Review

A Comprehensive Survey on Deep Graph Representation Learning

Wei Ju [a], Zheng Fang [b], Yiyang Gu [a], Zequn Liu [a], Qingqing Long [c], Ziyue Qiao [d], Yifang Qin [a], Jianhao Shen [a], Fang Sun [e], Zhiping Xiao [e], Junwei Yang [a], Jingyang Yuan [a], Yusheng Zhao [a], Yifan Wang [f], Xiao Luo [e,*], Ming Zhang [a,*]

[a] School of Computer Science, National Key Laboratory for Multimedia Information Processing, Peking University, Beijing, 100871, China
[b] School of Intelligence Science and Technology, Peking University, Beijing, 100871, China
[c] Computer Network Information Center, Chinese Academy of Sciences, Beijing, 100086, China
[d] Artificial Intelligence Thrust, The Hong Kong University of Science and Technology, Guangzhou, 511453, China
[e] Department of Computer Science, University of California, Los Angeles, 90095, USA
[f] School of Information Technology & Management, University of International Business and Economics, Beijing, 100029, China

ABSTRACT

Graph representation learning aims to effectively encode high-dimensional sparse graph-structured data into low-dimensional dense vectors, which is a fundamental task that has been widely studied in a range of fields, including machine learning and data mining. Classic graph embedding methods follow the basic idea that the embedding vectors of interconnected nodes in the graph can still maintain a relatively close distance, thereby preserving the structural information between the nodes in the graph. However, this is sub-optimal due to: (i) traditional methods have limited model capacity which limits the learning performance; (ii) existing techniques typically rely on unsupervised learning strategies and fail to couple with the latest learning paradigms; (iii) representation learning and downstream tasks are dependent on each other which should be jointly enhanced. With the remarkable success of deep learning, deep graph representation learning has shown great potential and advantages over shallow (traditional) methods, there exist a large number of deep graph representation learning techniques have been proposed in the past decade, especially graph neural networks. In this survey, we conduct a comprehensive survey on current deep graph representation learning algorithms by proposing a new taxonomy of existing state-of-the-art literature. Specifically, we systematically summarize the essential components of graph representation learning and categorize existing approaches by the ways of graph neural network architectures and the most recent advanced learning paradigms. Moreover, this survey also provides the practical and promising applications of deep graph representation learning. Last but not least, we state new perspectives and suggest challenging directions which deserve further investigations in the future.

1. Introduction

Graphs have recently emerged as a powerful tool for representing a variety of structured and complex data, including social networks, traffic networks, information systems, knowledge graphs, protein–protein interaction networks, and physical interaction networks. As a kind of general form of data organization, graph structures are capable of naturally expressing the intrinsic relationship of these data, and thus can characterize plenty of non-Euclidean structures that are crucial in a variety of disciplines and domains due to their flexible adaptability. For example, to encode a social network as a graph, nodes on the graph are used to represent individual users, and edges are used to represent the relationship between two individuals, such as friends. In the field of biology, nodes can be used to represent proteins, and edges can be

used to represent biological interactions between various proteins, such as the dynamic interactions between proteins. Thus, by analyzing and mining the graph-structured data, we can understand the deep meaning hidden behind the data, and further discover valuable knowledge, so as to benefit society and human beings.

In the last decade years, a wide range of machine learning algorithms have been developed for graph-structured data learning. Among them, traditional graph kernel methods (Gärtner, Flach, & Wrobel, 2003; Kashima, Tsuda, & Inokuchi, 2003; Shervashidze, Schweitzer, Van Leeuwen, Mehlhorn, & Borgwardt, 2011a; Shervashidze, Vishwanathan, Petri, Mehlhorn, & Borgwardt, 2009) usually break down graphs into different atomic substructures and then use kernel functions to measure the similarity between all pairs of them. Although

---

* Corresponding authors.
  E-mail addresses: xiaoluo@cs.ucla.edu (X. Luo), mzhang_cs@pku.edu.cn (M. Zhang).

graph kernels could provide a perspective on modeling graph topology, these approaches often generate substructures or feature representations based on given hand-crafted criteria. These rules are rather heuristic, prone to suffer from high computational complexity, and therefore have weak scalability and subpar performance.

