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: a deep quadruplet network for person re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 403–412.

[57] M. Kemertas, L. Pishdad, K. G. Derpanis, and A. Fazly, "Rankmi: A mutual information maximizing ranking loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 362–14 371.

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

[59] 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.

[60] 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.

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

[62] 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.

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

[64] 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.

[65] 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.

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

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

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

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] C.-Y. Chuang, J. Robinson, Y.-C. Lin, A. Torralba, and S. Jegelka, "Debiased contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[86] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, "Hard negative mixing for contrastive learning," *arXiv preprint arXiv:2010.01028*, 2020.

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

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

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

[90] 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.

Lirong Wu received the B.S. degree from the the Department of Information Science & Electronic Engineering, Zhejiang University, Hangzhou, China, in 2020. He is currently pursuing the Ph.D. degree with the School of Engineering, Westlake University, Hangzhou, China. His main research interests include low-level vision, video compression, deep clustering, graph learning, self-supervised learning, and deep learning.

Haitao Lin received the B.S. degree in Material Physics and Statistics from Sichuan University, Chengdu, China. He is currently pursuing the Ph.D. degree with the School of Engineering, Westlake University, Hangzhou, China. His main research interests include spatio-temporal model, multivariate time series forecasting with relational inference, and dynamic system in biology and physics.

Cheng Tan received the B.S. degree from the the College of Information Engineering, Northwest A&F University, Xianyang, China, in 2021. He is currently pursuing the Ph.D. degree with the School of Engineering, Westlake University, Hangzhou, China. His main research interests include self-supervised learning and spatiotemporal learning.

Zhangyang Gao received the B.S. degree from the School of Automation, Central South University, Changsha, China, in 2020. He is currently a Ph.D. candidate at the School of Engineering, Westlake University, Hangzhou, China. His main research interests include clustering, unsupervised video tasks, and deep learning.

Stan Z. Li (Fellow, IEEE) received the B.Eng. degree from Hunan University, China, the M.Eng. degree from the National University of Defense Technology, China, and the Ph.D. degree from the University of Surrey, U.K. He is currently a Professor and the Director of the Center for Biometrics and Security Research (CBSR), Institute of Automation, Chinese Academy of Sciences (CASIA). From 2000 to 2004, he worked at Microsoft Research Asia, as a Researcher. Prior to that, he was an Associate Professor with Nanyang Technological University, Singapore. He has published over 200 papers in international journals and conferences and authored and edited eight books. His research interests include pattern recognition and machine learning, image and vision processing, face recognition, biometrics, and intelligent video surveillance. He was elevated to a fellow of the IEEE, for his contributions to the fields of face recognition, pattern recognition, and computer vision. He served as the Program Co-Chair for the International Conference on Biometrics in 2007 and 2009, and has been involved in organizing other international conferences and workshops in the fields of his research interest.