Federation University ResearchOnline
https://researchonline.federation.edu.au
Copyright Notice

This is the peer-reviewed version of the following article:

Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., & Yu, P. (2022). Graph Self-Supervised Learning: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 1–1.

Available online: https://doi.org/10.1109/TKDE.2022.3172903

See this record in Federation ResearchOnline at:
http://researchonline.federation.edu.au/vital/access/HandleResolver/1959.17/189021

Graph Self-Supervised Learning: A Survey

Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, Philip S. Yu, *Life Fellow, IEEE*

Abstract—Deep learning on graphs has attracted significant interests recently. However, most of the works have focused on (semi-) supervised learning, resulting in shortcomings including heavy label reliance, poor generalization, and weak robustness. To address these issues, self-supervised learning (SSL), which extracts informative knowledge through well-designed pretext tasks without relying on manual labels, has become a promising and trending learning paradigm for graph data. Different from SSL on other domains like computer vision and natural language processing, SSL on graphs has an exclusive background, design ideas, and taxonomies. Under the umbrella of *graph self-supervised learning*, we present a timely and comprehensive review of the existing approaches which employ SSL techniques for graph data. We construct a unified framework that mathematically formalizes the paradigm of graph SSL. According to the objectives of pretext tasks, we divide these approaches into four categories: generation-based, auxiliary property-based, contrast-based, and hybrid approaches. We further describe the applications of graph SSL across various research fields and summarize the commonly used datasets, evaluation benchmark, performance comparison and open-source codes of graph SSL. Finally, we discuss the remaining challenges and potential future directions in this research field.

Index Terms—Self-supervised learning, graph analytics, deep learning, graph representation learning, graph neural networks.

✦

1 INTRODUCTION

IN recent years, deep learning on graphs [1], [2], [3], [4] has become increasingly popular for the artificial intelligence research community since graph-structured data is ubiquitous in numerous domains, including e-commerce [5], traffic [6], chemistry [7], and knowledge base [8]. Most deep learning studies on graphs focus on (semi-) supervised learning scenarios, where specific downstream tasks (*e.g.*, node classification) are exploited to train models with well-annotated manual labels. Despite the success of these studies, the heavy reliance on labels brings several shortcomings. Firstly, the cost of the collection and annotation of manual labels is prohibitive, especially for the research areas which have large-scale datasets (*e.g.*, citation and social networks [9]) or demand on domain knowledge (*e.g.*, chemistry and medicine [10]). Secondly, a purely supervised learning scenario usually suffers from poor generalization owing to the over-fitting problem, particularly when training data is scarce [11]. Thirdly, supervised graph deep learning models are vulnerable to label-related adversarial attacks, causing the weak robustness of graph supervised learning [12].

To address the shortcomings of (semi-) supervised learning, self-supervised learning (SSL) provides a promising learning paradigm that reduces the dependence on manual labels. In SSL, models are learned by solving a series of handcrafted auxiliary tasks (so-called pretext tasks), in which the supervision signals are acquired from data itself automatically without the need for manual annotation. With the help of well-designed pretext tasks, SSL enables the model to learn more informative representations from unlabeled data to achieve better performance [13], [14], generalization [9], [15], [16] and robustness [17], [18] on various downstream tasks.

Described as "the key to human-level intelligence" by Turing Award winners Yoshua Bengio and Yann LeCun, SSL has recently achieved great success in the domains of computer vision (CV) and natural language processing (NLP). Early SSL methods in CV domain design various semantics-related pretext tasks for visual representation learning [19], such as image inpainting [20], image colorizing [21], and jigsaw puzzle [22], etc. Lately, self-supervised contrastive learning frameworks (*e.g.*, MoCo [23], SimCLR [24] and BYOL [25]) leverage the invariance of semantics under image transformation to learn visual features. In the NLP domain, early word embedding methods [26], [27] share the same idea with SSL which learns from data itself. Pre-trained by linguistic pretext tasks, recent large-scale language models (*e.g.*, BERT [28] and XLNet [29]) achieve state-of-the-art performance on multiple NLP tasks.

Following the immense success of SSL on CV and NLP, very recently, there has been increasing interest in applying SSL to graph-structured data. However, it is non-trivial to transfer the pretext tasks designed for CV/NLP for graph data analytics. The main challenge is that graphs are in irregular non-Euclidean data space. Compared to the 2D/1D regular-grid Euclidean spaces where image/language data reside in, non-Euclidean spaces are more general but more complex. Therefore, some pretext tasks for grid-structure data cannot be mapped to graph data directly. Furthermore, the data examples (nodes) in graph data are correlated with the topological structure naturally, while the examples in

- *Y. Liu, M. Jin, and S. Pan are with Department of Data Science & AI, Faculty of IT, Monash University, VIC 3800, Australia (E-mail: yixin.liu@monash.edu; ming.jin@monash.edu; shirui.pan@monash.edu;).*
- *C. Zhou is with Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China (Email: zhouchuan@amss.ac.cn).*
- *Y. Zheng is with the Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia (E-mail: yu.zheng@latrobe.edu.au).*
- *F. Xia is with School of Engineering, Information Technology and Physical Sciences, Federation University, Australia (Email: f.xia@federation.edu.au).*
- *P. S. Yu is with Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7053, USA (Email: psyu@uic.edu)*
- *Corresponding author: Shirui Pan.*

(a) SSL pretext tasks in CV: image colorizing (upper) and image contrastive learning (bottom).

(b) SSL pretext tasks in NLP: masked language modeling (upper) and next sentence prediction (bottom).

(c) SSL pretext tasks in graph analytics: masked graph generation (upper) and node contrastive learning (bottom).
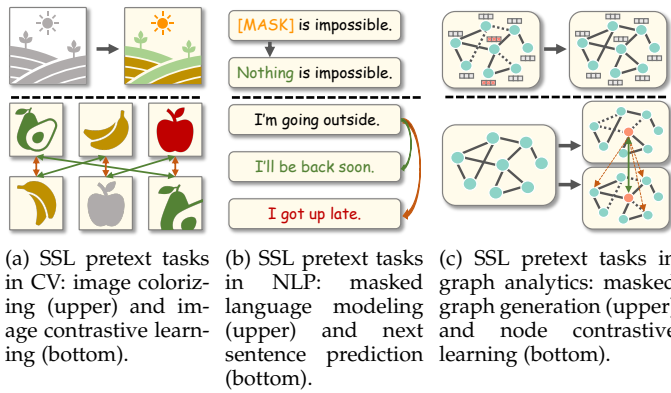
Fig. 1: Toy examples of different SSL pretext tasks in CV, NLP and graph analytics. In generative tasks, graph SSL should consider the topological structure in an irregular grid as well as node features, while SSL in CV/NLP just needs to recover the information in 2D/1D grid space. In contrastive tasks, the dependency between nodes is non-negligible in graph SSL, while the samples in CV/NLP are independent.

CV (image) and NLP (text) are often independent. Hence, how to deal with such dependency in graph SSL becomes a challenge for pretext task designs. Fig. 1 illustrates such differences with some toy examples. Considering the significant difference between SSL in graph analytics and other research areas, exclusive definitions and taxonomies are required for graph SSL.

The history of graph SSL goes back to at least the early studies on unsupervised graph embedding [30], [31] [1]. These methods learn node representations by maximizing the agreement between contextual nodes within truncated random walks. A classical unsupervised learning model, graph autoencoder (GAE) [32], can also be regarded as a graph SSL method that learns to rebuild the graph structure. Since 2019, the recent wave of graph SSL has brought about various designs of pretext tasks, from contrastive learning [13], [33] to graph property mining [10], [17]. Considering the increasing trend of graph SSL research and the diversity of related pretext tasks, there is an urgent need to construct a unified framework and systematic taxonomy to summarize the methodologies and applications of graph SSL.

To fill the gap, this paper conducts a comprehensive and up-to-date overview of the rapidly growing area of graph SSL, and also provides abundant resources and discussions of related applications. The intended audiences for this article are general machine learning researchers who would like to know about self-supervised learning on graph data, graph learning researchers who want to keep track of the most recent advances on graph neural networks (GNNs), and domain experts who would like to generalize graph SSL approaches to new applications or other fields. The core contributions of this survey are summarized as follows:

- **Unified framework and systematic taxonomy.** We propose a unified framework that mathematically formalizes graph SSL approaches. Based on our framework, we systematically categorize the existing

works into four groups: generation-based, auxiliary property-based, contrast-based, and hybrid methods. We also build the taxonomies of downstream tasks and SSL learning schemes.
- **Comprehensive and up-to-date review.** We conduct a comprehensive and timely review for classical and latest graph SSL approaches. For each type of graph SSL approach, we provide fine-grained classification, mathematical description, detailed comparison, and high-level summary.
- **Abundant resources and applications.** We collect abundant resources on graph SSL, including datasets, evaluation benchmark, performance comparison, and open-source codes. We also summarize the practical applications of graph SSL in various research fields.
- **Outlook on future directions.** We point out the technical limitations of current research. We further suggest six promising directions for future works from different perspectives.

Comparison with related survey articles. Some existing surveys mainly review from the perspectives of general SSL [34], SSL for CV [19], or self-supervised contrastive learning [35], while this paper purely focuses on SSL for graph-structured data. Compared to the recent surveys on graph self-supervised learning [36], [37], our survey has a more comprehensive overview on this topic and provides the following differences: (1) a unified encoder-decoder framework to define graph SSL; (2) a systematical and more fine-grained taxonomy from a mathematical perspective; (3) more up-to-date review; (4) more detailed summary of resources including performance comparison, datasets, implementations, and practical applications; and (5) more forward-looking discussion for challenges and future directions.

The remainder of this article is organized as follows. Section 2 defines the related concepts and provides notations used in the remaining sections. Section 3 describes the framework of graph SSL and provides categorization from multiple perspectives. Section 4-7 review four categories of graph SSL approaches respectively. Section 8 summarizes the useful resources for empirical study of graph SSL, including performance comparison, datasets, and open-source implementations. Section 9 surveys the real-world applications in various domains. Section 10 analyzes the remaining challenges and possible future directions. Section 11 concludes this article in the end.

2 DEFINITION AND NOTATION

In this section, we outline the related term definitions of graph SSL, list commonly used notations, and define graph-related concepts.

2.1 Term Definitions

In graph SSL, we provide the following definitions of related essential concepts.

Manual Labels Versus Pseudo Labels. *Manual labels*, a.k.a. human-annotated labels in some papers [19], indicate the labels that human experts or workers manually annotate. *Pseudo labels*, in contrast, denote the labels that can be

---

1. A timeline of milestone works are summarized in Appendix A.

acquired automatically from data by machines without any human knowledge. In general, pseudo labels require lower acquisition costs than manual labels so that they have advantages when manual labels are difficult to obtain or the amount of data is vast. In self-supervised learning settings, specific methods can be designed to generate pseudo labels, enhancing the representation learning.

Downstream Tasks Versus Pretext Tasks. *Downstream tasks* are the graph analytic tasks used to evaluate the quality or performance of the feature representation learned by different models. Typical applications include node classification and graph classification. *Pretext tasks* refer to the pre-designed tasks for models to solve (*e.g.*, graph reconstruction), which helps models to learn more generalized representations from unlabeled data, and thus benefits downstream tasks by providing a better initialization or more effective regularization. In general, solving downstream tasks needs manual labels, while pretext tasks are usually learned with pseudo labels.

Supervised Learning, Unsupervised Learning and Self-Supervised Learning. *Supervised learning* refers to the learning paradigm that leverages well-defined manual labels to train machine learning models. Conversely, *unsupervised learning* refers to the learning paradigm without using any manual labels. As a subset of unsupervised learning, *self-supervised learning* indicates the learning paradigm where supervision signals are generated from data itself. In self-supervised learning methods, models are trained with pretext tasks to obtain better performance and generalization on downstream tasks.

2.2 Notations

We provide important notations used in this paper (which are summarized in Appendix B.1) and the definitions of different types of graphs and GNNs in this subsection.

Definition 1 (Plain Graph). A plain graph[2] is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ ($|\mathcal{V}| = n$) is the set of nodes and $\mathcal{E}$ ($|\mathcal{E}| = m$) is the set of edges, and naturally we have $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The neighborhood of a node v_i is denoted as $\mathcal{N}(v_i) = \{v_j \in \mathcal{V} | e_{i,j} \in \mathcal{E}\}$. The topology of the graph is represented as an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}_{i,j} = 1$ means $e_{i,j} \in \mathcal{E}$, and $\mathbf{A}_{i,j} = 0$ means $e_{i,j} \notin \mathcal{E}$.

Definition 2 (Attributed Graph). An attributed graph refers to a graph where nodes and/or edges are associated with their own features (a.k.a attributes). The feature matrices of nodes and edges are represented as $\mathbf{X_{node}} \in \mathbb{R}^{n \times d_{node}}$ and $\mathbf{X_{edge}} \in \mathbb{R}^{m \times d_{edge}}$ respectively. In a more common scenario where only nodes have features, we use $\mathbf{X} \in \mathbb{R}^{n \times d}$ to denote the node feature matrix for short, and denote the attributed graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$.

There are also some dynamic graphs and heterogeneous graphs whose definitions are given in Appendix B.2.

Most of the reviewed methods leverage GNNs as backbone encoders to transform the input raw node features $\mathbf{X}$ into compact node representations $\mathbf{H}$ by leveraging the rich underlying node connectivity, i.e., adjacency matrix $\mathbf{A}$, with

---

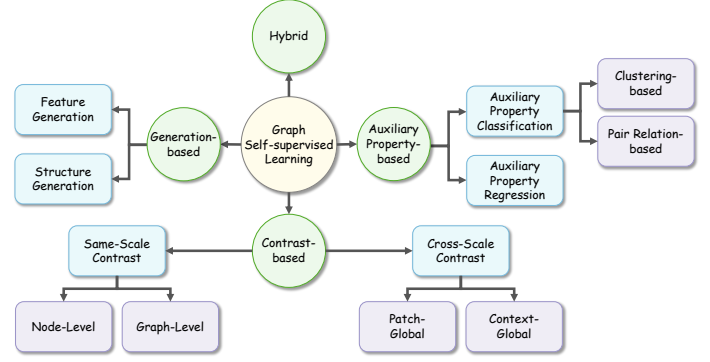2. A plain graph is an *unattributed, static, and homogeneous* graph.

Fig. 2: Categorization of graph SSL methods.

learnable parameters. Furthermore, readout functions $\mathcal{R}(\cdot)$ are often employed to generate a graph-level representation $\mathbf{h}_\mathcal{G}$ from node-level representations $\mathbf{H}$. The formulation of GNNs and readout functions are introduced in Appendix B.3. Besides, in Appendix B.4, we formulate the commonly used loss functions in this survey.

3 FRAMEWORK AND CATEGORIZATION

In this section, we provide a unified framework of graph SSL, and further categorize it from different perspectives, including pretext tasks, downstream tasks, and the combination of both (i.e., self-supervised training schemes).

3.1 Unified Framework and Mathematical Formulation of Graph Self-Supervised Learning

We construct an encoder-decoder framework to formalize graph SSL. The encoder f_θ (parameterized by θ) aims to learn a low-dimensional representation (a.k.a. embedding) $\mathbf{h_i} \in \mathbf{H}$ for each node v_i from graph $\mathcal{G}$. In general, the encoder f_θ can be GNNs [13], [33], [38] or other types of neural networks for graph learning [30], [31], [39]. The pretext decoder p_ϕ (parameterized by ϕ) takes $\mathbf{H}$ as its input for the pretext tasks. The architecture of p_ϕ depends on specific downstream tasks.

Under this framework, graph SSL can be formulated as:

$$\theta^*, \phi^* = \arg\min_{\theta, \phi} \mathcal{L}_{ssl}\left(f_\theta, p_\phi, \mathcal{D}\right), \qquad (1)$$

where $\mathcal{D}$ denotes the graph data distribution that satisfies $(\mathcal{V}, \mathcal{E}) \sim \mathcal{D}$ in an unlabeled graph $\mathcal{G}$, and $\mathcal{L}_{ssl}$ is the SSL loss function that regularizes the output of pretext decoder according to specific crafted pretext tasks.

By leveraging the trained graph encoder f_{θ^*}, the generated representations can then be used in various downstream tasks. Here we introduce a downstream decoder q_ψ (parameterized by ψ), and formulate the downstream task as a graph supervised learning task:

$$\theta^{**}, \psi^* = \arg\min_{\theta^*, \psi} \mathcal{L}_{sup}\left(f_{\theta^*}, q_\psi, \mathcal{G}, y\right), \qquad (2)$$

where y denotes the downstream task labels, and $\mathcal{L}_{sup}$ is the supervised loss that trains the model for downstream tasks.

In the following subsections, we specify four graph SSL variants based on Equation (1) in Section 3.2, three graph self-supervised training schemes in Section 3.3 by combining Equations (1) and (2) differently, and three types of downstream tasks based on Equation (2) in Section 3.4.
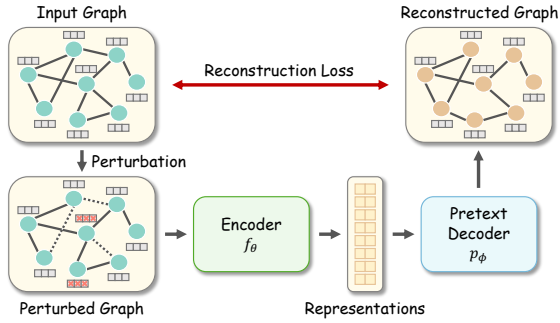
(a) Generation-based graph SSL methods. The model input is generated by a (optional) graph perturbation. In the pretext task, a generative decoder tries to recover the original graph from representation **H**, with a loss function aiming to minimize the difference between the reconstructed and original graphs.
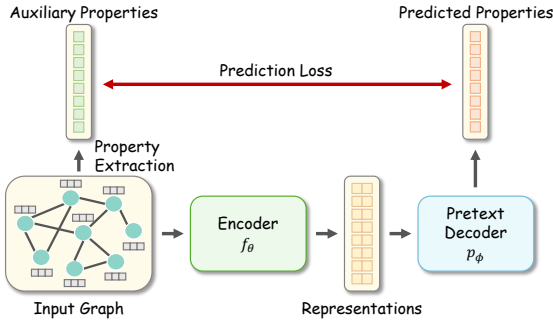
(b) Auxiliary property-based graph SSL methods. The auxiliary properties are extracted from graphs freely. A classification- or regression- based decoder aims to predict the extracted properties under the training of CE/MSE loss.
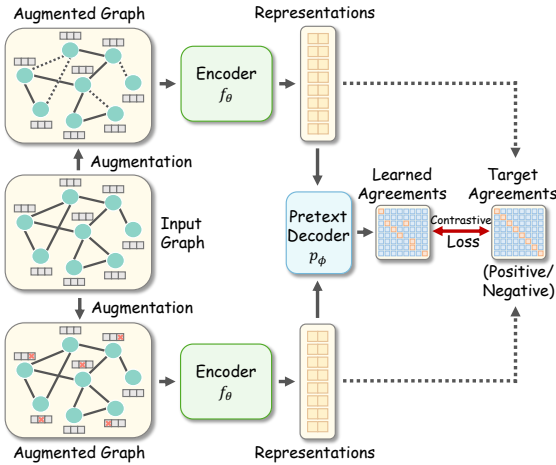
(c) Contrast-based graph SSL methods. Two different augmented views are constructed from the original graph. Then, the model is trained by maximizing the MI between two views. The MI estimator with (optional) projection often serves as the pretext decoder and SSL loss.
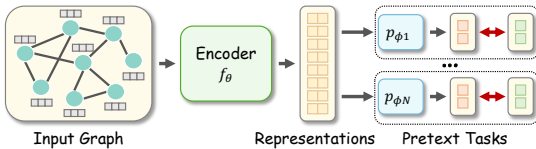
(d) Hybrid graph SSL methods. Multiple pretext tasks are designed to train the model together in a multi-task learning manner.