In the past few years, graph embedding algorithms (Ahmed, Shervashidze, Narayanamurthy, Josifovski, & Smola, 2013; Grover & Leskovec, 2016; Perozzi, Al-Rfou, & Skiena, 2014; Tang, Qu and Mei, 2015; Tang et al., 2015; Wang, Cui, & Zhu, 2016) have ever-increasing emerged, which attempt to encode the structural information of the graph (usually a high-dimensional sparse matrix) and map it into a low-dimensional dense vector embedding to preserve the topology information and attribute information in the embedding space as much as possible, so that the learned graph embeddings can be naturally integrated into traditional machine learning algorithms. Compared to previous works which use feature engineering in the pre-processing phase to extract graph structural features, current graph embedding algorithms are conducted in a data-driven way leveraging machine learning algorithms (such as neural networks) to encode the structural information of the graph. Specifically, existing graph embedding methods can be categorized into the following main groups: (i) matrix factorization based methods (Ahmed et al., 2013; Cao, Lu, & Xu, 2015; Ou, Cui, Pei, Zhang, & Zhu, 2016) that factorize the matrix to learn node embedding which preserves the graph property; (ii) deep learning based methods (Grover & Leskovec, 2016; Perozzi et al., 2014; Tang, Qu, Wang et al., 2015; Wang et al., 2016) that apply deep learning techniques specifically designed for graph-structured data; (iii) edge reconstruction based methods (Liu, Cheung, Li, & Liao, 2016; Man, Shen, Liu, Jin, & Cheng, 2016; Tang, Qu and Mei, 2015) that either maximizes edge reconstruction probability or minimizes edge reconstruction loss. Generally, these methods typically depend on shallow architectures, and fail to exploit the potential and capacity of deep neural networks, resulting in sub-optimal representation quality and learning performance.

Inspired by the recent remarkable success of deep neural networks, a range of deep learning algorithms has been developed for graph-structured data learning. The core of these methods is to generate effective node and graph representations using graph neural networks (GNNs), followed by a goal-oriented learning paradigm. In this way, the derived representations can be adaptively coupled with a variety of downstream tasks and applications. Following this line of thought, in this paper, we propose a new taxonomy to classify the existing graph representation learning algorithms, i.e., graph neural network architectures, learning paradigms, and various promising applications, as shown in Fig. 1. Specifically, for the architectures of GNNs, we investigate the studies on graph convolutions, graph kernel neural networks, graph pooling, and graph transformer. For the learning paradigms, we explore three advanced types namely supervised/semi-supervised learning on graphs, graph self-supervised learning, and graph structure learning. To demonstrate the effectiveness of the learned graph representations, we provide several promising applications to build tight connections between representation learning and downstream tasks, such as social analysis, molecular property prediction and generation, recommender systems, and traffic analysis. Last but not least, we present some perspectives for thought and suggest challenging directions that deserve further study in the future.

Differences between this survey and existing ones. Up to now, there exist some other overview papers focusing on different perspectives of graph representation learning (Bacciu, Errica, Micheli, & Podda, 2020; Chami, Abu-El-Haija, Perozzi, Ré, & Murphy, 2022; Chen, Wang, Wang & Kuo, 2020; Chen et al., 2022; Khoshraftar & An, 2022; Wu et al., 2020; Xia et al., 2021; Zhang, Cui and Zhu, 2020; Zhou et al., 2020, 2022) that are closely related to ours. However, there are very few comprehensive reviews have summarized deep graph representation learning simultaneously from the perspective of diverse GNN architectures and corresponding up-to-date learning paradigms. Therefore,

we here clearly state their distinctions from our survey as follows. There have been several surveys on classic graph embedding (Cai, Zheng, & Chang, 2018; Goyal & Ferrara, 2018), these works categorize graph embedding methods based on different training objectives. Wang et al. (2022) goes further and provides a comprehensive review of existing heterogeneous graph embedding approaches. With the rapid development of deep learning, there are a handful of surveys along this line. For example, Wu et al. (2020) and Zhang, Cui et al. (2020) mainly focus on several classical and representative GNN architectures without exploring deep graph representation learning from a view of the most recent advanced learning paradigms such as graph self-supervised learning and graph structure learning. Chami et al. (2022) and Xia et al. (2021) jointly summarize the studies of graph embeddings and GNNs. Zhou, Cui et al. (2020) explores different types of computational modules for GNNs. One recent survey under review (Khoshraftar & An, 2022) categorizes the existing works in graph representation learning from both static and dynamic graphs. However, these taxonomies emphasize the basic GNN methods but pay insufficient attention to the learning paradigms, and provide few discussions of the most promising applications, such as recommender systems as well as molecular property prediction and generation. To the best of our knowledge, the most relevant survey published formally is Zhou, Zheng et al. (2022), which presents a review of GNN architectures and roughly discusses the corresponding applications. Nevertheless, this survey merely covers methods up to the year of 2020, missing the latest developments in the past three years.