Fig. 3: Four categories of graph SSL.

3.2 Taxonomy of Graph Self-supervised Learning

Graph SSL can be divided into four types conceptually, including generation-based, auxiliary property-based, contrastive-based and hybrid methods, by leveraging different designs of pretext decoders and objective functions. The categorizations of these methods are briefly discussed below and shown in Fig. 2, and the concept map of each type of methods is given in Fig. 3.

Generation-based Methods form the pretext task as the graph data reconstruction from two perspectives: feature and structure. Specifically, they focus on the node/edge features or/and graph adjacency reconstructions. In such a case, Equation (1) can be further derived as:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \mathcal{L}_{ssl}\Big(p_\phi\big(f_\theta(\tilde{\mathcal{G}})\big), \mathcal{G}\Big), \qquad (3)$$

where $f_\theta(\cdot)$ and $p_\phi(\cdot)$ are graph encoder and pretext decoder. $\tilde{\mathcal{G}}$ denotes the graph data with perturbed node/edge features or/and adjacency matrix. For most of the generation-based approaches, the self-supervised objective function $\mathcal{L}_{ssl}$ is typically defined to measure the difference between the reconstructed and the original graph data. One of the representative approaches is GAE [32] which learns embeddings by rebuilding the graph adjacency matrix.

Auxiliary Property-based Methods enrich the supervision signals by capitalizing on a larger set of attributive and topological graph properties. In particular, for different crafted auxiliary properties, we further categorize these methods into two types: regression- and classification-based. Formally, they can be formulated as:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \mathcal{L}_{ssl}\Big(p_\phi\big(f_\theta(\mathcal{G})\big), c\Big), \qquad (4)$$

where c denotes the specific crafted auxiliary properties. For regression-based approaches, c can be localized or global graph properties, such as the node degree or distance to clusters within $\mathcal{G}$. For classification-based methods, on the other hand, the auxiliary properties are typically constructed as pseudo labels, such as the graph partition or cluster indices. Regarding to the objective function, $\mathcal{L}_{ssl}$ can be mean squared error (MSE) for regression-based and cross-entropy (CE) loss for classification-based methods. As a pioneering work, M3S [40] uses node clustering to construct pseudo labels that provide supervision signals.

Contrast-based Methods are usually developed based on the concept of mutual information (MI) maximization, where the estimated MI between augmented instances of the same object (*e.g.*, node, subgraph, and graph) is maximized. For contrastive-based graph SSL, Equation (1) is reformulated as:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \mathcal{L}_{ssl}\Bigg(p_\phi\Big(f_\theta(\tilde{\mathcal{G}}^{(1)}), f_\theta(\tilde{\mathcal{G}}^{(2)})\Big)\Bigg), \quad (5)$$

where $\tilde{\mathcal{G}}^{(1)}$ and $\tilde{\mathcal{G}}^{(2)}$ are two differently augmented instances of $\mathcal{G}$. In these methods, the pretext decoder p_ϕ indicates the discriminator that estimates the agreement between two instances (*e.g.*, the bilinear function or the dot product), and $\mathcal{L}_{ssl}$ denotes the contrastive loss. By combining them and optimizing $\mathcal{L}_{ssl}$, the pretext tasks aim to estimate and
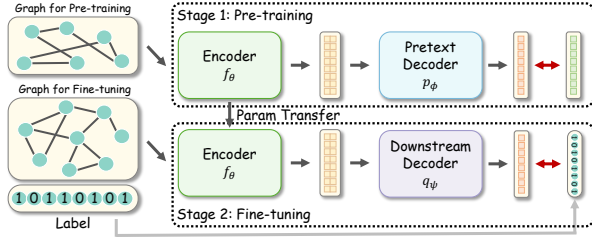
(a) Pre-training and Fine-tuning (PF). First, the encoder is pre-trained with pretext tasks in an unsupervised manner. Then, the pre-trained parameters are leveraged as the initial parameters in the fine-tuning phase where encoder is trained by downstream tasks independently.
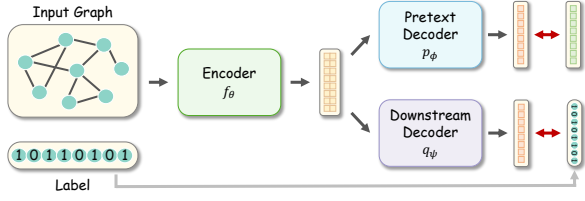
(b) Joint Learning (JL). The model is trained with pretext and downstream tasks in a multi-task learning manner.

(c) Unsupervised Representation Learning (URL). It first trains encoder with pretext tasks, and uses the fixed representations to learn the downstream decoder in Stage 2.

Fig. 4: Three types of learning schemes for SSL.

maximize the MI between positive pairs (e.g., augmented instances of the same object) and minimize the MI between negative samples (e.g., instances derived from different objects), which is implicitly included in $\mathcal{L}_{ssl}$. Representative works include cross-scale methods (e.g., DGI [13]) and same-scale methods (e.g., GraphCL [38] and GCC [15]).

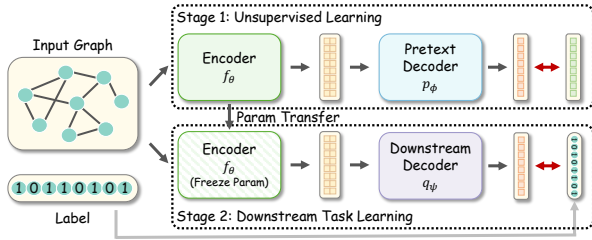**Hybrid Methods** take advantage of previous categories and consist of more than one pretext decoder and/or training objective. We formulate this branch of methods as the weighted or unweighted combination of two or more graph SSL schemes based on formulas from Equation (3) to (5). GMI [41], which jointly considers edge-level reconstruction and node-level contrast, is a typical hybrid method.

Discussion. Different graph SSL methods have different properties. Generation-based methods are simple to implement since the reconstruction task is easy to build, but sometimes recovering input data is memory-consuming for large-scale graphs. Auxiliary property-based methods enjoy the uncomplicated design of decoders and loss functions; however, the selection of helpful auxiliary properties often needs domain knowledge. Compared to other categories, contrast-based methods have more flexible designs and boarder applications. Nevertheless, the designs of contrastive frameworks, augmentation strategies, and loss functions usually rely on time-consuming empirical experiments. Hybrid

methods benefit from multiple pretext tasks, but a main challenge is how to design a joint learning framework to balance each component.

3.3 Taxonomy of Self-Supervised Training Schemes

According to the relationship among graph encoders, self-supervised pretext tasks, and downstream tasks, we investigate three types of graph self-supervised training schemes: Pre-training and Fine-tuning (PF), Joint Learning (JL), and Unsupervised Representation Learning (URL). Brief pipelines of them are given in Fig. 4.

Pre-training and Fine-tuning (PF). In PF scheme, the encoder f_θ is first pre-trained with pretext tasks on pre-training datasets, which can be viewed as an initialization for the encoder's parameters. After that, the pre-trained encoder $f_{\theta_{init}}$ is fine-tuned together on fine-tuning datasets (with labels) with a downstream decoder q_ψ under the supervision of specific downstream tasks. Note that the datasets for pre-training and fine-tuning could be the same or different. The formulation of PF scheme is defined as follows:

$$
\begin{aligned}
\theta^*, \phi^* &= \underset{\theta, \phi}{\arg\min}\, \mathcal{L}_{ssl}\left(f_\theta, p_\phi, \mathcal{D}\right), \\
\theta^{**}, \phi^* &= \underset{\theta^*, \psi}{\arg\min}\, \mathcal{L}_{sup}\left(p_{\theta^*}, q_\psi, \mathcal{G}, y\right).
\end{aligned}
\tag{6}
$$

Joint Learning (JL). In JL scheme, the encoder is jointly trained with the pretext and downstream tasks. The loss function consists of both the self-supervised and downstream task loss functions, where a trade-off hyper-parameter α controls the contribution of self-supervision term. This can be considered as a kind of multi-task learning where the pretext task is served as a regularization of the downstream task:

$$
\theta^*, \phi^*, \psi^* = \underset{\theta, \phi, \psi}{\arg\min}\left[\alpha\mathcal{L}_{ssl}\left(f_\theta, p_\phi, \mathcal{D}\right) + \mathcal{L}_{sup}\left(f_\theta, q_\psi, \mathcal{G}, y\right)\right].
\tag{7}
$$

Unsupervised Representation Learning (URL). The first stage of the URL scheme is similar to that of PF. The differences are: (1) In the second stage, the encoder's parameters are frozen (i.e., θ^*) when the model is trained with the downstream task; (2) The training of two stages is performed on the same dataset. The formulation of URL is defined as:

$$
\begin{aligned}
\theta^*, \phi^* &= \underset{\theta, \phi}{\arg\min}\, \mathcal{L}_{ssl}\left(f_\theta, p_\phi, \mathcal{D}\right), \\
\psi^* &= \underset{\psi}{\arg\min}\, \mathcal{L}_{sup}\left(f_{\theta^*}, q_\psi, \mathcal{G}, y\right).
\end{aligned}
\tag{8}
$$

Compared with other schemes, URL is more challenging since there is no supervision during the encoder training.

3.4 Taxonomy of Downstream Tasks

According to the scale of prediction target, we divide downstream tasks into node-, link-, and graph-level tasks. Specifically, node-level tasks aim to predict the property of nodes in graph(s) according to node representations. Link-level tasks infer the property of edges or pairs of nodes, where downstream decoders map the embeddings of two nodes into link-level predictions. Besides, graph-level tasks learn from a dataset with multiple graphs and forecast

(a) Graph Completion is a representative approach of feature generation-based graph SSL. The features of certain nodes are masked and then fed into the model, and the learning target is to reconstruct the masked features. An MSE loss is used to recover the features.

(b) The objective of Denoising Link Reconstruction is to rebuild the masked edges. A binary cross-entropy (BCE) loss is employed to train the model where existing edges are the positive samples and the unrelated node pairs are the negative samples. A negative sampling (Neg. Samp.) strategy is used to balance the classes.

Fig. 5: Examples of two categories of generation-based methods: Graph Completion and Denoising Link Reconstruction.

the property of each graph. Based on Equation (2), we provide the specific definitions of downstream decoders q_ψ, downstream objectives $\mathcal{L}_{sup}$, and downstream task labels y of three types of tasks, which are detailed in Appendix C.

4 GENERATION-BASED METHODS

The generation-based methods aim to reconstruct the input data and use the input data as their supervision signals. The origin of this category of methods can be traced back to Autoencoder [42] which learns to compress data vectors into low-dimensional representations with the encoder network and then try to rebuild the input vectors with the decoder network. Different from generic input data represented in vector formats, graph data are interconnected. As a result, generation-based graph SSL approaches often take the full graph or a subgraph as the model input, and reconstruct one of the components, i.e. feature or structure, individually. According to the objects of reconstruction, we divide these works into two sub-categories: (1) *feature generation* that learns to reconstruct the feature information of graphs, and (2) *structure generation* that learns to reconstruct the topological structure information of graphs. The pipelines of two example methods are given in Fig. 5, and a summary of the generation-based works is illustrated in Table 1.

4.1 Feature Generation

Feature generation approaches learn by recovering feature information from the perturbed or original graphs. Based on Equation (3), the feature generation approaches can be further formalized as:

$$\theta^*, \phi^* = \underset{\theta, \phi}{\arg\min} \, \mathcal{L}_{mse}\left(p_\phi\left(f_\theta\left(\tilde{\mathcal{G}}\right)\right), \hat{\mathbf{X}}\right), \qquad (9)$$

where $p_\phi(\cdot)$ is the decoder for feature regression (e.g., a fully connected network that maps the representations to reconstructed features), $\mathcal{L}_{mse}$ is the Mean Squared Error (MSE) loss function, and $\hat{\mathbf{X}}$ is a general expression of various kinds of feature matrices, *e.g.*, node feature matrix, edge feature matrix, or low-dimensional feature matrix.

To leverage the dependency between nodes, a representative branch of feature generation approaches follows the *masked feature regression* strategy, which is motivated by image inpainting in CV domain [20]. Specifically, the features of certain nodes/edges are masked with zero or

specific tokens in the pre-processing phase. Then, the model tries to recover the masked features according to the unmasked information. Graph Completion [17] is a representative method. It first masks certain nodes of the input graph by removing their features. Then, the learning objective is to predict the masked node features from the features of neighboring nodes with a GCN [1] encoder. We can consider Graph Completion as an implement of Equation (9) where $\hat{\mathbf{X}} = \mathbf{X}$ and $\tilde{\mathcal{G}} = (\mathbf{A}, \tilde{\mathbf{X}})$. Similarly, AttributeMask [43] aims to reconstruct the dense feature matrix processed by Principle Component Analysis (PCA) [52] ($\hat{\mathbf{X}} = PCA(\mathbf{X})$) instead of the raw features due to the difficulty of rebuilding high-dimensional and sparse features. AttrMasking [16] rebuilds not only node attributes but also the edge one, which can be written as $\hat{\mathbf{X}} = [\mathbf{X}, \mathbf{X}_{\mathbf{edge}}]$.

Another branch of methods aims to generate features from noisy features. Inspired by denoising autoencoder [53], MGAE [44] recovers raw features from noisy input features with each GNN layer. Here we also denote $\tilde{\mathcal{G}} = (\mathbf{A}, \tilde{\mathbf{X}})$ but here $\tilde{\mathbf{X}}$ is corrupted with random noise. Proposed in [45], Corrupted Features Reconstruction and Corrupted Embeddings Reconstruction aim to reconstruct raw features and hidden embeddings from corrupted features.

Besides, directly rebuilding features from the clean data is also an available solution. GALA [46] trains a Laplacian smoothing-sharpening graph autoencoder model with the objective that rebuilds the raw feature matrix according to the clean input graph. Similarly, autoencoding [45] reconstructs the raw features from clean inputs. For these two methods, we can formalize that $\tilde{\mathcal{G}} = (\mathbf{A}, \mathbf{X})$ and $\hat{\mathbf{X}} = \mathbf{X}$.

4.2 Structure Generation

Different from the feature generation approaches that rebuild the feature information, structure generation approaches learn by recovering the structural information. In most cases, the objective is to reconstruct the adjacency matrix, since the adjacency matrix can briefly represent the topological structure of graphs. Based on Equation (3), the structure generation methods can be formalized as follows:

$$\theta^*, \phi^* = \underset{\theta, \phi}{\arg\min} \, \mathcal{L}_{ssl}\left(p_\phi\left(f_\theta\left(\tilde{\mathcal{G}}\right)\right), \mathbf{A}\right), \qquad (10)$$

where $p_\phi(\cdot)$ is a decoder for structure reconstruction, and $\mathbf{A}$ is the (full or partial) adjacency matrix.

TABLE 1: Main characteristics of generation-based graph SSL approaches. "FG" and "SG" mean "Feature Generation" and "Structure Generation", respectively. Missing values ("-") in Input Data Perturbation indicate that the method takes the original graph data as input.

Approach	Pretext Task Category	Downstream Task Level	Training Scheme	Data Type of Graph	Input Data Perturbation	Generation Target
Graph Completion [17]	FG	Node	PF/JL	Attributed	Feature Masking	Node Feature
AttributeMask [43]	FG	Node	PF/JL	Attributed	Feature Masking	PCA Node Feature
AttrMasking [16]	FG	Node	PF	Attributed	Feature Masking	Node/Edge Feature
MGAE [44]	FG	Node	JL	Attributed	Feature Noising	Node Feature
Corrupted Features Reconstruction [45]	FG	Node	JL	Attributed	Feature Noising	Node Feature
Corrupted Embeddings Reconstruction [45]	FG	Node	JL	Attributed	Embedding Noising	Node Embedding
GALA [46]	FG	Node/Link	JL	Attributed	-	Node Feature
Autoencoding [45]	FG	Node	JL	Attributed	-	Node Feature
GAE/VGAE [32]	SG	Link	URL	Attributed	-	Adjacency Matrix
SIG-VAE [47]	SG	Node/Link	URL	Plain/Attributed	-	Adjacency Matrix
ARGA/ARVGA [48]	SG	Node/Link	URL	Attributed	-	Adjacency Matrix
SuperGAT [49]	SG	Node	JL	Attributed	-	Partial Edge
Denoising Link Reconstruction [50]	SG	Node/Link/Graph	PF	Attributed	Edge Masking	Masked Edge
EdgeMask [43]	SG	Node	PF/JL	Attributed	Edge Masking	Masked Edge
Zhu et al. [51]	SG	Node	PF	Attributed	Feature Masking/Edge Masking	Partial Edge

GAE [32] is the simplest instance of the structure generation method. In GAE, a GCN-based encoder first generates node embeddings $\mathbf{H}$ from the original graph ($\tilde{\mathcal{G}} = \mathcal{G}$). Then, an inner production function with sigmoid activation serves as its decoder to recover the adjacency matrix from $\mathbf{H}$. Since adjacency matrix $\mathbf{A}$ is usually binary and sparse, a BCE loss function is employed to maximize the similarity between the recovered adjacency matrix and the original one, where positive and negative samples are the existing edges ($\mathbf{A}_{i,j} = 1$) and unconnected node pairs ($\mathbf{A}_{i,j} = 0$), respectively. To avoid the imbalanced training sample problem caused by extremely sparse adjacency, two strategies can be used to prevent trivial solution: (1) re-weighting the terms with $\mathbf{A}_{i,j} = 1$; or (2) sub-sampling terms with $\mathbf{A}_{i,j} = 0$.

As a classic learning paradigm, GAE has a series of derivative works. VGAE [32] further integrates the idea of variational autoencoder [54] into GAE. It employs an inference model-based encoder that estimates the mean and deviation with two parallel output layers and uses Kullback-Leibler divergence between the prior distribution and the estimated distribution. Following VGAE, SIG-VAE [47] considers hierarchical variational inference to learn more generative representations for graph data. ARGA/ARVGA [48] regularizes the GAE/VGAE model with generative adversarial networks (GANs) [55]. Specifically, a discriminator is trained to distinguish the fake and real data, which forces the distribution of latent embeddings closer to the Gaussian prior. SuperGAT [49] further extends this idea to every layers in the encoder. Concretely, it rebuilds the adjacency matrix from the latent representations of every layer in the encoder.

Instead of rebuilding the full graph, another solution is to reconstruct the masked edges. Denoising Link Reconstruction [50] randomly drops existing edges to obtain the perturbed graph $\tilde{\mathcal{G}}$. Then, the model aims to recover the discarded connections with a pairwise similarity-based decoder trained by a BCE loss. EdgeMask [43] also has a similar perturbation strategy, where a non-parametric MAE function minimizes the difference between the embeddings of two connected nodes. Zhu et al. [51] apply two perturbing strategies, i.e. Randomly Removing Links and Randomly Covering Features, to the input graph ($\tilde{\mathcal{G}} = (\tilde{\mathbf{A}}, \tilde{\mathbf{X}})$), while its target is to recover the masked link by a decoder.

Discussion. Due to the different learning targets, two branches of generation-based methods have distinct designs of the decoder and loss functions. The learned representations by structure generation usually contain more node pair-level information since structure generation focuses on edge reconstruction; by contrary, feature generation methods often capture node-level knowledge.