Therefore, it is highly desired to summarize the representative GNN methods, the most recent advanced learning paradigms, and promising applications into one unified and comprehensive framework. Moreover, we strongly believe this survey with a new taxonomy of literature and more than 500 studies will strengthen future research on deep graph representation learning.

Contribution of this survey. The goal of this survey is to systematically review the literature on the advances of deep graph representation learning and discuss further directions. It aims to help the researchers and practitioners who are interested in this area, and support them in understanding the panorama and the latest developments of deep graph representation learning. The key contributions of this survey are summarized as follows:

- **Systematic Taxonomy.** We propose a systematic taxonomy to organize the existing deep graph representation learning approaches based on the ways of GNN architectures and the most recent advanced learning paradigms via providing some representative branches of methods. Moreover, several promising applications are presented to illustrate the superiority and potential of graph representation learning.
- **Comprehensive Review.** For each branch of this survey, we review the essential components and provide detailed descriptions of representative algorithms, and systematically summarize the characteristics to make the overview comparison.
- **Future Directions.** Based on the properties of existing deep graph representation learning algorithms, we discuss the limitations and challenges of current methods and propose the potential as well as promising research directions deserving of future investigations.

2. Background

In this section, we first briefly introduce some definitions in deep graph representation learning that need to be clarified, and then we explain the reasons why we need graph representation learning.
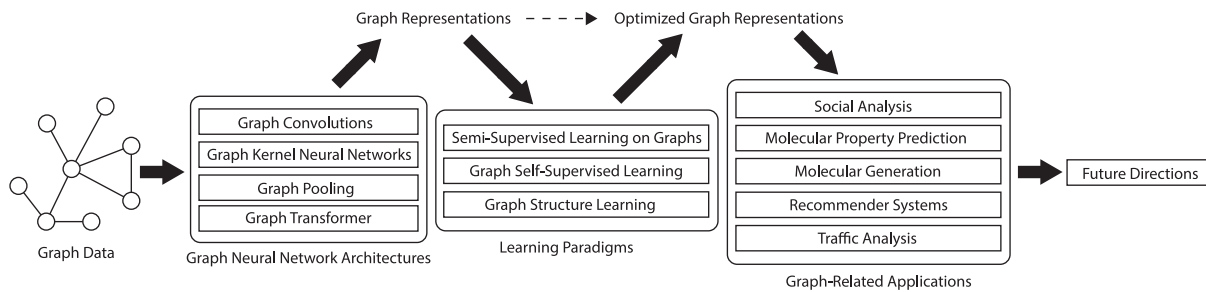
Fig. 1. The architecture of this paper.

2.1. Problem definition

Definition: Graph. Given a graph $G = (V, E, \mathbf{X})$, where $V = \{v_1, \ldots, v_{|V|}\}$ is the set of nodes, $E = \{e_1, \ldots, e_{|V|}\}$ is the set of edges, and the edge $e = (v_i, v_j) \in E$ represent the connection relationship between nodes v_i and v_j in the graph. $\mathbf{X} \in \mathbb{R}^{|V| \times M}$ is the node feature matrix with M being the dimension of each node feature. The adjacency matrix of a graph can be defined as $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$, where $\mathbf{A}_{ij} = 1$ if $(v_i, v_j) \in E$, otherwise $\mathbf{A}_{ij} = 0$.