5 AUXILIARY PROPERTY-BASED METHODS

The auxiliary property-based methods acquire supervision signals from the node-, link- and graph- level properties which can be obtained from the graph data freely. These methods have a similar training paradigm with supervised learning since both of them learn with "sample-label" pairs. Their difference lies in how the label is obtained: In supervised learning, the manual label is human-annotated which often needs expensive costs; in auxiliary property-based SSL, the pseudo label is self-generated automatically without any cost.

Following the general taxonomy of supervised learning, we divide auxiliary property-based methods into two subcategories: (1) *auxiliary property classification* which leverages classification-based pretext tasks to train the encoder and (2) *auxiliary property regression* which performs SSL via regression-based pretext tasks. Fig. 6 provides the pipelines of them, and Table 2 summarizes the auxiliary property-based methods.

5.1 Auxiliary Property Classification

Borrowing the training paradigm from supervised classification tasks, the methods of auxiliary property classification create discrete pseudo labels automatically, build a classifier as the pretext decoder, and use a cross entropy (CE) loss $\mathcal{L}_{ce}$ to train the model. Originated from Equation (4), we provide the formalization of this branch of methods as:

$$\theta^*, \phi^* = \underset{\theta, \phi}{\arg\min} \, \mathcal{L}_{ce}\Big(p_\phi\big(f_\theta(\mathcal{G})\big), c\Big), \qquad (11)$$

where p_ϕ is the neural network classifier-based decoder which outputs a k-dimensional probability vector (k is the number of classes), and $c \in \mathcal{C} = \{c_1, \cdots, c_k\}$ is the corresponding pseudo label which belongs to a discrete and finite label set $\mathcal{C}$. According to the definition of pseudo label set $\mathcal{C}$, we further construct two sub-categories under auxiliary property classification, i.e., clustering-based and pair relation-based methods.
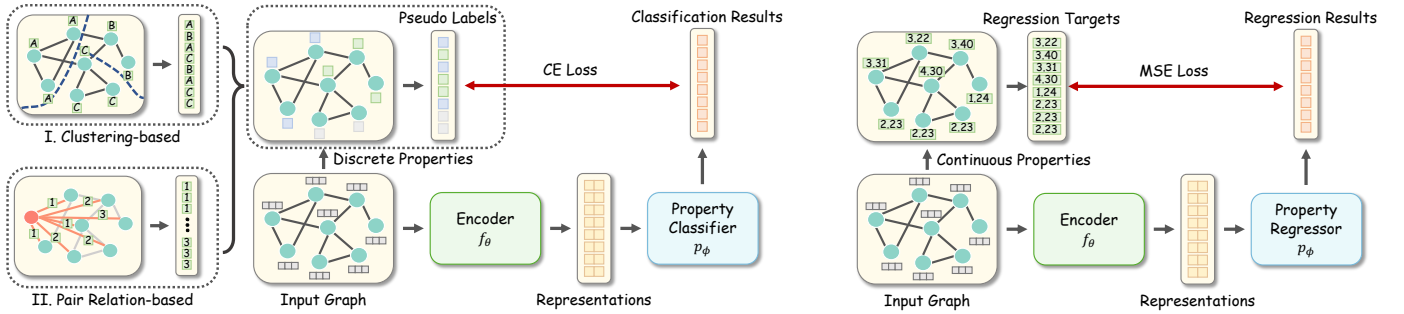
(a) Auxiliary property classification methods extract discrete properties as pseudo labels, and the pretext decoder is used to predict the classification results. A CE loss function is used to train the models. Two types of properties (clustering-based and pair relation-based) can be used to define the pseudo labels.

(b) Auxiliary property regression methods aim to predict the continuous auxiliary properties with the decoder, where models are trained by the MSE loss.

Fig. 6: Two categories of auxiliary property-based graph SSL.

TABLE 2: Main characteristics of auxiliary property-based graph SSL approaches. "CAPC", "PAPC" and "APR" mean clustering-based auxiliary property classification, pair relation-based auxiliary property classification and auxiliary property regression, respectively.

Approach	Pretext Task Category	Downstream Task Level	Training Scheme	Data Type of Graph	Property Level	Mapping Function
Node Clustering [17]	CAPC	Node	PF/JL	Attributed	Node	Feature-based Clustering
M3S [40]	CAPC	Node	JL	Attributed	Node	Feature-based Clustering
Graph Partitioning [17]	CAPC	Node	PF/JL	Attributed	Node	Structure-based Clustering
Cluster Preserving [50]	CAPC	Node/Link/Graph	PF	Attributed	Node	Structure-based Clustering
CAGNN [56]	CAPC	Node	URL	Attributed	Node	Feature-based Clustering with Structural Refinement
S^{2}GRL [57]	PAPC	Node/Link	URL	Attributed	Node Pair	Shortest Distance Function
PairwiseDistance [43]	PAPC	Node	PF/JL	Attributed	Node Pair	Shortest Distance Function
Centrality Score Ranking [50]	PAPC	Node/Link/Graph	PF	Attributed	Node Pair	Centrality Scores Comparison
NodeProperty [43]	APR	Node	PF/JL	Attributed	Node	Degree Calculation
Distance2Cluster [43]	APR	Node	PF/JL	Attributed	Node Pair	Distance to Cluster Center
PairwiseAttrSim [43]	APR	Node	PF/JL	Attributed	Node Pair	Cosine Similarity of Feature
SimP-GCN [58]	APR	Node	JL	Attributed	Node Pair	Cosine Similarity of Feature

5.1.1 Clustering-based Methods

A promising way to construct pseudo label is to divide nodes into different clusters according to their attributive or structural characteristics. To achieve that, a mapping function $\Omega : \mathcal{V} \to \mathcal{C}$ is introduced to acquire the pseudo label for each node, which is built on specific unsupervised clustering/partitioning algorithms [59], [60], [61], [62]. Then, the learning objective is to classify each node into its corresponding cluster. Following Equation (11), the learning objective is refined as:

$$\theta^*, \phi^* = \arg\min_{\theta, \phi} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{L}_{ce}\Big(p_\phi([f_\theta(\mathcal{G})]_{v_i}), \Omega(v_i)\Big), \quad (12)$$

where $[\cdot]_{v_i}$ is the picking function that extracts the representation of v_i.

Node Clustering [17] is a representative approach that utilizes attributive information to generate pseudo labels. Specifically, it leverages a feature-based clustering algorithm (which is an instance of Ω) taking $\mathbf{X}$ as input to divide node set into k clusters, and each cluster indicates a pseudo label for classification. The intuition behind Node Clustering is that nodes with similar features tend to have consistent semantic properties. M3S [40] introduces a multi-stage self-training mechanism for SSL using DeepCluster [61] algorithm. In each stage, it first runs K-means clustering on node

embedding $\mathbf{H}$. After that, an alignment is executed to map each cluster to a class label. Finally, the unlabeled nodes with high confidence are given the corresponding (pseudo) labels and used to train the model. In M3S, $\mathcal{C}$ is borrowed from the manual label set $\mathcal{Y}$, and Ω is composed of the K-means and alignment algorithms.

In addition to feature-based clustering, Graph Partitioning [17] divides the nodes according to the structural characteristics of nodes. Concretely, it groups nodes into multiple subsets by minimizing the connections across subsets [59], defining Ω as the graph partitioning algorithm. Cluster Preserving [50] first leverages graph clustering algorithm [63] to acquire non-overlapping clusters, and then calculates the representation of each cluster via an attention-based aggregator. After that, a vector representing the similarities between each node and the cluster representations is assigned as the soft pseudo label for each node. Besides, CAGNN [56] first runs feature-based clusters to generate pseudo labels and then refines the clusters by minimizing inter-cluster edges, which absorbs the advantages of both attributive and structural clustering algorithms.

5.1.2 Pair Relation-based Methods

Apart from the clustering and graph properties, an alternative supervision signal is the relationship between each pair of nodes within a graph. In these methods, the input of the

decoder is not a single node or graph but a pair of nodes. A mapping function $\Omega : \mathcal{V} \times \mathcal{V} \to \mathcal{C}$ is utilized to define the pseudo label according to pair-wise contextual relationship. We write the objective function as:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \frac{1}{|\mathcal{P}|} \sum_{v_i,v_j \in \mathcal{P}} \mathcal{L}_{ce}\Big(p_\phi([f_\theta(\mathcal{G})]_{v_i,v_j}), \Omega(v_i, v_j)\Big),$$
(13)

where $\mathcal{P} \subseteq \mathcal{V} \times \mathcal{V}$ is the node pair set defined by specific pretext tasks, and $[\cdot]_{v_i,v_j}$ is the picking function that extracts and concatenates the node representations of v_i and v_j.

Some approaches regard the distance between two nodes as the auxiliary property. For instance, S²GRL [57] learns by predicting the shortest path between two nodes. Specifically, the label for a pair of nodes is defined as the shortest distance between them. Formally, we can write the mapping function as $\Omega(v_i, v_j) = \mathrm{dist}(v_i, v_j)$. The decoder is built to measure the interaction between pairs of nodes, which is defined as an element-wise distance between two embedding vectors. The node pair set $\mathcal{P}$ collects all possible node pairs including the combination of all nodes with their 1 to K hops neighborhoods. PairwiseDistance [43] has a very similar learning target and decoder with S²GRL, but introduces an upper bound of distance, which can be represented as $\Omega(v_i, v_j) = \max(\mathrm{dist}(v_i, v_j), 4)$.

Centrality Score Ranking [50] presents a pretext task that predicts the relative order of centrality scores between a pair of nodes. For each node pair (v_i, v_j), it first calculates four types of centrality scores s_i, s_j (eigencentrality, betweenness, closeness, and subgraph centrality), and then creates its pseudo label by comparing the value of s_i and s_j. We formalize the mapping function as: $\Omega(v_i, v_j) = \mathbb{I}(s_i > s_j)$, where $\mathbb{I}(\cdot)$ is the identity function.

5.2 Auxiliary Property Regression

Auxiliary property regression approaches construct the pretext tasks on predicting extensive numerical properties of graphs. Compared to auxiliary property classification, the most significant difference is that the auxiliary properties are continuous values within a certain range instead of discrete pseudo labels in a limited set. We refine Equation (4) into a regression version:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \mathcal{L}_{mse}\left(p_\phi\big(f_\theta(\mathcal{G})\big), c\right),$$
(14)

where $\mathcal{L}_{mse}$ is the MSE loss function for regression, and $c \in \mathbb{R}$ is a continuous property value.

NodeProperty [43] is a node-level pretext task that predicts the property for each node. The available choices of node properties include their degree, local node importance, and local clustering coefficient. Taking node degree as an example, the objective function is illustrated as follows:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{L}_{mse}\left(p_\phi([f_\theta(\mathcal{G})]_{v_i}), \Omega(v_i)\right),$$
(15)

where $\Omega(v_i) = \sum_{j=1}^{n} \mathbf{A}_{ij}$ is the mapping function that calculates the degree of node v_i. Distance2Cluster [43] aims to regress the distances from each node to predefined graph

clusters. Specifically, it first partitions the graph into several clusters with the METIS algorithm [64] and defines the node with the highest degree within each cluster as its cluster center. Then, the target is to predict the distances between each node and all cluster centers.

Another type of methods take the pair-wise property as their regression targets. For instance, the target of PairwiseAttrSim [43] is to predict the feature similarity of two nodes according to their embeddings. We formalize its objective function as follows:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \frac{1}{|\mathcal{P}|} \sum_{v_i,v_j \in \mathcal{P}} \mathcal{L}_{mse}\left(p_\phi([f_\theta(\mathcal{G})]_{v_i,v_j}), \Omega(v_i, v_j)\right),$$
(16)

where mapping function $\Omega(v_i, v_j) = \mathrm{cosine}(\mathbf{x}_i, \mathbf{x}_j)$ is the cosine similarity of raw features. In PairwiseAttrSim, the node pairs with the highest similarity and dissimilarity are selected to form the node pair set $\mathcal{P}$. Similar to PairwiseAttrSim, SimP-GCN [58] also considers predicting the cosine similarity of raw features as a self-supervised regularization for downstream tasks.

Discussion. As we can observe, the auxiliary property classification methods are more diverse than the regression methods, since the discrete pseudo labels can be acquired by various algorithms. In future works, more continuous properties are expected to be leveraged for regression methods.

6 CONTRAST-BASED METHODS

The contrast-based methods are built on the idea of mutual information (MI) maximization [65], which learns by predicting the agreement between two augmented instances. Specifically, the MI between graph instances with the similar semantic information (i.e., positive samples) is maximized, while the MI between those with unrelated information (i.e., negative samples) is minimized. Similar to the visual domain [24], [66], there exist various graph augmentations and contrastive pretext tasks on multiple granularities to enrich the supervision signals.

Following the taxonomy of contrast-based graph SSL defined in Section 3.2.3, we survey this branch of methods from three perspectives: (1) *Graph augmentations* that generate various graph instances; (2) *Graph contrastive learning* which forms various contrastive pretext tasks on the non-Euclidean space; (3) *Mutual information estimation* that measures the MI between instances and forms the contrastive learning objective together with specific pretext tasks.

6.1 Graph Augmentations

Recent success of contrastive learning on the visual domain relies heavily on well-crafted image augmentations, which reveals that data augmentations benefit the model to explore richer underlying semantic information by making pretext tasks more challenging to solve [67]. However, due to the nature of graph-structured data, it is difficult to apply the augmentations from the Euclidean to the non-Euclidean space directly. Motivated by image augmentations (*e.g.*, image cutout and cropping [24]), existing graph augmentations can be categorized into three types: *attributive-based*, *topological-based*, and the combination of both (i.e., *hybrid*
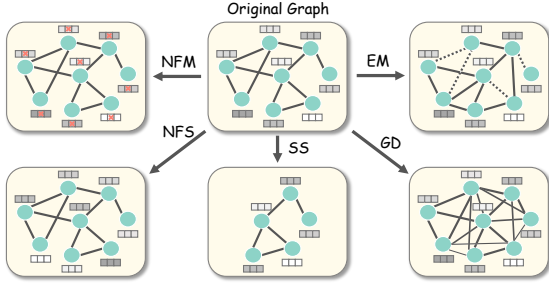
Fig. 7: Brief examples of five types of common graph augmentations, including Node Feature Masking (NFM), Node Feature Shuffle (NFS), Edge Modification (EM), Graph Diffusion (GD), and Subgraph Sampling (SS).

augmentations). The examples of five representative augmentation strategies are demonstrated in Fig. 7. Formally, given a graph $\mathcal{G}$, we define the i-th augmented graph instance as $\tilde{\mathcal{G}}^{(i)} = t_i(\mathcal{G})$, where $t_i \sim \tau$ is a selected graph augmentation and τ is a set of available augmentations.

6.1.1 Attributive augmentations

This category of augmentations is typically placed on node attributes. Given $\mathcal{G} = (\mathbf{A}, \mathbf{X})$, the augmented graph is represented as:

$$\tilde{\mathcal{G}}^{(i)} = (\mathbf{A}, \tilde{\mathbf{X}}^{(i)}) = (\mathbf{A}, t_i(\mathbf{X})), \quad (17)$$

where $t_i(\cdot)$ is placed on the node feature matrix only, and $\tilde{\mathbf{X}}^{(i)}$ denotes the augmented node features. Specifically, attributive augmentations have two variants. The first type is **Node feature masking (NFM)** [16], [33], [38], [43], [68], which randomly masks the features of a portion of nodes within the given graph. In particular, we can completely (i.e., row-wisely) mask selected feature vectors with zeros [16], [43], or partially (i.e., column-wisely) mask a number of selected feature channels with zeros [33], [68]. We formulate the node feature masking operation as:

$$t_i(\mathbf{X}) = \mathbf{M} \circ \mathbf{X}, \quad (18)$$

where $\mathbf{M}$ is the masking matrix with the same shape of $\mathbf{X}$, and $\circ$ denotes the Hadamard product. For a given masking matrix, its elements have been initialized to one and masking entries are assigned to zero. In addition to randomly sampling a masking matrix $\mathbf{M}$, we can also calculate it adaptively [69], [70]. For example, GCA [69] keeps important node features unmasked while assigning a higher masking probability for those unimportant nodes, where the importance is measured by node centrality.

On the other hand, instead of masking a part of the feature matrix, **node feature shuffle (NFS)** [13], [71], [72] partially and row-wisely perturbs the node feature matrix. In other words, several nodes in the augmented graph are placed to other positions when compared with the input graph, as formulated below:

$$t_i(\mathbf{X}) = [\mathbf{X}]_{\tilde{\mathcal{V}}}, \quad (19)$$

where $[\cdot]_{v_i}$ is a picking function that indexes the feature vector of v_i from the node feature matrix, and $\tilde{\mathcal{V}}$ denotes the partially shuffled node set.

6.1.2 Topological augmentations

Graph augmentations from the structural perspectives mainly work on the graph adjacency matrix, which is formulated as follows:

$$\tilde{\mathcal{G}}^{(i)} = (\tilde{\mathbf{A}}^{(i)}, \mathbf{X}) = (t_i(\mathbf{A}), \mathbf{X}), \quad (20)$$

where $t_i(\cdot)$ is typically placed on the graph adjacency matrix. For this branch of methods, **edge modification (EM)** [9], [38], [51], [68], [73], [74] is one of the most common approaches, which partially perturbs the given graph adjacency by randomly dropping and inserting a portion of edges. We define this process as follows:

$$t_i(\mathbf{A}) = \mathbf{M_1} \circ \mathbf{A} + \mathbf{M_2} \circ (1 - \mathbf{A}), \quad (21)$$

where $\mathbf{M}_1$ and $\mathbf{M}_2$ are edge dropping and insertion matrices. Specifically, $\mathbf{M}_1$ and $\mathbf{M}_2$ are generated by randomly masking a portion of elements with the value equal to one in $\mathbf{A}$ and $(1-\mathbf{A})$. Similar to node feature masking, $\mathbf{M}_1$ and $\mathbf{M}_2$ can also be calculated adaptively [69]. Furthermore, edge modification matrices can be generated based on adversarial learning [18], [75], which increases the robustness of learned representations.

Different from the edge modification, **graph diffusion (GD)** [76] is another type of structural augmentations [14], [68], which injects the global topological information to the given graph adjacency by connecting nodes with their indirectly connected neighbors with calculated weights:

$$t_i(\mathbf{A}) = \sum_{k=0}^{\infty} \Theta_k \mathbf{T}^k, \quad (22)$$

where Θ and $\mathbf{T}$ are weighting coefficient and transition matrix, respectively. Specifically, the above diffusion formula has two instantiations [14]. Let $\Theta_k = \frac{e^{-\iota} \iota^k}{k!}$ and $\mathbf{T} = \mathbf{A}\mathbf{D}^{-1}$, we have the heat kernel-based graph diffusion:

$$t_i(\mathbf{A}) = \exp(\iota \mathbf{A}\mathbf{D}^{-1} - \iota), \quad (23)$$

where ι denotes the diffusion time. Similarly, the Personalized PageRank-based graph diffusion is defined below by letting $\Theta_k = \beta(1-\beta)^k$ and $\mathbf{T} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$:

$$t_i(\mathbf{A}) = \beta(\mathbf{I} - (1-\beta)\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}), \quad (24)$$

where β denotes the tunable teleport probability.

6.1.3 Hybrid augmentations

It is worth noting that a given graph augmentation may involve not only the attributive but also the topological augmentations simultaneously, where we define it as the hybrid augmentation and formulate as:

$$\tilde{\mathcal{G}}^{(i)} = (\tilde{\mathbf{A}}^{(i)}, \tilde{\mathbf{X}}^{(i)}) = (t_i(\mathbf{A}, \mathbf{X})). \quad (25)$$

In such a case, the augmentation $t_i(\cdot)$ is placed on both the node feature and graph adjacency matrices. **Subgraph sampling (SS)** [14], [16], [74], [77] is a typical hybrid graph augmentation which is similar to image cropping. Specifically, it samples a portion of nodes and their underlying linkages as augmented graph instances:

$$t_i(\mathbf{A}, \mathbf{X}) = [(\mathbf{A}, \mathbf{X})]_{\mathcal{V}' \in \mathcal{V}}, \quad (26)$$

where $\mathcal{V}'$ denotes a subset of $\mathcal{V}$, and $[\cdot]_{\mathcal{V}'}$ is a picking function that indexes the node feature and adjacency matrices of the subgraph with node set $\mathcal{V}'$. Regarding the generation of $\mathcal{V}'$, several approaches have been proposed, such as uniform sampling [74], random walk-based sampling [15], and top-k importance-based sampling [77].

Apart from the subgraph sampling, most of graph contrastive methods heavily rely on hybrid augmentations by combining the aforementioned strategies. For example, GRACE [33] applies the edge dropping and node feature masking, while MVGRL [14] adopts the graph diffusion and subgraph sampling to generate different contrastive views.

6.2　Graph Contrastive Learning

As contrastive learning aims to maximize the MI between instances with similar semantic information, various pretext tasks can be constructed to enrich the supervision signals from such information. Regarding the formulation of pretext decoder $p_\phi(\cdot)$ in Equation (5), we classify existing works into two mainstreams: *same-scale* and *cross-scale* contrastive learning. The former branch of methods discriminates graph instances in an equal scale (*e.g.*, node versus node), while the second type of methods places the contrasting across multiple granularities (*e.g.*, node versus graph). Fig. 8 and Table 3 provide the pipelines and summaries of contrast-based methods, respectively.

6.2.1　Same-Scale Contrast

According to the scale for contrast, we further divide the same-scale contrastive learning approaches into two subtypes: node-level and graph-level.

6.2.1.1　*Node-Level Same-Scale Contrast*: Early methods [30], [31], [78], [79] under this category are mainly to learn node-level representations and built on the idea that nodes with similar contextual information should share the similar representations. In other word, these methods are trying to pull the representation of a node closer to its contextual neighborhood without relying on complex graph augmentations. We formulate them as below:

$$\theta^* = \arg\min_\theta \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{L}_{con}\big(p\big([f_\theta(\mathbf{A}, \mathbf{X})]_{v_i}, [f_\theta(\mathbf{A}, \mathbf{X})]_{v_c}\big)\big),$$
(27)

where v_c denotes the contextual node of v_i, for example, a neighboring node in a random walk starting from v_i. In those methods, the pretext discriminator (i.e., decoder) is typically the dot product and thus we omit its parameter in equation. Specifically, DeepWalk [30] introduces a random walk (RW)-based approach to extract the contextual information around a selected node in an unattributed graph. It maximizes the co-occurrence (i.e., MI measured by the binary classifier) of nodes within the same walk as in the Skip-Gram model [26], [27]. Similarly, node2vec [31] adopts biased RWs to explore richer node contextual information and yields a better performance. GraphSAGE [78], on the other hand, extends aforementioned two methods to attributed graphs, and proposes a novel GNN to calculate node embedding in an inductive manner, which applies RW as its internal sampling strategy as well. On heterogeneous graphs, SELAR [80] samples meta-paths to

capture the contextual information. It consists of a primary link prediction task and several meta-paths prediction auxiliary tasks to enforce nodes within the same meta-path to share closer semantic information.

Different from the aforementioned approaches, modern node-level same-scale contrastive methods are exploring richer underlying semantic information via various graph augmentations, instead of limiting on subgraph sampling:

$$\theta^*, \phi^* = \arg\min_{\theta, \phi} \mathcal{L}_{con}\Big(p_\phi\big(f_\theta(\tilde{\mathbf{A}}^{(1)}, \tilde{\mathbf{X}}^{(1)}), f_\theta(\tilde{\mathbf{A}}^{(2)}, \tilde{\mathbf{X}}^{(2)})\big)\Big),$$
(28)

where $\tilde{\mathbf{A}}^{(1)}$ and $\tilde{\mathbf{A}}^{(2)}$ are two augmented graph adjacency matrices. Similarly, $\tilde{\mathbf{X}}^{(1)}$ and $\tilde{\mathbf{X}}^{(2)}$ are two node feature matrices under different augmentations. The discriminator $p_\phi(\cdot)$ in above equation can be parametric with Φ (*e.g.*, bilinear transformation) or not (*e.g.*, cosine similarity where $\Phi = \emptyset$). In those methods, most of them deal with attributed graphs: GRACE [33] adopts two graph augmentation strategies, namely node feature masking and edge dropping, to generate two contrastive views, which then pulls the representations of the same nodes closer between two graph views while pushing the rest of nodes away (i.e., intra- and inter-view negatives). Based on this framework, GCA [69] further introduces an adaptive augmentation for graph-structured data based on underlying graph properties, which results in a more competitive performance. Differently, GROC [18] proposes an adversarial augmentation on graph linkages to increase the robustness of learned node representations. Because of the success of SimCLR [24] in the visual domain, GraphCL(N) [81] [3] further extends this idea to graph-structured data, which relies on the node feature masking and edge modification to generate two contrastive views, and then the MI between two target nodes within different views is maximized. CGPN [82] introduces Poisson learning to node-level contrastive learning, which benefits node classification task under extremely limited labeled data. On plain graphs, GCC [15] utilizes RW as augmentations to extract the contextual information of a node, which then contrasts the representation of it with its counterparts by leveraging the contrastive framework of MoCo [23]. On the other hand, HeCo [83] is contrasting on heterogeneous graphs, where two contrastive views are generated from two perspectives, i.e., network schema and meta-path, while the encoder is trained by maximizing the MI between the embeddings of the same node in two views.

Apart from those methods relying on carefully-crafted negative samples, approaches like BGRL [84] propose to contrast on graph instances themselves and thus alleviate the reliance on deliberately designed negative sampling strategies. BGRL takes the advantage of knowledge distillation in BYOL [25], where a momentum-driven Siamese architecture has been introduced to guide the extraction of supervision signals. Specifically, it uses node feature masking and edge modification as augmentations, and the objective of BGRL is the same as in BYOL where the MI between node representations from online and target networks is maximized. SelfGNN [85] adopts the same technique while

---

3. The approaches proposed in [81] and [38] have the same name "GraphCL". For distinction, we denote the node-level approach [81] as GraphCL(N) and the graph-level approach [38] as GraphCL(G).

(a) In same-scale contrast, different views are generated with various graph augmentations based on the input graph at first. Then, the contrast can be placed on node or graph scales.

(b) In cross-scale contrast, the augmented graph views are generated firstly. Then, the cross-scale contrast aims to discriminate patch- or context- level embeddings with global representations.
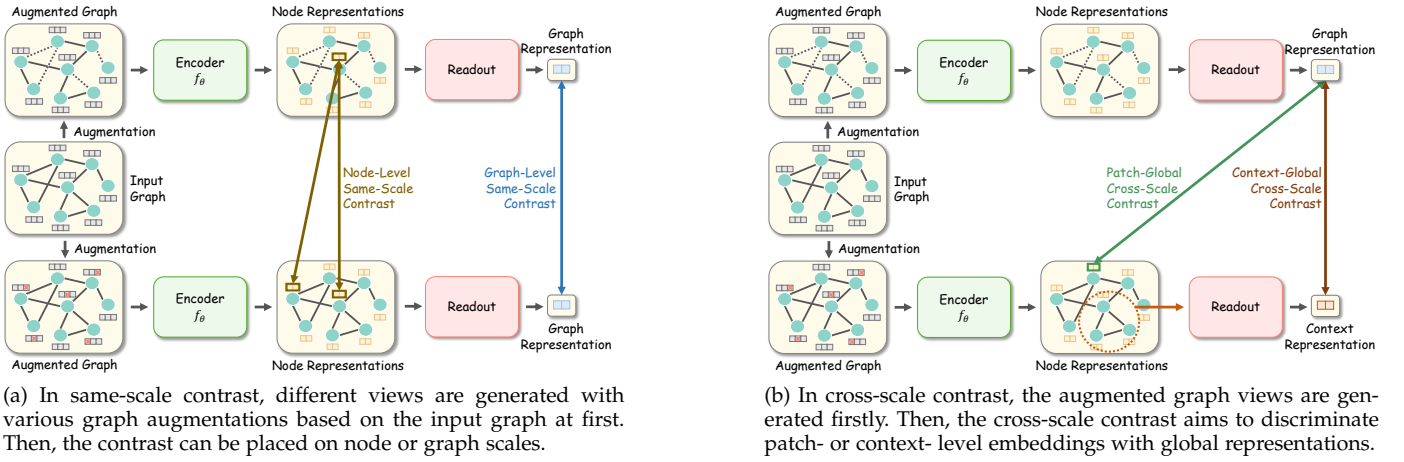
Fig. 8: Two categories of contrast-based graph SSL.

the difference is that SelfGNN uses other graph augmentations, such as graph diffusion [76], node feature split, standardization, and pasting. Apart from BYOL, Barlow Twins [86] is another similar yet powerful method without using negative samples to prevent the model from collapsing. G-BT [87] extends the redundancy-reduction principle for graph data analytics, where the optimization objective is to minimize the dissimilarity between the identity and cross-correlation metrics generated via node embeddings of two augmented graph views. MERIT [68], on the other hand, proposes to combine the advantages of Siamese knowledge distillation and conventional graph contrastive learning. It leverages a self-distillation framework in SimSiam [88] while introducing extra node-level negatives to further exploit the underlying semantic information and enrich the supervision signals.

6.2.1.2 *Graph-Level Same-Scale Contrast*: For graph-level representation learning under same-scale contrasting, the discrimination is typically placed on graph representations:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \mathcal{L}_{con}\Big(p_\phi\big(\tilde{\mathbf{g}}^{(1)}, \tilde{\mathbf{g}}^{(2)}\big)\Big), \qquad (29)$$

where $\tilde{\mathbf{g}}^{(i)} = \mathcal{R}\big(f_\theta(\tilde{\mathbf{A}}^{(i)}, \tilde{\mathbf{X}}^{(i)})\big)$ denotes the representation of augmented graph $\tilde{\mathcal{G}}^{(i)}$, and $\mathcal{R}(\cdot)$ is a readout function to generate the graph-level embedding based on node representations. Methods under Equation (29) may share similar augmentations and backbone contrastive frameworks with the aforementioned node-level approaches. For example, GraphCL(G) [38] adopts SimCLR [24] to form its contrastive pipeline which pulls the graph-level representations of two views closer. Similarly, DACL [89] is also built on SimCLR but it designs a general yet effective augmentation strategy, namely mixup-based data interpolation. AD-GCL [75] proposes an adversarial edge dropping mechanism as augmentations to reduce the amount of redundant information taken by encoders. JOAO [70] proposes the concept of joint augmentation optimization, where a bi-level optimization problem is formulated by jointly optimizing the augmentation selection together with the contrastive objectives. Similar to GCC [15], CSSL [74] is built on MoCo [23] but it contrasts graph-level embeddings. A similar design can

also be found in LCGNN [90]. On the other hand, regarding to the knowledge-distillation, IGSD [73] is leveraging the concept of BYOL [25] and similar to MERIT [68].

6.2.2 Cross-Scale Contrast

Different from contrasting graph instances in an equivalent scale, this branch of methods places the discrimination across various graph topologies (*e.g.*, node versus graph). We further build two sub-classes under this category, namely patch-global and context-global contrast.

6.2.2.1 *Patch-Global Cross-Scale Contrast*: For node-level representation learning, we define this contrast as below:

$$\theta^*, \phi^* =$$
$$\arg\min_{\theta,\phi} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{L}_{con}\Big(p_\phi\big([f_\theta(\mathbf{A}, \mathbf{X})]_{v_i}, \mathcal{R}\big(f_\theta(\mathbf{A}, \mathbf{X})\big)\big)\Big),$$
$$(30)$$

where $\mathcal{R}(\cdot)$ denotes the readout function as we mentioned in previous subsection. Under this category, DGI [13] is the first method that proposes to contrast node-level embeddings with the graph-level representation, which aims to maximize the MI between such two representations from different scales to assist the graph encoder to learn both localized and global semantic information. Based on this idea, GIC [91] first clusters nodes within a graph based on their embeddings, and then pulls nodes closer to their corresponding cluster summaries, which is optimized with a DGI objective simultaneously. Apart from attributed graphs, some works on heterogeneous graphs are based on the similar schema: HDGI [92] can be regarded as a version of DGI on heterogeneous graphs, where the difference is that the final node embeddings of a graph are calculated by aggregating node representations under different meta-paths. Similarly, ConCH [93] shares the same objective with DGI and aggregates meta-path-based node representations to calculate node embeddings of a heterogeneous graph. Differently, DMGI [94] considers a multiplex graph as the combination of several attributed graphs. For each of them, given a selected target node and its associated relation type, the relation-specific node embedding is firstly calculated.

TABLE 3: Main characteristics of contrast-based graph SSL approaches. "NSC", "GSC", "PGCC" and "CGCC" mean node-level same-scale, graph-level same-scale, patch-global cross-scale, and context-global cross-scale contrast, respectively.

Approach	Pretext Task Category	Downstream Task Level	Training Scheme	Data Type of Graph	Graph Augmentation	Objective Function
DeepWalk [30]	NSC	Node	URL	Plain	SS	SkipGram
node2vec [31]	NSC	Node	URL	Plain	SS	SkipGram
GraphSAGE [78]	NSC	Node	URL	Attributed	SS	JSD
SELAR [80]	NSC	Node	JL	Heterogeneous	Meta-path sampling	JSD
LINE [79]	NSC	Node	URL	Plain	SS	JSD
GRACE [33]	NSC	Node	URL	Attributed	NFM+EM	InfoNCE
GROC [18]	NSC	Node	URL	Attributed	NFM+Adversarial EM	InfoNCE
GCA [69]	NSC	Node	URL	Attributed	Adaptive NFM+Adaptive EM	InfoNCE
GraphCL(N) [81]	NSC	Node	URL	Attributed	SS+NFS+EM	InfoNCE
CGPN [82]	NSC	Node	JL	Attributed	None	InfoNCE
GCC [15]	NSC	Node/Graph	PF/URL	Plain	SS	InfoNCE
HeCo [83]	NSC	Node	URL	Heterogeneous	NFM	InfoNCE
BGRL [84]	NSC	Node	URL	Attributed	NFM+EM	BYOL
SelfGNN [85]	NSC	Node	URL	Attributed	GD+Node attributive transformation	BYOL
G-BT [87]	NSC	Node	URL	Attributed	NFM+EM	Barlow Twins
MERIT [68]	NSC	Node	URL	Attributed	SS+GD+NFM+EM	BYOL+InfoNCE
GraphCL(G) [38]	GSC	Graph	PF/URL	Attributed	SS+NFM+EM	InfoNCE
DACL [89]	GSC	Graph	URL	Attributed	Noise Mixing	InfoNCE
AD-GCL [75]	GSC	Graph	PF/URL	Attributed	Adversarail EM	InfoNCE
JOAO [70]	GSC	Graph	PF/URL	Attributed	Automated	InfoNCE
CSSL [74]	GSC	Graph	PF/JL/URL	Attributed	SS+Node insertion/deletion+EM	InfoNCE
LCGNN [90]	GSC	Graph	JL	Attributed	Arbitrary	InfoNCE
IGSD [73]	GSC	Graph	JL/URL	Attributed	GD+EM	BYOL+InfoNCE
DGI [13]	PGCC	Node	URL	Attributed	None	JSD
GIC [91]	PGCC	Node	URL	Attributed	Arbitrary	JSD
HDGI [92]	PGCC	Node	URL	Heterogeneous	None	JSD
ConCH [93]	PGCC	Node	JL	Attributed	None	JSD
DMGI [94]	PGCC	Node	JL/URL	Heterogeneous	None	JSD
EGI [95]	PGCC	Node	PF/JL	Attributed	SS	JSD
STDGI [71]	PGCC	Node	URL	Spatial-temporal	Node feature shuffling	JSD
MVGRL [14]	PGCC	Node/Graph	URL	Attributed	GD+SS	JSD
SUBG-CON [77]	PGCC	Node	URL	Attributed	SS+Node representation shuffling	Triplet
SLiCE [96]	PGCC	Edge	JL	Heterogeneous	None	JSD
InfoGraph [97]	PGCC	Graph	JL/URL	Attributed	None	JSD
Robinson et al. [98]	PGCC	Graph	URL	Attributed	Arbitrary	JSD
BiGI [99]	CGCC	Graph	URL	Heterogeneous	SS	JSD
HTC [100]	CGCC	Graph	JL	Attributed	NFS	JSD
MICRO-Graph [101]	CGCC	Graph	URL	Attributed	SS	InfoNCE
SUGAR [102]	CGCC	Graph	JL	Attributed	SS	JSD

The MI between the graph-level representation and such an node embedding is maximized as in DGI. EGI [95] extracts high-level transferable graph knowledge by enforcing node features to be structure-respecting and then maximizing the MI between the embedding of a node and its surrounding ego-graphs. On spatial-temporal graphs, STDGI [71] maximizes the agreement between the node representations at timestep t with the raw node features at $t + 1$ to guide the graph encoder to capture rich semantic information to predict future node features.

Note that aforementioned methods are not explicitly using any graph augmentations. For patch-global contrastive approaches based on augmentations, we reformulate Equation (30) as follows:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \mathcal{L}_{con}\Big(p_\phi\big(\tilde{\mathbf{h}}_i^{(1)}, \tilde{\mathbf{g}}^{(2)}\big)\Big), \qquad (31)$$

where $\tilde{\mathbf{h}}_i^{(1)} = [f_\theta(\tilde{\mathbf{A}}^{(1)}, \tilde{\mathbf{X}}^{(1)})]_{v_i}$ is the representation of node v_i in augmented view 1, and $\tilde{\mathbf{g}}^{(2)} = \mathcal{R}\big(f_\theta(\tilde{\mathbf{A}}^{(2)}, \tilde{\mathbf{X}}^{(2)})\big)$ denotes the representation of differently augmented view 2. Under the umbrella of this definition, MVGRL [14] first generates two graph views via graph diffusion [76] and subgraph sampling. Then, it enriches the localized and global supervision signals by maximizing the MI between the node embeddings in a view and the graph-level representation of another view. SUBG-CON [77], on the other hand, inherits the objective of MVGRL while it adopts different graph augmentations. Specifically, it first extracts the top-k most informative neighbors of a central node from a large-scale input graph. Then, the encoded node representations are further shuffled to increase the difficulty of pretext task. On heterogeneous graphs, SLiCE [96] pulls nodes closer to their closest contextual graphs, instead of explicitly contrasting nodes with the entire graph. In addition, SLiCE enriches the localized information of node embeddings via a contextual translation mechanism.

For graph-level representation learning based on patch-global contrast, we can formulate it by using Equation (30). InfoGraph [97] shares a similar schema with DGI [13]. It contrasts the graph representation directly with node embeddings to discriminate whether a node belongs to the given graph. To further boost contrastive methods like InfoGraph, Robinson et al. [98] propose a general yet effective hard negative sampling strategy to make the underlying pretext task more challenging to solved.