The adjacency matrix can be regarded as the structural representation of the graph-structured data, in which each row of the adjacency matrix $\mathbf{A}$ represents the connection relationship between the corresponding node of the row and all other nodes, which can be regarded as a discrete representation of the node. However, in real-life circumstances, the adjacency matrix $\mathbf{A}$ corresponding to G is a highly sparse matrix, and if $\mathbf{A}$ is used directly as node representations, it will be seriously affected by impractical storage demands and computational overhead. The storage space of the adjacency matrix $\mathbf{A}$ is $|V| \times |V|$, which is usually unacceptable when the total number of nodes grows to the order of millions. At the same time, the value of most dimensions in the node representation is 0. The sparsity will make subsequent machine learning tasks very difficult.

Graph representation learning is a bridge between the original input data and the task objectives in the graph. The fundamental idea of the graph representation learning algorithm is first to learn the embedded representations of nodes or the entire graph from the input graph structure data and then apply these embedded representations to downstream related tasks, such as node classification, graph classification, link prediction, community detection, and visualization, etc. Specifically, it aims to learn low-dimensional, dense distributed embedding representations for nodes in the graph. Formally, the goal of graph representation learning is to learn its embedding vector representation $R_v \in \mathbb{R}^d$ for each node $v \in V$, where the dimension d of the vector is much smaller than the total number of nodes $|V|$ in the graph.

2.2. Traditional graph embedding

Traditional graph embedding learning methods, as part of dimensionality reduction techniques, aimed to embed graph data into a lower-dimensional vector space with the idea that connected nodes in the graph should still be closer to each other in this lower-dimensional space, thereby preserving the structural information between nodes in the graph. Influenced by classical dimensionality reduction techniques, early graph embedding methods are primarily inspired by classic matrix factorization techniques (Belkin & Niyogi, 2001) and multi-dimensional scaling (Kruskal, 1964). The following three sections describe these methods in more detail, distinguishing among matrix factorization-based methods, random walks-based methods and other non-GNN deep methods. In Table 1, we summarize different categories of traditional graph embedding methods.

2.2.1. Matrix factorization-based methods

Matrix factorization-based methods are the early endeavors in graph embedding learning. These approaches can be outlined in a two-step process. In the initial step, a proximity-based matrix is constructed for the graph, where each element of the matrix represents the proximity measure between two nodes in the graph. Subsequently, a dimensionality reduction technique is employed on this matrix in the second step to generate the node embeddings.

Locally Linear Embedding (LLE) (Roweis & Saul, 2000). LLE assumes that node representations are sampled from the same manifold space, and any node in the graph and its neighboring nodes are located in a local region of that manifold space. Therefore, node representations can be obtained by linearly combining them with their neighboring nodes. LLE first constructs a local reconstruction weight matrix, W_{ij}, for nodes in the graph to linearly combine neighboring nodes. By computing the distance between the linear combination and the central node, the problem is reduced to solving for matrix eigenvalues to learn low-dimensional vector representations for nodes. The objective function is computed as follows:

$$\phi(Z) = \frac{1}{2} \sum_i |z_i - \sum_{j \in N_i} W_{ij} z_j|^2, \tag{1}$$

where z_i represents the low-dimensional representation of the ith node, and N_i is the set of neighboring nodes for the central node i.

Laplacian Eigenmaps (LE) (Anderson & Morley, 1985). LE believes that nodes directly connected in graph data should be kept as close as possible in the embedding space. Specifically, it achieves this by defining the distance between connected nodes in the embedding space using the square of the Euclidean distance. It transforms the final optimization objective into the computation of the Laplacian matrix's eigenvectors. The objective function is computed as follows:

$$\phi(Z) = \frac{1}{2} \sum_{i,j} |z_i - z_j|^2 W_{ij} = Z^T L Z, \quad \text{s.t.} \quad Z^T D Z = I, \tag{2}$$

where W_{ij} represents the connection weight between nodes i and j in the graph. After linear transformation, the optimization of $\phi(Z)$ can be reformulated as $Z^T L Z$, where $L = D - W$ is the constructed graph Laplacian matrix, and D is a symmetric matrix.