6.2.2.2 *Context-Global Cross-Scale Contrast*: Another popular design under the category of cross-scale graph contrastive learning is context-global contrast, which is defined below:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathcal{L}_{con}\Big(p_\phi(\tilde{\mathbf{h}}_s, \tilde{\mathbf{g}})\Big), \qquad (32)$$

where $\mathcal{S}$ denotes a set of contextual subgraphs in an augmented input graph $\tilde{\mathcal{G}}$, where augmentations are typically based on graph sampling under this category. In above formula, $\tilde{\mathbf{h}}_s$ is the representation of augmented contextual subgraph s, and $\tilde{\mathbf{g}}$ represents the graph-level representation over all subgraphs in $\mathcal{S}$. Specifically, we let $\tilde{\mathbf{h}}_s = \mathcal{R}\big([f_\theta(\tilde{\mathbf{A}}, \tilde{\mathbf{X}})]_{v_i \in s}\big)$, and $\tilde{\mathbf{g}} = \mathcal{R}\big(f_\theta(\tilde{\mathbf{A}}, \tilde{\mathbf{X}})\big)$. However, for some methods, such as [99] and [100], the graph-level representation is calculated on the original input graph, where $\tilde{\mathbf{g}} = \mathcal{R}\big(f_\theta(\mathbf{A}, \mathbf{X})\big)$. Among them, BiGI [99] is a node-level representation learning approach on bipartite graphs, inheriting the contrasting schema of DGI [13]. Specifically, it first calculates graph-level representation of the input graph by aggregating two types of node embeddings. Then, it samples the original graph, and then calculates the local contextual representation of a target edge between two nodes. The optimization objective of BiGI is to maximize the MI between such a local contextual and global representations, where the trained graph encoder can then be used in various edge-level downstream tasks. Aiming to learn graph-level embedding, HTC [100] maximizes the MI between full-graph representation and the contextual embedding which is the aggregation of sampled subgraphs. Similar to but different from HTC, MICRO-Graph [101] proposes a different yet novel motif learning-based sampling as the implicit augmentation to generate several semantically-informative subgraphs, where the embedding of each subgraph is pulled closer to the representation of entire graph. Considering a scenario that the graph-level representation is based on the augmented input graph, as the default setting shown in Equation (32), SUGAR [102] first samples n subgraphs from the given graph, and then proposes a reinforcement learning-based top-k sampling strategy to select the n' informative subgraphs among the candidate set with size n. Finally, the contrast of SUGAR is established between the subgraph embedding and the representation of sketched graph, i.e., the generated graph by combining these n' subgraphs.

6.3 Mutual Information Estimation

Most of contrast-based methods rely on the MI estimation between two or more instances. Specifically, the representations of a pair of instances sampled from the positive pool are being pulled closer while the counterparts from negative sets are pushed away. Given a pair of instances (x_i, x_j), we let $(\mathbf{h}_i, \mathbf{h}_j)$ to denote their representations. Thus, the MI between (i, j) is given by [103]:

$$\mathcal{MI}(\mathbf{h}_i, \mathbf{h}_j) = KL\big(P(\mathbf{h}_i, \mathbf{h}_j) || P(\mathbf{h}_i)P(\mathbf{h}_j)\big)$$
$$= \mathbb{E}_{P(\mathbf{h}_i, \mathbf{h}_j)}\Big[\log \frac{P(\mathbf{h}_i, \mathbf{h}_j)}{P(\mathbf{h}_i)P(\mathbf{h}_j)}\Big], \qquad (33)$$

where $KL(\cdot)$ denotes the Kullback-Leibler divergence, and the end goal is to train the encoder to be discriminative between a pair of instances from the joint density $P(\mathbf{h}_i, \mathbf{h}_j)$ and negatives from marginal densities $P(\mathbf{h}_i)$ and $P(\mathbf{h}_j)$. In this subsection, we define two common forms of lower

bound and three specific forms of non-bound MI estimators derived from Equation (33).

6.3.1 Jensen-Shannon Estimator

Although Donsker-Varadhan representation provides a tight lower bound of KL divergence [36], Jensen-Shannon divergence (JSD) is more common on graph contrastive learning, which provides a lower bound and more efficient estimation on MI. We define the contrastive loss based on it as follows:

$$\mathcal{L}_{con}\Big(p_\phi(\mathbf{h}_i, \mathbf{h}_j)\Big) = -\mathcal{MI}_{JSD}(\mathbf{h}_i, \mathbf{h}_j)$$
$$= \mathbb{E}_{\mathcal{P} \times \tilde{\mathcal{P}}}\Big[\log\big(1 - p_\phi(\mathbf{h}_i, \mathbf{h}'_j)\big)\Big] - \mathbb{E}_{\mathcal{P}}\Big[\log\big(p_\phi(\mathbf{h}_i, \mathbf{h}_j)\big)\Big].$$
$$(34)$$

In above equation, $\mathbf{h}_i$ and $\mathbf{h}_j$ are sampled from the same distribution $\mathcal{P}$, and $\mathbf{h}'_j$ is sampled from a different distribution $\tilde{\mathcal{P}}$. For the discriminator $p_\phi(\cdot)$, it can be taken from various forms, where a bilinear transformation [104] is typically adopted, i.e., $p_\phi(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{h}_i^T \mathbf{\Phi} \mathbf{h}_j$, such as in [13], [14], [102]. Specifically, by letting $p_\phi(\mathbf{h}_i, \mathbf{h}_j) = \text{sigmoid}\big(p'_\phi(\mathbf{h}_i, \mathbf{h}_j)\big)$, Equation (34) can be presented in another form as in InfoGraph [97].

6.3.2 Noise-Contrastive Estimator

Similar to JSD, noise-contrastive estimator (a.k.a. InfoNCE) provides a lower bound MI estimation that naturally consists of a positive and N negative pairs [36]. An InfoNCE-based contrasitve loss is defined as follows:

$$\mathcal{L}_{con}\Big(p_\phi(\mathbf{h}_i, \mathbf{h}_j)\Big) = -\mathcal{MI}_{NCE}(\mathbf{h}_i, \mathbf{h}_j)$$
$$= -\mathbb{E}_{\mathcal{P} \times \tilde{\mathcal{P}}^N}\Big[\log \frac{e^{p_\phi(\mathbf{h}_i, \mathbf{h}_j)}}{e^{p_\phi(\mathbf{h}_i, \mathbf{h}_j)} + \sum_{n \in N} e^{p_\phi(\mathbf{h}_i, \mathbf{h}'_n)}}\Big], \qquad (35)$$

where the discriminator $p_\phi(\cdot)$ can be the dot product with a temperature parameter τ, i.e., $p(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{h}_i^T \mathbf{h}_j / \tau$, such as in GRACE [33] and GCC [15].

6.3.3 Triplet Loss

Apart from aforementioned two lower bound MI estimators, a triplet margin loss can also be adopted to estimate the MI between data instances. However, minimizing this loss can not guarantee that the MI is being maximized because it cannot represent the lower bound of MI. Formally, Jiao et al. [77] define this loss function as follows:

$$\mathcal{L}_{con}\Big(p(\mathbf{h}_i, \mathbf{h}_j)\Big) = \mathbb{E}_{\mathcal{P} \times \tilde{\mathcal{P}}}\Big[\max\big[p_\phi(\mathbf{h}_i, \mathbf{h}_j) - p_\phi(\mathbf{h}_i, \mathbf{h}'_j) + \epsilon, 0\big]\Big], \qquad (36)$$

where ϵ is a margin value, and the discriminator $p(\mathbf{h}_i, \mathbf{h}_j) = 1/1 + e^{(-\mathbf{h}_i^T \mathbf{h}_j)}$.

6.3.4 BYOL Loss

For the methods inspired by BYOL [25] and not relying on negative samples, such as BGRL [25], their objective functions can also be interpreted as a non-bound MI estimator. Given $\mathbf{h}_i, \mathbf{h}_j \sim \mathcal{P}$, we define this loss as in below:

$$\mathcal{L}_{con}\Big(p(\mathbf{h}_i, \mathbf{h}_j)\Big) = \mathbb{E}_{\mathcal{P} \times \mathcal{P}}\Big[2 - 2 \cdot \frac{[p_\psi(\mathbf{h}_i)]^T \mathbf{h}_j}{\|p_\psi(\mathbf{h}_i)\| \|\mathbf{h}_j\|}\Big], \qquad (37)$$

where p_ψ denotes an online predictor parameterized by ψ in Siamese networks, which prevents the model from collapsing with other mechanisms such as momentum encoders, stop gradient, etc. In particular, the pretext decoder in this case denotes the mean square error between two instances, which has been expanded in the above equation.

6.3.5 Barlow Twins Loss

Similar to BYOL, this objective alleviates the reliance on negative samples but much simpler in implementation, which is motivated by the redundancy-reduction principle. Specifically, given the representations of two views $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ for a batch of data instances sampled from a distribution $\mathcal{P}$, we define this loss function as below [86]:

$$
\mathcal{L}_{con}\big(\mathbf{H}^{(1)}, \mathbf{H}^{(2)}\big) = \mathbb{E}_{\mathcal{B} \sim \mathcal{P}^{|\mathcal{B}|}} \Bigg[\sum_a (1 - \frac{\sum_{i \in \mathcal{B}} \mathbf{H}_{ia}^{(1)} \mathbf{H}_{ia}^{(2)}}{\left\| \mathbf{H}_{ia}^{(1)} \right\| \left\| \mathbf{H}_{ia}^{(2)} \right\|})^2
$$
$$
+ \lambda \sum_a \sum_{b \neq a} \Big(\frac{\sum_{i \in \mathcal{B}} \mathbf{H}_{ia}^{(1)} \mathbf{H}_{ib}^{(2)}}{\left\| \mathbf{H}_{ia}^{(1)} \right\| \left\| \mathbf{H}_{ib}^{(2)} \right\|} \Big)^2 \Bigg],
$$
(38)

where a and b index the dimension of a representation vector, and i indexes the samples within a batch $\mathcal{B}$.

Discussion. In general, same-scale methods usually follow the two-stream contrastive learning frameworks (e.g., SimCLR [24] and BYOL [25]), where InfoNCE and BYOL losses are widely used; In contrast, cross-scale methods often derive from DGI [13], hence they prefer the JSD loss.

7 HYBRID METHODS

Compared to the aforementioned methods that only utilize a single pretext task to train models, hybrid methods adopt multiple pretext tasks to better leverage the advantages of various types of supervision signals. The hybrid methods integrate various pretext tasks together in a *multi-task learning* fashion, where the objective function is the weighted sum of two or more self-supervised objectives. The formulation of hybrid graph SSL methods is:

$$
\theta^*, \phi^* = \arg\min_{\theta, \phi} \sum_{i=1}^{N} \alpha_i \mathcal{L}_{ssl_i} \left(f_\theta, p_{\phi_i}, \mathcal{D}_i \right),
$$
(39)

where N is the number of pretext tasks, α_i, $\mathcal{L}_{ssl_i}$, p_{ϕ_1} and $\mathcal{D}_i$ are the trade-off weight, loss function, pretext decoder and data distribution of the i-th pretext task, respectively.

A common idea of hybrid graph SSL is to combine different generation-based tasks together. GPT-GNN [9] integrates feature and structure generation into a pre-training framework for GNNs. Specifically, for each sampled input graph, it first randomly masks a certain amount of edges and nodes. Then, two generation tasks are used to train the encoder simultaneously: Attribute Generation that rebuilds the masked features with MSE loss, and Edge Generation that predicts the masked edges with a contrastive loss. Graph-Bert [39] combines attributive and structural pretext tasks to pre-train a graph transformer model. Concretely, Node Raw Attribute Reconstruction reconstructs the raw features from the node's embedding, while Graph Structure Recovery aims to recover the graph diffusion value between two nodes with a cosine similarity decoder. PT-DGNN [105]

TABLE 4: Main characteristics of hybrid graph SSL approaches. "M. et al." and "Hetero." are the abbreviations for "Manessi et al." and "Heterogeneous", respectively. The abbreviations for pretext tasks categories please refer to Table 1, 2 and 3.

Approach	Pretext Task Categories	Downstream Task Level	Training Scheme	Data Type of Graph
GPT-GNN [9]	FG/SG	Node/Link	PF	Hetero.
Graph-Bert [39]	FG/SG	Node	PF	Attributed
PT-DGNN [105]	FG/SG	Link	PF	Dynamic
M. et al. [45]	FG/FG/FG	Node	JL	Attributed
GMI [41]	SG/NSC	Node/Link	URL	Attributed
CG³ [106]	SG/NSC	Node	JL	Attributed
MVMI-FT [107]	SG/PGCC	Node	URL	Attributed
GraphLoG [108]	NSC/GSC/CGCC	Graph	PF	Attributed
HDMI [109]	NSC/PGCC	Node	URL	Multiplex
G-Zoom [110]	NSC/NSC/GSC	Node	URL	Attributed
LnL-GNN [111]	NSC/NSC	Node	JL	Attributed
Hu et al. [50]	SG/APC/APC	Node/Link/Graph	PF	Attributed
GROVER [10]	APC/APC	Node/Link/Graph	PF	Attributed
Kou et al. [112]	FG/SG/APC	Node	JL	Attributed

extends the idea of combining attributive and structural generation to pre-train GNNs for dynamic graphs. Besides, Manessi et al. [45] propose to train GNNs with three types of feature generation tasks.

Another idea is to integrate generative and contrastive pretext tasks together. GMI [41] adopts a joint learning objective for graph representation learning. In GMI, the contrastive learning target (i.e., feature MI) is to maximize the agreement between node embeddings and neighbors' features with a JSD estimator, and the generative target (i.e., edge MI) is to minimize the reconstruction error of the adjacency matrix with a BCE loss. CG³ [106] considers contrastive and generative SSL jointly for semi-supervised node classification problem. In CG³, two parallel encoders (GCN and HGCN) are established to provide local and global views for graphs. In contrastive learning, an Info-NCE contrastive loss is used to maximize the MI between the node embeddings from two views. In generative learning, a generative decoder is used to rebuild the topological structure from the concatenation of two views' embeddings. MVMI-FT [107] presents a cross-scale contrastive learning framework that learns node representation from different views, and also uses a graph reconstruction module to learn the cross-view sharing information.

Since different types of contrasts can provide supervision signals from different views, some approaches integrate multiple contrast-based tasks together. GraphLoG [108] consists of three contrastive objectives: the subgraph versus subgraph, graph versus graph, and graph versus contextual. The InfoNCE loss serves as the MI estimator for three types of contrasts. HDMI [109] mixes both same-scale and cross-scale contrastive learning, which dissects a given multiplex network into multiple attributed graphs. For each of them, HDMI proposes three different objectives to maximize the MI between raw node features, node embeddings, and graph-level representations. G-Zoom [110] uses same-scale contrasts in three scales to learns representations, which extracts valuable clues from multiple perspectives. LnL-GNN

[111] leverages a bi-level MI maximization to learn from local and non-local neighborhoods obtained by community detection and feature-based clustering respectively.

Different auxiliary property-based tasks can also be integrated into a hybrid method. Hu et al. [50] present to pre-train GNNs with multiple tasks simultaneously to capture transferable generic graph structures, including Denoising Link Reconstruction, Centrality Score Ranking, and Cluster Preserving. In GROVER [10], the authors pre-train the GNN Transformer model with auxiliary property classification tasks in node level (Contextual Property Prediction) and graph level (Motif Prediction) simultaneously. Kou et al. [112] mix structure generation, feature generation, and auxiliary property classification tasks into a clustering model.

8 EMPIRICAL STUDY

In this section, we summarize essential resources for empirical study of graph SSL. Specifically, we conduct an experimental comparison of the representative methods on two commonly used downstream tasks on graph learning, i.e., node classification and graph classification. We also collect useful resources for empirical research, including benchmark datasets and open-source implementations.

Performance Comparison of Node Classification. We consider two learning settings for node classification, i.e., semi-supervised transductive learning and supervised inductive learning. For transductive learning, we consider three citation network datasets, including Cora, Citeseer and Pubmed [113], for performance evaluation. The standard split of train/valid/test often follows [1], where 20 nodes per class are used for training, 500/1000 nodes are used for validation/testing. For inductive learning, we use PPI dataset [78] to evaluate the performance. Following [78], 20 graphs are employed to train the model, while 2 graphs are used to validate and 2 graphs are used to test. In both setting, the performance is measured by classification accuracy.

We compare the performance of two groups of graph SSL methods. In URL group, the encoder is purely trained by SSL pretext tasks, and the learned representations are directly fed into classification decoders. In PF/JL group, the training labels are accessible for encoders' learning. We consider two conventional (semi-) supervised classification methods (i.e., GCN [1] and GAT [2]) as baselines.

The results of performance comparison are illustrated in Table 5. According to the results, we have the following observations and analysis: (1) Early random walk-based contrastive methods (e.g., DeepWalk and Graph-SAGE) and autoencoder-based generative methods (e.g., GAE and SIG-VAE) perform worse than the majority of graph SSL methods. The possible reason is that they train encoders with simple unsupervised learning targets instead of well-designed self-supervised pretext tasks, hence failing to fully leverage the original data to acquire supervision signals. For example, DeepWalk only maximizes the MI among nodes within a random walk, ignoring the global structural information of graphs. (2) The methods employing advanced contrastive objectives from visual contrastive learning (e.g., BGRL which uses BYOL loss [25] and G-BT which uses Barlow Twins loss [86]) do not show a superior performance like their prototypes performing on visual data. Such an observation indicates that directly

TABLE 5: A summary of experimental results for node classification in four benchmark datasets.

Group	Approach	Category	Cora	Citeseer	Pubmed	PPI
Base-lines	GCN [1]	-	81.5	70.3	79.0	-
	GAT [2]	-	83.0	72.5	79.0	97.3
URL	GAE [32]	SG	80.9	66.7	77.1	-
	SIG-VAE [47]	SG	79.7	70.4	79.3	-
	S²GRL [57]	PAPC	83.7	72.1	82.4	66.0
	DeepWalk [30]	NSC	67.2	43.2	65.3	-
	GraphSAGE [78]	NSC	78.7	69.4	78.1	50.2
	GRACE [33]	NSC	80.0	71.7	79.5	-
	GCA [69]	NSC	81.2	71.8	82.8	-
	GraphCL(N) [81]	NSC	83.6	72.5	79.8	65.9
	BGRL [84]	NSC	80.5	71.0	79.5	-
	G-BT [87]	NSC	81.0	70.8	79.0	-
	MERIT [68]	NSC	83.1	74.0	80.1	-
	DGI [13]	PGCC	82.3	71.8	76.8	63.8
	MVGRL [14]	PGCC	82.9	72.6	79.4	-
	SubG-Con [77]	PGCC	83.5	73.2	81.0	66.9
	GMI [41]	Hybrid	82.7	73.0	80.1	65.0
	MVMI-FT [107]	Hybrid	83.1	72.7	81.0	-
	G-Zoom [110]	Hybrid	84.7	74.2	81.2	-
PF/JL	G. Comp. [17]	FG	81.3	71.7	79.2	
	SuperGAT [49]	SG	84.3	72.6	81.7	74.4
	N. Clu. [17]	CAPC	81.8	71.7	79.2	-
	M3S [40]	CAPC	81.6	71.9	79.3	-
	G. Part. [17]	CAPC	81.8	71.3	80.0	-
	SimP-GCN [58]	APR	82.8	72.6	81.1	-
	Graph-Bert [39]	Hybrid	84.3	71.2	79.3	-
	M. et al. [45]	Hybrid	82.2	71.1	79.3	-
	CG³ [106]	Hybrid	83.4	73.6	80.2	-

borrowing self-supervised objectives from other domains does not always bring enhancement. (3) Some representative contrast-based methods (e.g., MVGRL, MERIT, and SubG-Con) perform better than the generalization-based and auxiliary property-based methods, which reflects the effectiveness of contrastive pretext tasks and the potential room for improvement of other methods. (4) The hybrid methods have competitive performance and some of them even outperform the supervised baselines. The outperformance suggests that integrating multiple pretext tasks can provide supervision signals from diverse perspectives, which brings significant performance gain. For instance, G-Zoom [110] achieves excellent results by combining contrastive pretext tasks in three different levels. (5) The performance of methods in PF/JL groups is generally better than that in URL groups, which demonstrates that the accessibility of label information leads to further improvement for graph SSL.

More resources. For the evaluation and performance comparison of *graph classification*, we please readers refer to Appendix D. We also collect widely applied benchmark datasets and divide them into four groups. The description and statistics of the selected *benchmark datasets* are detailed in Appendix E. Besides, we provide a collection of the *open-source implementations* of the surveyed works in Appendix F, which can facilitate the reproduction, improvement, and baseline experiments in further research.

9 PRACTICAL APPLICATIONS

Graph SSL has also been applied to a wide range of disciplines. We summarize the applications of graph SSL in three research fields. More can be found in Appendix G.

Recommender Systems. Graph-based recommender system has drawn great research attention since it can model items and users with networks and leverage their underlying linkages to produce high-quality recommendations [4]. Recently, researchers introduce graph SSL in recommender systems to deal with several issues, including the cold-start problem, pre-training for recommendation model, se-

lection bias, etc. For instance, Hao et al. [114] present a reconstruction-based pretext task to pre-train GNNs on the cold-start users and items. S^2-MHCN [115] and DHCN [116] employ contrastive tasks for hypergraph representation learning for social- and session- based recommendation, respectively. Liu et al. [117] overcome the message dropout problem and reduce the selection bias in GNN-based recommender system by introducing a graph contrastive learning module with a debiased loss. PMGT [118] utilizes two generation-based tasks to capture multimodal side information for recommendation.

Anomaly Detection. Graph anomaly detection is often performed under an unsupervised scenario due to the lack of annotated anomalies, which is naturally consistent with the setting of SSL [119]. Hence, various works apply SSL to graph anomaly detection problem. To be concrete, DOMINANT [120], SpecAE [121] and AEGIS [122] employ hybrid SSL frameworks that combine structure and feature generation to capture the patterns of anomalies. CoLA [119] and ANEMONE [123] utilize contrastive learning to detect anomalies on graphs. SL-GAD [124] applies hybrid graph SSL to anomaly detection. HCM [125] introduces an auxiliary property classification task that predicts the hop-count of each node pair for graph anomaly detection.

Chemistry. In the domain of chemistry, researchers usually model molecules or compounds as graphs where atoms and chemical bonds are denoted as nodes and edges respectively. Note that GROVER [10] and Hu et al. [16] also focus on graph SSL for molecule data, which have been reviewed before. Additionally, MolCLR [126] and CKGNN [127] learn molecular representations with graph-level contrast-based pretext tasks. Besides, GraphMS [128] and MIRACLE [129] employ contrastive learning to solve the drug–target and drug-drug interaction prediction problems.

10 FUTURE DIRECTIONS

In this section, we analyze existing challenges in graph SSL and pinpoint a few future research directions aiming to address the shortcomings.

Theoretical Foundation. Despite its great success in various tasks and datasets, graph SSL still lacks a theoretical foundation to prove its usefulness. Most existing methods are mainly designed with intuition and evaluated by empirical experiments. Although MI estimation theory [65] supports some of the works on contrastive learning, the choice of the MI estimator still relies on empirical studies [14]. Setting up a solid theoretical foundation for graph SSL is urgently needed. It is desirable to bridge the gap between empirical SSL and fundamental graph theories, including the graph signal processing and spectral graph theory.

Interpretability and Robustness. Graph SSL applications may be risk-sensitive and privacy-related (*e.g.*, fraud detection), an explainable and robust SSL framework is of great significance to adapt to such learning scenarios. However, most existing graph SSL methods only aim to reach a higher performance on downstream tasks with black-box models, ignoring the explainability of learned representations and predicted results. Moreover, except for a few pioneering works [17], [18] that consider the robustness problem, most graph SSL methods assume input data is perfect, despite the fact that real-world data is often noisy and GNNs

are vulnerable to adversarial attacks [18]. It would be an interesting and practical direction to explore explainable and robust graph SSL methods in future.

Pretext Tasks for Complex Types of Graphs. Most current works concentrate on SSL for attributed graphs, and only a few focus on complex graph types, *e.g.*, heterogeneous or spatial-temporal graphs. For complex graphs, the main challenge is how to design pretext tasks to capture unique data characteristics of these complex graphs. Some existing methods use MI maximization [13] for complex graph learning, which is limited in its ability to leverage rich information from data, *e.g.*, the temporal dynamics in spatial-temporal/dynamic graphs. A future opportunity is to produce various SSL tasks for complex graph data, where specific data characteristics are the main focus. Furthermore, extending SSL to more ubiquitous graph types (*e.g.*, hypergraphs) would be a feasible direction for further exploration.

Augmentation for Graph Contrastive Learning. In contrastive learning for CV [24], a large amount of augmentation strategies (including rotation, color distort, crop, etc.) provide diverse views of image data, maintaining the representation invariance during contrastive learning. However, due to the nature of graph-structured data (*e.g.*, complex and non-Euclidean structure), data augmentation schemes on graphs are not well explored and thus compromise the effectiveness of graph augmentation-based approaches as discussed in Section 6.1. Most of the existing graph augmentations consider uniformly masking/shuffling node features, modifying edges, or other alternative ways like subgraph sampling and graph diffusion [76], which provides limited diversity and uncertain invariance when generating multiple graph views. To bridge the gaps, adaptively performing graph augmentations [69], automatically selecting augmentations [70] or jointly considering stronger augmented samples [67] by mining the rich underlying structural and attributive information would be interesting directions for further investigation.

Learning with Multiple Pretext Tasks. Most existing graph SSL approaches learn representations by solving one pretext task, while only a few hybrid methods explore the combination of multiple pretext tasks. As shown in previous NLP pre-training models [28] and the reviewed hybrid methods, the integration of diverse pretext tasks can provide different supervision signals from various perspectives, which facilitates graph SSL methods to produce more informative representations. Therefore, more advanced hybrid approaches that consider a diverse and adaptive combination of multiple pretext tasks deserve further studies.

Broader Scope of Applications. Graphs are ubiquitous data structures in numerous domains; nevertheless, acquiring manual labels is often costly in most application fields. In that case, graph SSL has a promising prospect on a wide range of applications, especially those that highly depend on domain knowledge to annotate data. However, most current practical applications merely concentrate on a few areas (*e.g.*, recommender systems, anomaly detection, and chemistry), indicating that graph SSL holds untapped potential for most application fields. It is promising to extend graph SSL to more expansive fields of applications, for instance, financial networks, cybersecurity, community detection [130], and federated learning.

11 CONCLUSION

This paper conducts a comprehensive overview of self-supervised learning on graphs. We present a unified framework and further provide a systematic taxonomy that groups graph SSL into four categories: generation-based, auxiliary property-based, contrast-based, and hybrid approaches. For each category, we provide mathematical summary, up-to-date review, and comparison between methods. More importantly, we collect abundant resources including datasets, evaluation methods, performance comparison and open-source codes for graph SSL approaches. A wide range of practical applications of graph SSL are also introduced in our paper. Finally, we suggest open challenges and promising research directions of graph SSL in the future.

APPENDIX A
TIMELINE OF GRAPH SELF-SUPERVISED LEARNING

Fig. 9 provides a timeline of some of the most representative graph SSL methods since 2014. A group of early works is the random walk-based node embedding methods including DeepWalk [30], LINE [79], and node2vec [31]. Another line of early works is the graph autoencoder-based generation methods, i.e. GAE/VGAE [32], MGAE [44], ARGA [48], and SIG-VAE [47]. In 2019, a representation contrast-based work, DGI [13], was proposed, bringing the flourishing development of graph contrastive learning. In 2020, more types graph SSL methods were introduced, containing the first auxiliary property-based method (M3S [40]) and hybrid method (GMI [41]). In 2021, more advanced techniques are integrated with graph SSL, such as adaptive augmentation (GCA [69]), automatic machine learning (JOAO [70]), and adversarial augmentation (AD-GCL [75]).

APPENDIX B
NOTATIONS AND DEFINITIONS

B.1 Notations

Throughout this paper, we use bold uppercase letters (e.g. $\mathbf{X}$), bold lowercase letters (e.g. $\mathbf{x}$), and calligraphic fonts (e.g. $\mathcal{V}$) to denote matrices, vectors and sets, respectively. Unless specified otherwise, the notations used in this paper are summarized in Table 6.

B.2 Dynamic and Heterogeneous Graphs

Definition 3 (Dynamic Graph). A dynamic graph indicates a graph where nodes and edges change with respect to time, i.e., $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$, which denotes a dynamic graph at time $t \in \mathbb{R}^+$ that consists of a node set $\mathcal{V}^{(t)}$ and an edge set $\mathcal{E}^{(t)}$ indexed by time. In practice, a dynamic graph is also known as temporal dependency interaction graph, where an interaction at time t is represented as $e_{i,j,t} \in \mathcal{E}^{(t)}$, where node $v_i \in \mathcal{V}^{(t)}$ and $v_j \in \mathcal{V}^{(t)}$. For an attributed dynamic graph with node and edge features, we can also let $\mathcal{G}^{(t)} = (\mathbf{A}^{(t)}, \mathbf{X}_{\mathbf{node}}^{(t)}, \mathbf{X}_{\mathbf{edge}}^{(t)})$, where the adjacency matrix $\mathbf{A}^{(t)} \in \mathbb{R}^{n \times n}$, node feature matrix $\mathbf{X}_{\mathbf{node}}^{(t)} \in \mathbb{R}^{n \times d_{node}}$ and edge feature matrix $\mathbf{X}_{\mathbf{edge}}^{(t)} \in \mathbb{R}^{n \times d_{node}}$ are evolving with respect to time.

TABLE 6: Commonly used notations with explanations.

Notation	Explanation
$\lvert \cdot \rvert$	The length of a set.
$\circ$	Hadamard product.
$\mathcal{G}$	A graph.
$\mathcal{V}$	The set of nodes in a graph.
$\mathcal{E}$	The set of edges in a graph.
v_i	A node in the node set $\mathcal{V}$.
$e_{i,j}$	An edge in the edge set $\mathcal{E}$.
n	The number of nodes in a graph.
m	The number of edges in a graph.
$\mathbf{A} \in \mathbb{R}^{n \times n}$	The adjacency matrix of a graph.
d, d_{node}	The dimension of node features.
d_{edge}	The dimension of edge features.
$\mathbf{X}, \mathbf{X}_{\mathbf{node}} \in \mathbb{R}^{n \times d}$	The node feature matrix of a graph.
$\mathbf{X}_{\mathbf{edge}} \in \mathbb{R}^{m \times d_{edge}}$	The edge feature matrix of a graph.
$\mathbf{x_i} = [\mathbf{X}]_{v_i}$	The node feature of v_i.
d_r	The dimension of node representation.
$\mathbf{H} \in \mathbb{R}^{n \times d_r}$	The node representation matrix of a graph.
$\mathbf{h_i} = [\mathbf{H}]_{v_i}$	The node representation vector of v_i.
$\mathbf{h}_{\mathcal{G}}$	The graph representation vector of a graph.
$\mathcal{Y}$	Manual label set of downstream task.
$\mathcal{C}$	Pseudo label set of pretext task.
y	Manual label of downstream task.
c	Pseudo label of pretext task.
$\tilde{\mathcal{G}}$	Augmented/Perturbed graph.
$f_\theta(\cdot)$	Encoder parameterized by θ.
$p_\phi(\cdot)$	Pretext decoder parameterized by ϕ.
$q_\psi(\cdot)$	Downstream decoder parameterized by ψ.
$\mathcal{L}(\cdot)$	Loss function.
$\mathcal{R}(\cdot)$	Readout function.
$\Omega(\cdot)$	Auxiliary property mapping function.
$\mathcal{MI}(\cdot, \cdot)$	Mutual information function.

Special type: *Spatial-temporal graph* can be regarded as an special type of attributed dynamic graph with fixed graph adjacency and dynamic features at different time steps. Concretely, at each time step t, the dynamic feature matrix is denoted as $\mathbf{X}^{(t)} \in \mathbb{R}^{n \times d}$.

Definition 4 (Heterogeneous Graph). A heterogeneous graph denotes a graph consisting of different types of nodes and/ or edges. For a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each node $v_i \in \mathcal{V}$ and each edge $e_{i,j} \in \mathcal{E}$ are associated with a corresponding mapping function, i.e., $\phi_v(v_i) : \mathcal{V} \to \mathcal{S}_v$ and $\phi_e(e_{i,j}) : \mathcal{E} \to \mathcal{S}_e$. $\mathcal{S}_v$ and $\mathcal{S}_e$ are the node types and link types, respectively, and they satisfy $\lvert \mathcal{S}_v \rvert + \lvert \mathcal{S}_e \rvert > 2$.

Special types: *Bipartite graph* and *multiplex graph* can be viewed as two special types of heterogeneous graphs. Specifically, a bipartite graph is a heterogeneous graph with two types of nodes ($\lvert \mathcal{S}_v \rvert = 2$) and a single type of edge ($\lvert \mathcal{S}_e \rvert = 1$); a multiplex graph has one type of node ($\lvert \mathcal{S}_v \rvert = 1$) and multiple types of edges ($\lvert \mathcal{S}_e \rvert > 1$).

B.3 Graph Neural Networks and Readout Layers

A general definition of Graph Neural Networks (GNNs) is represented as:

Definition 5 (Graph Neural Networks). Given an attributed graph $\mathcal{G}$ with its feature matrix $\mathbf{X}$ where $\mathbf{x_i} = \mathbf{X}[i, :]^T$ is a d-dimensional feature vector of the node v_i, a GNN learns a node representation $\mathbf{h_i}$ for each node $v_i \in \mathcal{V}$ with two core functions: the aggregation function and
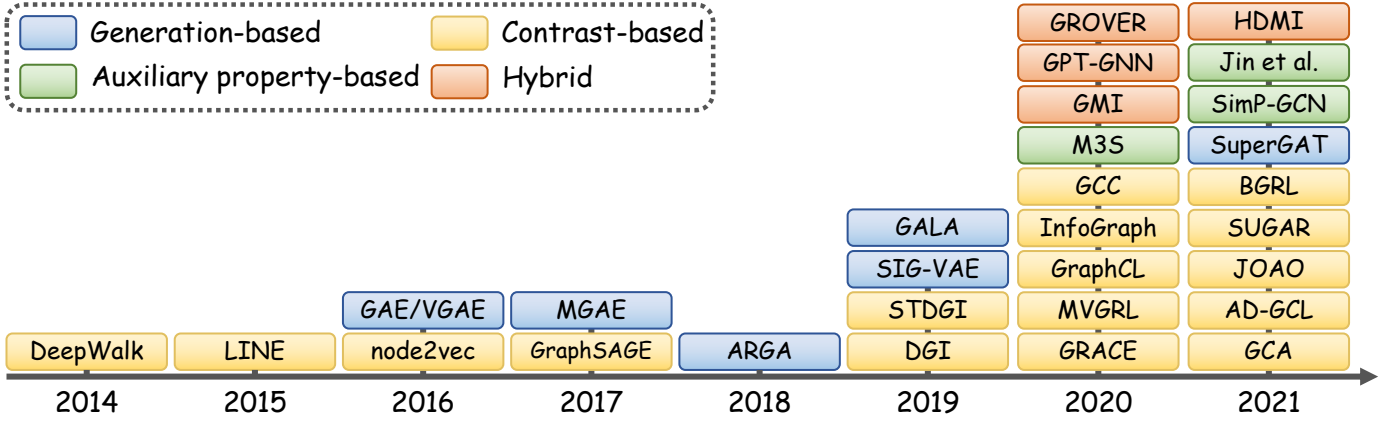
Fig. 9: A timeline of graph self-supervised learning (SSL) development.

combination function. Considering a K-layer GNN, the k-th layer performs:

$$
\begin{aligned}
\mathbf{a_i^{(k)}} &= \text{aggregate}^{(k)} \left(\left\{ \mathbf{h_j^{(k-1)}} : v_j \in \mathcal{N}(v_i) \right\} \right), \\
\mathbf{h_i^{(k)}} &= \text{combine}^{(k)} \left(\mathbf{h_i^{(k-1)}}, \mathbf{a_i^{(k)}} \right),
\end{aligned}
\tag{40}
$$

where $\mathbf{h_i^{(k)}}$ is the latent vector of node v_i at the k-th iteration/layer with $\mathbf{h_i^{(0)}} = \mathbf{x_i}$ and $\mathbf{h_i^{(K)}} = \mathbf{h_i}$, $\text{aggregate}^{(k)}(\cdot)$ and $\text{combine}^{(k)}(\cdot)$ are the aggregation and combination function of the $k - th$ layer, respectively. The design of the core functions is crucial to different GNNs, and we refer the reader to the recent survey [4] for a thorough review.

To learn the property of each node, the node representation $\mathbf{h_i}$ is used for downstream tasks directly. Nevertheless, when learning the representation of the entire graph, an extra readout layer is needed, which is defined as:

Definition 6 (Readout layer). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with its node representation $\mathbf{h_0^{(K)}}, \cdots, \mathbf{h_n^{(K)}}$, a readout layer generates the graph representation $\mathbf{h}_\mathcal{G}$ by:

$$
\mathbf{h}_\mathcal{G} = \mathcal{R} \left(\left\{ \mathbf{h_i}^{(K)} \mid v_i \in \mathcal{V} \right\} \right)
\tag{41}
$$

where $\mathcal{R}(\cdot)$ is the readout function that fuses the node-level representations into a graph-level representation. It can be a simple permutation invariant function like summation or complex pooling algorithms like DiffPool [131] and HaarPool [132].

B.4 Commonly Used Loss Functions

We formulate two commonly used loss functions, i.e., mean squared error (MSE) loss and cross-entropy (CE) loss.

Definition 7 (Mean squared error (MSE) loss).
Given a predicted vector $\hat{\mathbf{y}} \in \mathbb{R}^{d_y}$ and a target vector $\mathbf{y} \in \mathbb{R}^{d_y}$, the formulation of MSE loss is defined as follows:

$$
\mathcal{L}_{mse}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{d_y} \sum_{i=1}^{d_y} (\hat{y}_i - y_i)^2,
\tag{42}
$$

where $\hat{y}_i$ and y_i are the i-th element of $\hat{\mathbf{y}}$ and $\mathbf{y}$, respectively. Note that $\mathcal{L}_{mse}$ can also be applied to matrices and scalars.

Definition 8 (Cross-entropy (CE) loss).
Given a predicted vector $\hat{\mathbf{y}} \in \mathbb{R}^{d_y}$ after Softmax activation and a one-hot target vector $\mathbf{y} \in \mathbb{R}^{d_y}$, the formulation of CE loss is defined as follows:

$$
\mathcal{L}_{ce}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{d_y} y_i \log \hat{y}_i.
\tag{43}
$$

Specially, the binary version of CE loss is denoted as binary cross-entropy (BCE) loss, which is expressed by:

$$
\mathcal{L}_{bce}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log (1 - \hat{y}),
\tag{44}
$$

where $\hat{y} \in [0, 1]$ and $y \in \{0, 1\}$ are the predicted and target scalars, respectively.