Graph Factorization (GF) (Ahmed et al., 2013). The matrix eigenvector-based methods mentioned before consider the similarity between nodes throughout the entire graph, which can result in excellent node feature representations. However, with the ever-growing scale of real-world graph data, computing matrix eigenvectors for large graphs can be computationally expensive and memory-intensive. GF introduces a graph embedding method with a time complexity of $O(|E|)$ by factorizing the adjacency matrix of the graph. The objective function is as follows:

$$\phi(Z, \lambda) = \frac{1}{2} \sum_{i,j \in E} |W_{i,j} - \langle z_i, z_j \rangle|^2 + \frac{\lambda}{2} \sum_i |z_i|^2, \tag{3}$$

where λ is a regularization coefficient, and $\langle z_i, z_j \rangle$ represents the corresponding inner-product operation. Moreover, these inner-product methods also contain GraRep (Cao et al., 2015) and HOPE (Ou et al., 2016), which consider higher-order and general node similarity respectively.

Table 1
Summary of traditional graph embedding methods.

Type	Method	Similarity measure	Loss function (L)
Matrix factorization	LLE (Roweis & Saul, 2000)	General	$\|z_i - \sum_{j \in N_i} W_{ij} z_j\|^2$
	LE (Anderson & Morley, 1985)	General	$Z^T L Z, \text{s.t.} Z^T D Z = I$
	GF (Ahmed et al., 2013)	$A_{i,j}$	$\|W_{i,j} - \langle z_i, z_j \rangle\|^2$
	GraRep (Cao et al., 2015)	$A_{i,j}, A_{i,j}^2, \ldots, A_{i,j}^k$	$\|W_{i,j} - \langle z_i, z_j \rangle\|^2$
	HOPE (Ou et al., 2016)	General	$\|W_{i,j} - \langle z_i, z_j \rangle\|^2$
Random walk	DeepWalk (Perozzi et al., 2014)	$p(v_i\|v_i)$	$-A_{ij} \log\langle z_i, z_j \rangle$
	Node2vec (Grover & Leskovec, 2016)	$p(v_i\|v_i)$ (biased)	$-A_{ij} \log\langle z_i, z_j \rangle$
	HARP (Chen, Perozzi, Hu, & Skiena, 2018)	$p(v_i\|v_i)$ (biased)	$-A_{ij} \log\langle z_i, z_j \rangle$
	LINE (Tang, Qu, Wang et al., 2015)	Two-order similarities	Corresponding loss
Non-GNN deep	SDNE (Wang et al., 2016)	Two-order proximities	Corresponding loss
	DNGR (Cao, Lu, & Xu, 2016)	Two-order proximities	Corresponding loss

2.2.2. Random walk-based methods

Random walk-based methods have also attracted a lot of attention in graph embedding learning. The basic idea of these methods is to create random walks among nodes in the graph to capture its structural characteristics. Thus, nodes tend to have similar embedding if they co-occur on short random walks. Compared to fixed proximity measures in traditional matrix factorization-based methods, these approaches use co-occurrence in a random walk as a measure of node similarity, which is more flexible and has demonstrated promising performance across various applications.

DeepWalk (Perozzi et al., 2014). DeepWalk analogizes nodes in a graph to words in text. It uses random walks on the graph to generate numerous node sequences $S = \{v_1, \ldots, v_{|s|}\}$, treating these sequences as sentences, and then inputting them into the Word2vec (Mikolov, Chen, Corrado and Dean, 2013), which aims to maximize the probability of node context given the target node v_i. It can be written as:

$$\frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{-t \leq j \leq t, j \neq 0} \log p(v_{i+j}|v_i), \tag{4}$$

where t is the context window size. Compared to matrix factorization-based methods, DeepWalk exhibits extremely low time complexity and is suitable for large-scale graph representation learning. However, DeepWalk only considers local information between nodes in the graph, making it challenging to find the optimal random walk sampling sequences.

Node2vec (Grover & Leskovec, 2016). Based on DeepWalk, Node2vec utilizes parameters p and q to guide the random walk. Parameter p allows the algorithm to revisit previously traversed nodes t, with smaller values of p increasing the likelihood of returning to t. Parameter q facilitates both inward and outward exploration; when $q > 1$, the algorithm tends to visit nodes closer to t; while for $q < 1$, it leans toward nodes farther away from t. Thus, it approximates a balance between two sampling strategies, Depth-First Search (DFS) and Breadth-First Search (BFS), which can better capture topological information in the graph. Moreover, there are also some approaches that leverage graph preprocessing, i.e., HARP (Chen et al., 2018) or different random walk strategies (Perozzi, Kulkarni, Chen, & Skiena, 2017) to extend random walk approaches.