APPENDIX C
DOWNSTREAM TASKS

C.1 Node-level tasks

Node-level tasks includes node regression and node classification. As an representative task, node classification aims to predict the label $y_i \in \mathcal{Y}$ for each node $v_i \in \mathcal{V}$, where $\mathcal{Y}$ is a finite and discrete label set. A typical downstream decoder q_ψ for node classification is an MLP layer with Softmax activation that takes node representation as its input. The cross-entropy (CE) loss $\mathcal{L}_{ce}$ is typically adopted for model training. The formulation of node classification is defined as follows based on Equation (2):

$$
f_{\theta^*}, q_{\psi^*} = \arg\min_{f_{\theta^*}, q_\psi} \frac{1}{|\mathcal{V}_L|} \sum_{v_i \in \mathcal{V}_L} \mathcal{L}_{ce} \Big(q_\psi \big([f_\theta(\mathcal{G})]_{v_i} \big), y_i \Big),
\tag{45}
$$

where $\mathcal{V}_L$ is the training node set (where each node $v_i \in \mathcal{V}_L$ has a known label y_i for training), and $[\cdot]_{v_i}$ is a picking function that index the embedding of v_i from the whole embedding matrix.

C.2 Link-level tasks

Link-level tasks includes edge classification and link prediction. Taking edge classification as an example, given an edge (v_i, v_j), the goal is to predict its label $y_{i,j} \in \mathcal{Y}$ from a given label set. A downstream decoder q_ψ could be a classifier with the embeddings of two nodes as input, while CE

$\mathcal{L}_{ce}$ serves as the loss function. We formalize the objective function as follows:

$$f_{\theta^*}, q_{\psi^*} = \operatorname*{arg\,min}_{f_{\theta^*}, q_\psi} \frac{1}{|\mathcal{E}_L|} \sum_{v_i, v_j \in \mathcal{E}_L} \mathcal{L}_{ce}\Big(q_\psi\big([f_\theta(\mathcal{G})]_{v_i, v_j}\big), y_{i,j}\Big),$$
(46)

where $\mathcal{E}_L$ is the training edge set, and $[\cdot]_{v_i, v_j}$ is a picking function that index the embedding of v_i and v_j from the whole embedding matrix.

C.3 Graph-level Tasks

Graph-level tasks, including graph classification and graph regression, often rely on graph-level representations. For instance, in graph classification task, each graph $\mathcal{G}_i$ has its target value $y_i \in \mathcal{Y}$, and the objective is to train a model to predict the labels of input graphs. In general, an encoder f_θ first learns node embeddings and then aggregates them into a graph embedding via a readout function. After that, the graph embedding is fed into the decoder q_ψ composed by a classifier with Softmax function. The model is trained by CE loss $\mathcal{L}_{ce}$, which can be formalized as:

$$f_{\theta^*} = \operatorname*{arg\,min}_{f_\theta, p_\phi} \frac{1}{|\mathcal{U}_L|} \sum_{\mathcal{G}_i \in \mathcal{U}_L} \mathcal{L}_{ce}\Big(p_\phi\big(f_\theta(\mathcal{G}_i)\big), y_i\Big),$$
(47)

where $\mathcal{U}_L$ is the training graph set.

APPENDIX D
PERFORMANCE COMPARISON OF GRAPH CLASSIFICATION

The task of graph classification often adapts an inductive supervised learning setting. We collect the experimental results on nine benchmark datasets, including IMDB-B, IMDB-M, RDT-B, RDT-M, COLLAB, MUTAG, PROTEINS, PTC, and NCI-1 [133]. We consider the dataset split which is utilized in [3]. Classification accuracy is leveraged to evaluate the performance. Similar to node classification, we divide the approaches into two groups: URL and PF/JL. A powerful GNN for graph classification, GIN [3], is employed as our baseline.

The evaluation results are demonstrated in Table 7. Based on the results, we have the following observations: (1) The overall performance of the methods in URL group is lower than that of the supervised baseline (i.e., GIN). The reason is that current SSL methods cannot learn optimal representations as supervised methods do. Considering the URL methods for node classification usually achieve higher results than baselines, future SSL methods for graph-level representation learning are expected to reach higher performance. (2) The cross-scale contrast-based methods, in general, have better performance than the same-scale contrast-based methods. A possible reason is that cross-scale contrastive learning can improve the quality of global-wise (i.e., graph-level) representations, which benefits graph-level learning tasks. (3) Taking GCC as an example, the involvement of graph labels (i.e., PF/JL schemes) does not necessarily lead to better encoders on all datasets compared to pure unsupervised learning (URL) scheme. We

conclude that pre-trained encoders with SSL sometimes have a negative effect on downstream tasks. In this case, how to circumvent such a negative transfer phenomenon in graph-level SSL deserves further investigation. (4) Most of these methods are measured on different sets of graph-level datasets due to the lack of general benchmarks for evaluation. Therefore, unified benchmarks for graph-level SSL evaluation are significantly needed in future works.

APPENDIX E
DATASETS

In this section, we conduct an introduction and a summary of commonly used datasets in four categories, including citation networks, co-purchase networks, social networks, and bio-chemical graphs. A statistic of these datasets is given in Table 8.

E.1 Citation Networks

In citation networks, nodes often represent the published papers and/or authors, while edges denote the relationship between nodes, such as citation, authorship, and co-authorship. The features of each node usually contain the context of the papers or authors, and the labels denote the fields of study for each paper or author. Specifically, in Cora, Citeseer and Pubmed [113], the nodes are papers, the edges are citation relationships, and the features are the bag-of-word representation for papers. In Coauthor CS and Coauthor Physics [134], the nodes are authors, the edges are co-authorship between authors, and the features are the keywords for each author's papers. In Wiki-CS [135], nodes represent papers about computer science, edges represent the hyperlinks between papers, and node features are the mean vectors of GloVe word embeddings of articles. The definition in ogbn-arxiv [136] is similar to Wiki-CS.

E.2 Co-purchase Networks

In the two co-purchase graphs from Amazon (Amazon Computers and Amazon Photo [134]), the nodes indicate goods, and the edges indicate that two goods are frequently bought together. The features of each node are the bag-of-words encoded product reviews, while the class labels are obtained by the category of goods.

E.3 Social Networks

The social network datasets are often formed by users and their interactions on online services or communities. For instance, in Reddit [78], the data is collected from a large online discussion forum named Reddit. The nodes are the users post of the forum, the edges are the comments between users, and the class labels is the communities. The features are composed by: (1) the average embedding of the post title, (2) the average embedding of all the post's comments (3) the post's score, and (4) the number of comments made on the post. IMDB-B and IMDB-M [133] are two movie-collaboration datasets. Each graph includes the actors/actresses and genre information of a movie. In a graph, nodes indicate actors/actresses, while edges indicate if them if they play the same movie. In REDDIT-BINARY (RDT-B)

TABLE 7: A summary of experimental results for graph classification in nine benchmark datasets.

Group	Approach	Category	IMDB-B	IMDB-M	RDT-B	RDT-M	COLLAB	MUTAG	PROTEINS	PTC	NCI-1
Baselines	GIN [3]	-	75.1	52.3	92.4	57.5	80.2	89.4	76.2	64.6	82.7
URL	GCC [15]	NSC	72.0	49.4	89.8	53.7	78.9	-	-	-	-
	GraphCL(G) [38]	GSC	71.1	-	89.5	-	71.4	86.8	74.4	-	77.9
	AD-GCL [75]	GSC	72.3	49.9	85.5	54.9	73.3	89.7	73.8	-	-
	JOAO [70]	GSC	70.8	-	86.4	56.0	69.5	87.7	74.6	-	78.3
	IGSD [73]	GSC	74.7	51.5	-	-	70.4	90.2	-	61.4	75.4
	MVGRL [14]	PGCC	74.2	51.2	84.5	-	-	89.7	-	-	-
	InfoGraph [97]	PGCC	73.0	49.7	82.5	53.5	-	89.0	-	-	
	HTC [100]	CGCC	73.3	50.6	91.3	55.2	-	91.8	-	-	-
PF/JL	GCC [15]	NSC	73.8	50.3	87.6	53.0	81.1	-	-	-	-
	CSSL [74]	GSC	-	-	-	-	-	-	85.8	-	80.1
	LCGNN [90]	GSC	76.1	52.4	-	-	77.5	90.5	85.2	65.9	82.9

TABLE 8: A summary of selected benchmark datasets. The markers "*" in the "# Classes" column indicate that it is a multi-label classification dataset.

Category	Dataset	# Graphs	# Nodes(Avg.)	# Edges (Avg.)	#Features	# Classes	Citation
Citation Networks	Cora [113]	1	2,708	5,429	1,433	7	[17], [32], [40], [43], [45], [50], [56] [13], [14], [33], [49], [57], [77], [91] [39], [41], [44], [46], [47], [48], [106] [18], [51], [58], [68], [81], [85], [112] [82], [107], [110], [111]
	Citeseer [113]	1	3,327	4,732	3,703	6	[17], [32], [40], [43], [45], [56], [57] [13], [14], [33], [41], [49], [77], [91] [39], [44], [46], [47], [48], [51], [106] [18], [58], [68], [81], [85], [107], [112] [82], [110], [111]
	Pubmed [113]	1	19,717	44,338	500	3	[17], [32], [40], [43], [45], [50], [56] [13], [33], [41], [49], [57], [77], [91] [18], [39], [46], [47], [48], [51], [106] [58], [68], [81], [82], [85], [107], [112] [111]
	CoAuthor CS [134]	1	18,333	81,894	6,805	15	[49], [68], [69], [84], [85], [91], [106] [87]
	CoAuthor Physics [134]	1	34,493	247,962	8,415	5	[49], [69], [84], [85], [87], [91]
	Wiki-CS [135]	1	11,701	216,123	300	10	[18], [49], [69], [84], [87]
	ogbn-arxiv [136]	1	169,343	1,166,243	128	40	[49], [84], [87]
Co-purchase Networks	Amazon Photo [134]	1	7,650	119,081	745	8	[18], [49], [69], [82], [84], [91], [106] [68], [85], [87], [107]
	Amazon Computers [134]	1	13,752	245,861	767	10	[49], [69], [84], [85], [91], [106], [107] [87]
Social Networks	Reddit [78]	1	232,965	11,606,919	602	41	[9], [13], [33], [43], [57], [77], [78] [41], [81]
	IMDB-B [133]	1,000	19.77	193.06	-	2	[14], [15], [50], [73], [89], [90], [97] [75], [98], [100]
	IMDB-M [133]	1,500	13.00	65.93	-	3	[14], [15], [50], [73], [89], [90], [97] [75], [98], [100]
	RDT-B [133]	2,000	429.63	497.75	-	2	[14], [15], [38], [89], [97], [98], [100] [75]
	RDT-M [133]	4,999	594.87	508.52	-	5	[15], [75], [89], [97], [100]
	COLLAB [133]	5,000	74.49	2,457.78	-	3	[15], [38], [73], [75], [90]
Bio-chemical Graphs	PPI [137]	24	56,944	818,716	50	121*	[13], [33], [41], [49], [57], [77], [78] [81], [84], [87]
	MUTAG [138]	188	17.93	19.79	7	2	[14], [73], [89], [90], [97], [100], [102] [70], [75], [98]
	PROTEINS [139]	1,113	39.06	72.81	4	2	[38], [70], [74], [75], [90], [98], [102]
	D&D [140]	1,178	284.31	715.65	82	2	[70], [74], [75], [90], [98], [102]
	PTC [141]	349	14.10	14.50	19	2	[73], [90], [98], [102]
	PTC-MR [142]	344	14.29	14.69	18	2	[14], [89], [97], [100]
	NCI-1 [143]	4,110	29.87	32.30	37	2	[38], [70], [73], [74], [75], [90], [102]
	NCI-109 [143]	4,127	29.68	32.13	38	2	[74], [102]
	BBBP [144]	2,039	24.06	25.95	-	2	[10], [16], [70], [75], [101], [108]
	Tox21 [145]	7,831	18.51	25.94	-	12*	[10], [16], [70], [75], [101], [108]
	ToxCast [146]	8,575	18.78	19.26	-	167*	[10], [16], [70], [75], [101], [108]
	SIDER [147]	1,427	33.64	35.36	-	27*	[10], [16], [70], [75], [101], [108]
	ClinTox [148]	1,478	26.13	27.86	-	2	[10], [16], [70], [75], [101], [108]
	MUV [149]	93,087	24.23	26.28	-	17*	[16], [70], [75], [108]
	HIV [150]	41,127	25.53	27.48	-	2	[16], [70], [75], [101], [108]
	BACE [151]	1,513	34.12	36.89	-	2	[10], [16], [70], [75], [101], [108]

TABLE 9: A summary of open-source implementations.

Model	Framework	Github Link
You et al. (2020) [17]	torch	https://github.com/Shen-Lab/SS-GCNs
Jin et al. (2021) [43]	torch	https://github.com/ChandlerBang/SelfTask-GNN
Hu et al. (2020) [16]	torch	https://github.com/snap-stanford/pretrain-gnns
MGAE (2017) [44]	matlab	https://github.com/GRAND-Lab/MGAE
GAE/VGAE (2016) [32]	tensorflow	https://github.com/tkipf/gae
SIG-VAE (2019) [47]	tensorflow	https://github.com/sigvae/SIGraphVAE
ARGA/ARVGA (2018) [48]	tensorflow	https://github.com/GRAND-Lab/ARGA
SuperGAT (2021) [49]	torch	https://github.com/dongkwan-kim/SuperGAT
M3S (2020) [40]	tensorflow	https://github.com/datake/M3S
SimP-GCN (2021) [58]	torch	https://github.com/ChandlerBang/SimP-GCN
DeepWalk (2014) [30]	gensim	https://github.com/phanein/deepwalk
node2vec (2016) [31]	gensim	https://github.com/aditya-grover/node2vec
GraphSAGE (2017) [78]	tensorflow	https://github.com/williamleif/GraphSAGE
SELAR (2020) [80]	torch	https://github.com/mlvlab/SELAR
LINE (2015) [79]	c++	https://github.com/tangjianpku/LINE
GRACE (2020) [33]	torch	https://github.com/CRIPAC-DIG/GRACE
GCA (2021) [69]	torch	https://github.com/CRIPAC-DIG/GCA
GCC (2020) [15]	torch	https://github.com/THUDM/GCC
HeCo (2021) [83]	torch	https://github.com/liun-online/HeCo
BGRL (2021) [84]	torch	https://github.com/Namkyeong/BGRL_Pytorch
SelfGNN (2021) [85]	torch	https://github.com/zekarias-tilahun/SelfGNN
G-BT (2021) [87]	torch	https://github.com/pbielak/graph-barlow-twins
MERIT (2021) [68]	torch	https://github.com/GRAND-Lab/MERIT
GraphCL(G) (2020) [38]	torch	https://github.com/Shen-Lab/GraphCL
AD-GCL (2021) [75]	torch	https://github.com/susheels/adgcl
JOAO (2021) [70]	torch	https://github.com/Shen-Lab/GraphCL_Automated
LCGNN (2021) [90]	torch	https://github.com/YuxiangRen/Label-Contrastive-Coding-based-Graph-Neural-Network-for-Graph-Classification-
DGI (2019) [13]	torch	https://github.com/PetarV-/DGI
GIC (2021) [91]	torch	https://github.com/cmavro/Graph-InfoClust-GIC
HDGI (2020) [92]	torch	https://github.com/YuxiangRen/Heterogeneous-Deep-Graph-Infomax
ConCH (2020) [93]	torch	https://github.com/dingdanhao110/Conch
DMGI (2020) [94]	torch	https://github.com/pcy1302/DMGI
MVGRL (2020) [14]	torch	https://github.com/kavehhassani/mvgrl
SUBG-CON (2020) [77]	torch	https://github.com/yzjiao/Subg-Con
InfoGraph (2020) [97]	torch	https://github.com/fanyun-sun/InfoGraph
SLiCE (2021) [96]	torch	https://github.com/pnnl/SLiCE
Robinson et al. (2020) [98]	torch	https://github.com/joshr17/HCL
EGI (2020) [95]	torch	https://openreview.net/attachment?id=J_pvI6ap5Mn&name=supplementary_material
BiGI (2021) [99]	torch	https://github.com/caojiangxia/BiGI
HTC (2021) [100]	torch	https://github.com/Wastedzz/LGR
MICRO-Graph (2020) [101]	torch	https://openreview.net/attachment?id=qcKh_Msv1GP&name=supplementary_material
SUGAR (2021) [102]	tensorflow	https://github.com/RingBDStack/SUGAR
GPT-GNN (2020) [9]	torch	https://github.com/acbull/GPT-GNN
Graph-Bert (2020) [39]	torch	https://github.com/jwzhanggy/Graph-Bert
PT-DGNN (2021) [105]	torch	https://github.com/Mobzhang/PT-DGNN
GMI (2020) [41]	torch	https://github.com/zpeng27/GMI
MVMI-FT (2021) [107]	torch	https://github.com/xiaolongo/MaxMIAcrossFT
GraphLoG (2021) [108]	torch	https://openreview.net/attachment?id=DAaaaqPv9-q&name=supplementary_material
HDMI (2021) [109]	torch	https://github.com/baoyujing/HDMI
LnL-GNN (2021) [111]	torch	https://github.com/forkkr/LnL-GNN
GROVER (2020) [10]	torch	https://github.com/tencent-ailab/grover

and REDDIT-MULTI-5K (RDT-M) [133], each graph denotes an online discussion thread. In each graph, the nodes are the users, and the edges are the comments between users. COLLAB [133] is a scientific-collaboration dataset, where each graph is a ego-network of an researcher, and the label is the field of this researcher.

E.4 Bio-chemical Graphs

Biochemical graphs are related to biology and chemistry domains. The Protein-Protein Interaction (PPI) [137] dataset contains 24 biological graphs where nodes are proteins and edges are the interactions between proteins. In MUTAG [138], each graph indicates a nitro compounds, and its label denotes whether they are aromatic or heteroaromatic. PRO-TEINS [139] and D&D [140] are also protein datasets, where graphs indicate proteins, and labels represent whether they are enzymes or non-enzymes. The PTC [141] and PTC-MR [142] datasets contain a series of chemical compounds, while the labels indicate whether they are carcinogenic for male and female rats. NCI-1 and NCI-109 [143] also consist of chemical compounds, labeled as to whether they are active to hinder the growth of human cancer cell lines. The Open Graph Benchmark (OGB) molecule property prediction benchmark [136] contains 8 molecule graph datasets from different sources, including BBBP, Tox21, ToxCast, SIDER, ClinTox, MUV, HIB, and BACE. In these datasets, the graphs indicate different types of molecules, while the labels express their specific properties. For detailed information of the domains please refer to [136].

APPENDIX F
OPEN-SOURCE IMPLEMENTATIONS

We collect the implementations of graph SSL approaches reviewed in this survey if there exists an open-source code for this method. The hyperlinks of the source codes are provided in Table 9.

APPENDIX G
OTHER GRAPH SSL APPLICATIONS

For zero-shot expert linking problem in the field of expert finding, COAD [152] leverages a same-scale contrastive learning framework to pre-train the expert encoder model. SLAPS [153] integrates a denoising node feature generation task into a classification model for graph structure learning. SCRL [154] applies a prototype-based pretext tasks for few-label graph learning. SDGE [155] is a community detection approach where the model is trained by a same-scale contrastive learning objective. Aiming to repair program from diagnostic feedback, DrRepair [156] pre-trains the model by a repaired line prediction task which is learned with the automatically corrupted programs from online dataset. C-SWM [157] introduces a same-scale node-level contrastive learning strategy to train structured world models that simulate multi-object systems. Sehanobish et al. [158] use a clustering-based auxiliary property classification task to train a GAT [2] model and regard the learned edge weights as the edge feature for downstream tasks learned on biological datasets including SARS-CoV-2 and COVID-19. To learn representations for medical images, Sun et al. [159] apply a context-based hybrid contrastive learning model that maximizes patch- and graph- level agreement on anatomy-aware graphs extracted from medical images. For federated learning [160], [161], FedGL [162] introduces an auxiliary property classification task to provide a global view for the local training in clients. Specifically, the pseudo labels for each node is acquired by a weighted average fusion of clients' prediction results in a server.

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017, pp. 1–14.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018, pp. 1–12.

[3] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019, pp. 1–17.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE TNNLS*, vol. 32, no. 1, pp. 4–24, 2020.

[5] Z. Li, X. Shen, Y. Jiao, X. Pan, P. Zou, X. Meng, C. Yao, and J. Bu, "Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications," in *ICDE*. IEEE, 2020.

[6] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *IJCAI*, 2019.

[7] Q. Liu, M. Allamanis, M. Brockschmidt, and A. L. Gaunt, "Constrained graph variational autoencoders for molecule design," in *NeurIPS*, vol. 31, 2018, pp. 7806–7815.

[8] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE TNNLS*, 2021.

[9] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "GPT-GNN: Generative pre-training of graph neural networks," in *SIGKDD*, 2020, pp. 1857–1867.

[10] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. WEI, W. Huang, and J. Huang, "Self-supervised graph transformer on large-scale molecular data," in *NeurIPS*, vol. 33, 2020, pp. 12 559–12 571.

[11] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020, pp. 1–17.

[12] M. Zhang, L. Hu, C. Shi, and X. Wang, "Adversarial label-flipping attack and defense for graph neural networks," in *ICDM*. IEEE, 2020, pp. 791–800.

[13] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2019, pp. 1–17.

[14] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *ICML*. PMLR, 2020.

[15] 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 *SIGKDD*, 2020, pp. 1150–1160.

[16] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *ICLR*, 2020, pp. 1–22.

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

[18] N. Jovanović, Z. Meng, L. Faber, and R. Wattenhofer, "Towards robust graph contrastive learning," in *WWW Workshop*, 2021.

[19] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE TPAMI*, 2020.

[20] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016, pp. 2536–2544.

[21] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV*. Springer, 2016, pp. 649–666.

[22] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *ECCV*. Springer, 2016, pp. 69–84.

[23] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020, pp. 9729–9738.

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

[25] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - a new approach to self-supervised learning," in *NeurIPS*, vol. 33, 2020, pp. 21 271–21 284.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, vol. 26, 2013, pp. 3111–3119.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.

[29] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *NeurIPS*, vol. 32, 2019.

[30] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.

[31] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.

[32] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *NeurIPS Workshop*, 2016, pp. 1–3.

[33] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep Graph Contrastive Representation Learning," in *ICML Workshop*, 2020.

[34] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE TKDE*, 2021.

[35] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2021.

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

[37] L. Wu, H. Lin, Z. Gao, C. Tan, S. Li *et al.*, "Self-supervised on graphs: Contrastive, generative, or predictive," *arXiv:2105.07342*, 2021.

[38] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *NeurIPS*, 2020.

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

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

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

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

[43] 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," in *WWW Workshop*, 2021.

[44] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized graph autoencoder for graph clustering," in *CIKM*, 2017.

[45] F. Manessi and A. Rozza, "Graph-based neural network models with multiple self-supervised auxiliary tasks," *Pattern Recognition Letters*, vol. 148, pp. 15–21, 2021.

[46] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *ICCV*, 2019, pp. 6519–6528.

[47] E. Hajiramezanali, A. Hasanzadeh, N. Duffield, K. Narayanan, M. Zhou, and X. Qian, "Semi-implicit graph variational autoencoders," in *NeurIPS*, vol. 32, 2019, pp. 10712–10723.

[48] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *IJCAI*, 2018, pp. 2609–2615.

[49] D. Kim and A. Oh, "How to find your friendly neighborhood: Graph attention design with self-supervision," in *ICLR*, 2021.

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

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

[52] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, 1987.

[53] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010.

[54] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014, pp. 1–14.

[55] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, vol. 2.   MIT Press, 2014, p. 2672–2680.

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

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

[58] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *WSDM*, 2021, pp. 148–156.

[59] G. Karypis and V. Kumar, "Multilevel graph partitioning schemes," in *ICPP*, 1995, pp. 113–122.

[60] Z. Lin, Z. Kang, L. Zhang, and L. Tian, "Multi-view attributed graph clustering," *IEEE TKDE*, 2021.

[61] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *ECCV*, 2018, pp. 132–149.

[62] Z. Kang, Z. Lin, X. Zhu, and W. Xu, "Structured graph learning for scalable subspace clustering: From single view to multiview," *IEEE TCYB*, 2021.

[63] S. E. Schaeffer, "Graph clustering," *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007.

[64] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[65] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *ICLR*, 2019.

[66] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *ECCV*.   Springer, 2020, pp. 776–794.

[67] X. Wang and G.-J. Qi, "Contrastive learning with stronger augmentations," *arXiv:2104.07713*, 2021.

[68] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, and S. Pan, "Multi-scale contrastive siamese networks for self-supervised graph representation learning," in *IJCAI*, 2021.

[69] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *WWW*, 2021.

[70] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *ICML*.   PMLR, 2021.

[71] F. L. Opolka, A. Solomon, C. Cangea, P. Veličković, P. Liò, and R. D. Hjelm, "Spatio-temporal deep graph infomax," in *ICLR Workshop*, 2019, pp. 1–6.

[72] Y. Ren, B. Liu, C. Huang, P. Dai, L. Bo, and J. Zhang, "HDGI: An unsupervised graph neural network for representation learning in heterogeneous graph," in *AAAI Workshop*, 2020.

[73] H. Zhang, S. Lin, W. Liu, P. Zhou, J. Tang, X. Liang, and E. P. Xing, "Iterative graph self-distillation," in *WWW Workshop*, 2021.

[74] J. Zeng and P. Xie, "Contrastive self-supervised learning for graph classification," in *AAAI*, vol. 35, no. 12, 2021.

[75] S. Suresh, P. Li, C. Hao, and J. Neville, "Adversarial graph augmentation to improve graph contrastive learning," in *NeurIPS*, 2021.

[76] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *NeurIPS*, vol. 32, 2019.

[77] Y. Jiao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu, "Sub-graph contrast for scalable self-supervised graph representation learning," in *ICDM*.   IEEE, 2020, pp. 222–231.

[78] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, p. 1025–1035.

[79] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *WWW*, 2015.

[80] D. Hwang, J. Park, S. Kwon, K. Kim, J.-W. Ha, and H. J. Kim, "Self-supervised auxiliary learning with meta-paths for heterogeneous graphs," in *NeurIPS*, vol. 33, 2020, pp. 10294–10305.

[81] H. Hafidi, M. Ghogho, P. Ciblat, and A. Swami, "GraphCL: Contrastive self-supervised learning of graph representations," *arXiv:2007.08025*, 2020.

[82] S. Wan, Y. Zhan, L. Liu, B. Yu, S. Pan, and C. Gong, "Contrastive graph poisson networks: Semi-supervised learning with extremely limited labels," in *NeurIPS*, 2021.

[83] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," in *SIGKDD*, 2021, pp. 1726–1736.

[84] S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Veličković, and M. Valko, "Bootstrapped representation learning on graphs," in *ICLR Workshop*, 2021.

[85] Z. T. Kefato and S. Girdzijauskas, "Self-supervised graph neural networks without explicit negative sampling," in *WWW Workshop*, 2021.

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

[87] P. Bielak, T. Kajdanowicz, and N. V. Chawla, "Graph Barlow Twins: A self-supervised representation learning framework for graphs," *arXiv:2106.02466*, 2021.

[88] X. Chen and K. He, "Exploring simple siamese representation learning," in *CVPR*, 2021, pp. 15750–15758.

[89] V. Verma, T. Luong, K. Kawaguchi, H. Pham, and Q. Le, "Towards domain-agnostic contrastive learning," in *ICML*.   PMLR, 2021, pp. 10530–10541.

[90] Y. Ren, J. Bai, and J. Zhang, "Label contrastive coding based graph neural network for graph classification," in *Database Systems for Advanced Applications*, 2021, pp. 123–140.

[91] C. Mavromatis and G. Karypis, "Graph infoclust: Maximizing coarse-grain mutual information in graphs," in *PAKDD*.   Springer, 2021, pp. 541–553.

[92] Y. Ren and B. Liu, "Heterogeneous deep graph infomax," in *AAAI Workshop*, 2020, pp. 1–6.

[93] X. Li, D. Ding, B. Kao, Y. Sun, and N. Mamoulis, "Leveraging meta-path contexts for classification in heterogeneous information networks," in *ICDE*.   IEEE, 2021, pp. 912–923.

[94] C. Park, D. Kim, J. Han, and H. Yu, "Unsupervised attributed multiplex network embedding," in *AAAI*, vol. 34, no. 04, 2020.

[95] Q. Zhu, Y. Xu, H. Wang, C. Zhang, J. Han, and C. Yang, "Transfer learning of graph neural networks with ego-graph information maximization," in *WWW Workshop*, 2021.

[96] P. Wang, K. Agarwal, C. Ham, S. Choudhury, and C. K. Reddy, "Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks," in *WWW*, 2021.

[97] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, "InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *ICLR*, 2020.

[98] J. D. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, "Contrastive learning with hard negative samples," in *ICLR*, 2021, pp. 1–29.

[99] J. Cao, X. Lin, S. Guo, L. Liu, T. Liu, and B. Wang, "Bipartite graph embedding via mutual information maximization," in *WSDM*, 2021, pp. 635–643.

[100] C. Wang and Z. Liu, "Learning graph representation by aggregating subgraphs via mutual information maximization," *arXiv:2103.13125*, 2021.

[101] A. Subramonian, "Motif-driven contrastive learning of graph representations," in *AAAI*, vol. 35, no. 18, 2021, pp. 15 980–15 981.

[102] Q. Sun, J. Li, H. Peng, J. Wu, Y. Ning, P. S. Yu, and L. He, "SUGAR: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism," in *WWW*, 2021.

[103] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic, "On mutual information maximization for representation learning," in *ICLR*, 2020, pp. 1–16.

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

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

[106] S. Wan, S. Pan, J. Yang, and C. Gong, "Contrastive and generative graph convolutional networks for graph-based semi-supervised learning," in *AAAI*, vol. 35, no. 11, 2021, pp. 10 049–10 057.

[107] X. Fan, M. Gong, Y. Wu, and H. Li, "Maximizing mutual information across feature and topology views for learning graph representations," *arXiv:2105.06715*, 2021.

[108] M. Xu, H. Wang, B. Ni, H. Guo, and J. Tang, "Self-supervised graph-level representation learning with local and global structure," in *ICML*. PMLR, 2021.

[109] B. Jing, C. Park, and H. Tong, "HDMI: High-order deep multiplex infomax," in *WWW*, 2021, pp. 2414–2424.

[110] Y. Zheng, M. Jin, S. Pan, Y.-F. Li, H. Peng, M. Li, and Z. Li, "Towards graph self-supervised learning with contrastive adjusted zooming," *arXiv:2111.10698*, 2021.

[111] K. K. Roy, A. Roy, A. Rahman, M. A. Amin, and A. A. Ali, "Node embedding using mutual information and self-supervision based bi-level aggregation," in *IJCNN*, 2021.

[112] S. Kou, W. Xia, X. Zhang, Q. Gao, and X. Gao, "Self-supervised graph convolutional clustering by preserving latent distribution," *Neurocomputing*, vol. 437, pp. 218–226, 2021.

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

[114] B. Hao, J. Zhang, H. Yin, C. Li, and H. Chen, "Pre-training graph neural networks for cold-start users and items representation," in *WSDM*, 2021, pp. 265–273.

[115] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung, and X. Zhang, "Self-supervised multi-channel hypergraph convolutional network for social recommendation," in *WWW*, 2021, pp. 413–424.

[116] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," in *AAAI*, vol. 35, no. 5, 2021.

[117] Z. Liu, Y. Ma, Y. Ouyang, and Z. Xiong, "Contrastive learning for recommender system," *arXiv:2101.01317*, 2021.

[118] Y. Liu, S. Yang, C. Lei, G. Wang, H. Tang, J. Zhang, A. Sun, and C. Miao, "Pre-training graph transformer with multimodal side information for recommendation," in *ACM Multimedia*, 2021.

[119] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly detection on attributed networks via contrastive self-supervised learning," *IEEE TNNLS*, 2021.

[120] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *SDM*. SIAM, 2019, pp. 594–602.

[121] Y. Li, X. Huang, J. Li, M. Du, and N. Zou, "SpecAE: Spectral autoencoder for anomaly detection in attributed networks," in *CIKM*, 2019, pp. 2233–2236.

[122] K. Ding, J. Li, N. Agarwal, and H. Liu, "Inductive anomaly detection on attributed networks," in *IJCAI*, 2020, pp. 1288–1294.

[123] M. Jin, Y. Liu, Y. Zheng, L. Chi, Y.-F. Li, and S. Pan, "Anemone: Graph anomaly detection with multi-scale contrastive learning," in *CIKM*, 2021, pp. 3122–3126.

[124] Y. Zheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, and Y.-P. P. Chen, "Generative and contrastive self-supervised learning for graph anomaly detection," *IEEE TKDE*, 2021.

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

[126] Y. Wang, J. Wang, Z. Cao, and A. B. Farimani, "MolCLR: Molecular contrastive learning of representations via graph neural networks," *arXiv:2102.10056*, 2021.

[127] Y. Fang, H. Yang, X. Zhuang, X. Shao, X. Fan, and H. Chen, "Knowledge-aware contrastive molecular graph learning," *arXiv:2103.13047*, 2021.

[128] S. Cheng, L. Zhang, B. Jin, Q. Zhang, X. Lu, M. You, and X. Tian, "GraphMS: Drug target prediction using graph representation learning with substructures," *Applied Sciences*, 2021.

[129] Y. Wang, Y. Min, X. Chen, and J. Wu, "Multi-view graph contrastive representation learning for drug-drug interaction prediction," in *WWW*, 2021, pp. 2921–2933.

[130] D. Jin, Z. Yu, P. Jiao, S. Pan, P. S. Yu, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE TKDE*, 2021.

[131] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NeurIPS*, 2018, p. 4805–4815.

[132] Y. G. Wang, M. Li, Z. Ma, G. Montufar, X. Zhuang, and Y. Fan, "Haar graph pooling," in *ICML*. PMLR, 2020, pp. 9952–9962.

[133] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *SIGKDD*, 2015, pp. 1365–1374.

[134] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *NeurIPS*, 2018.

[135] P. Mernyei and C. Cangea, "Wiki-CS: A wikipedia-based benchmark for graph neural networks," in *ICML Workshop*, 2020.

[136] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open Graph Benchmark: Datasets for machine learning on graphs," in *NeurIPS*, vol. 33, 2020, pp. 22 118–22 133.

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

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

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

[140] 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.

[141] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma, "Statistical evaluation of the predictive toxicology challenge 2000–2001," *Bioinformatics*, vol. 19, no. 10, pp. 1183–1193, 2003.

[142] C. Helma, R. D. King, S. Kramer, and A. Srinivasan, "The predictive toxicology challenge 2000–2001," *Bioinformatics*, 2001.

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

[144] I. F. Martins, A. L. Teixeira, L. Pinheiro, and A. Falcao, "A Bayesian approach to in silico blood-brain barrier penetration modeling," *JCIM*, vol. 52, no. 6, pp. 1686–1697, 2012.

[145] Tox21, "Tox21 data challenge 2014," 2014. [Online]. Available: https://tripod.nih.gov/tox21/challenge/

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

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

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

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

[150] "AIDS antiviral screen data," 2017. [Online]. Available: http://wiki.nci.nih.gov/display/NCIDTPdata/AIDS

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

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

[153] B. Fatemi, L. E. Asri, and S. M. Kazemi, "SLAPS: Self-supervision improves structure learning for graph neural networks," in *NeurIPS*, 2021.

[154] C. Liu, L. Wen, Z. Kang, G. Luo, and L. Tian, "Self-supervised consensus representation learning for attributed graph," in *ACM Multimedia*, 2021, pp. 2654–2662.

[155] S. Xu, S. Liu, and L. Feng, "Self-supervised deep graph embedding with high-order information fusion for community discovery," *arXiv:2102.03302*, 2021.

[156] M. Yasunaga and P. Liang, "Graph-based, self-supervised program repair from diagnostic feedback," in *ICML*. PMLR, 2020.

[157] T. Kipf, E. van der Pol, and M. Welling, "Contrastive learning of structured world models," in *ICLR*, 2020, pp. 1–21.

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

[159] L. Sun, K. Yu, and K. Batmanghelich, "Context matters: Graph-based self-supervised representation learning for medical images," in *AAAI*, vol. 35, no. 6, May 2021, pp. 4874–4882.

[160] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "Fedproto: Federated prototype learning over heterogeneous devices," in *AAAI*, 2022.

[161] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017, pp. 1273–1282.

[162] C. Chen, W. Hu, Z. Xu, and Z. Zheng, "FedGL: Federated graph learning framework with global self-supervision," *arXiv:2105.03170*, 2021.

Chuan Zhou obtained Ph.D. degree from Chinese Academy of Sciences in 2013. He won the outstanding doctoral dissertation of Chinese Academy of Sciences in 2014, the best paper award of ICCS-14, and the best student paper award of IJCNN-17. Currently, he is an Associate Professor at the Academy of Mathematics and Systems Science, Chinese Academy of Sciences. His research interests include social network analysis and graph mining. To date, he has published more than 70 papers, including IEEE TKDE, ICDM, AAAI, CIKM, IJCAI and WWW.

Yu Zheng received the B.S. and M.S. degrees in computer science from Northwest A&F University, China, in 2008 and 2011, respectively. She is currently pursuing her Ph.D. degree in computer science at La Trobe University, Melbourne, Australia. Her research interests include image classification, data mining, and machine learning.

Feng Xia (M'07-SM'12) received the BSc and PhD degrees from Zhejiang University, Hangzhou, China. He was Full Professor and Associate Dean (Research) in School of Software, Dalian University of Technology, China. He is Associate Professor and former Discipline Leader (IT) in School of Engineering, IT and Physical Sciences, Federation University Australia. Dr. Xia has published 2 books and over 300 scientific papers in international journals and conferences. His research interests include data science, artificial intelligence, graph learning, and systems engineering. He is a Senior Member of IEEE and ACM.

Yixin Liu received the B.S. degree and M.S. degree from Beihang University, Beijing, China, in 2017 and 2020, respectively. He is currently pursuing his Ph.D. degree in computer science at Monash University, Australia. His research concentrates on data mining, machine learning, and deep learning on graphs.

Ming Jin received the B.Eng. degree from the Hebei University of Technology, Tianjin, China, in 2017, and M.Inf.Tech. degree from the University of Melbourne, Melbourne, Australia, in 2019. He is currently pursuing his Ph.D. degree in computer science at Monash University, Melbourne, Australia. His research focuses on graph neural networks (GNNs), time series analyse, data mining, and machine learning.

Philip S. Yu received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA. He is currently a Distinguished Professor of computer science with the University of Illinois at Chicago, Chicago, IL, USA, where he is also the Wexler Chair in Information Technology. He has published more than 830 articles in refereed journals and conferences. He holds or has applied for more than 300 U.S. patents. His research interests include big data, data mining, data streams, databases, and privacy. Dr. Yu is a fellow of the ACM and IEEE.

Shirui Pan received a Ph.D. in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia. He is currently an ARC Future Fellow and Senior Lecturer with the Faculty of Information Technology, Monash University, Australia. His research interests include data mining and machine learning. To date, Dr. Pan has published over 150 research papers in top-tier journals and conferences, including the TPAMI, TKDE, TNNLS, NeurIPS, ICML, and KDD. He is recognised as one of the *AI 2000 AAAI/IJCAI Most Influential Scholars* in Australia (2021).