Large-Scale Information Network Embeddings (*LINE*) (Tang, Qu, Wang et al., 2015). LINE is not based on random walks but is often compared with DeepWalk and Node2vec. The method models first-order and second-order similarity between nodes in a graph. First-order similarity characterizes the similarity between two directly connected nodes in the graph and is represented through the joint probability distribution of the two nodes.

$$p(v_i, v_j) = \frac{1}{1 + \exp(-z_i^\top z_j)}, \tag{5}$$

where z_i represents the low-dimensional representation of node v_i. Second-order similarity, on the other hand, describes the number of common neighboring nodes between nodes in the graph, measuring

the similarity of the local neighborhood structures (context) between nodes, which can be expressed as:

$$p(v_j|v_i) = \frac{\exp(z_j'^\top z_i)}{\sum_{k=1}^{|V|} \exp(z_k'^\top \cdot z_i)}, \tag{6}$$

where $|V|$ represents the number of nodes in the graph, and z_j' represents the representation of node v_j when it serves as a specific context.

2.2.3. Non-GNN deep methods

In contrast to the above shallow graph embedding methods, non-GNN deep methods directly integrate the graph's structural information into the encoder algorithm through deep neural networks. The fundamental concept underlying these methods involves employing the autoencoder approach (Hinton & Salakhutdinov, 2006) to compress information related to a node's local neighborhood.

Structural Deep Network Embeddings (SDNE) (Wang et al., 2016). SDNE jointly preserves the first-order and second-order proximities between the nodes in the graph. The second-order proximity of the nodes can be defined as:

$$L_1 = \sum_{v_i \in V} |(x_i - x_i') \odot b_i|, \tag{7}$$

where x_i is the row corresponding to node v_i in the adjacency matrix A, $\odot$ means the Hadamard product. $b_i = \{b_{ij}\}_{j=1}^{|V|}$, if $A_{ij} = 0, b_{ij} = 1$; otherwise, $b_{ij} = \beta > 1$. And the first-order similarity can be defined as:

$$L_2 = \sum_{(v_i, v_j) \in E} A_{ij} |z_i - z_j|, \tag{8}$$

where z_i is the learned representation of node v_i.

Deep Neural Graph Representations (DNGR) (Cao et al., 2016). Similar to SDNE, DNGR utilizes pointwise mutual information between two nodes co-occurring in random walks instead of the adjacency matrix values.

2.3. Why study deep graph representation learning

With the rapid development of deep learning techniques, deep neural networks such as convolutional neural networks and recurrent neural networks have made breakthroughs in the fields of computer vision, natural language processing, and speech recognition. They can well abstract the semantic information of images, natural languages, and speeches. However, current deep learning techniques fail to handle more complex and irregular graph-structured data. To effectively analyze and model this kind of non-Euclidean structure data, many graph representation learning algorithms have emerged in recent years, including graph embedding and graph neural networks. At present, compared with Euclidean-style data such as images, natural language, and speech, graph-structured data is high-dimensional, complex, and irregular. Therefore, the graph representation learning algorithm is a rather powerful tool for studying graph-structured data. To encode

Table 2
Summary of graph convolution methods.

Method	Category	Aggregation	Time complexity
Spectral CNN (Bruna, Zaremba, Szlam, & LeCun, 2013)	Spectral graph convolution	–	$O(n^3)$
Henaff, Bruna, and LeCun (2015)	Spectral graph convolution	–	$O(n^3)$
ChebNet (Defferrard, Bresson, & Vandergheynst, 2016)	Spectral graph convolution	–	$O(m)$
GCN (Kipf & Welling, 2016a)	Spectral/Spatial	Weighted average	$O(m)$
CayleyNet (Levie, Monti, Bresson, & Bronstein, 2018)	Spectral graph convolution	–	$O(m)$
GraphSAGE (Hamilton, Ying, & Leskovec, 2017)	Spatial graph convolution	General	$O(m)$
GAT (Veličković et al., 2017)	Spatial graph convolution	Attentive	$O(m)$
DGCNN (Wang et al., 2019)	Spatial graph convolution	General	$O(m)$
LanzcosNet (Liao, Zhao, Urtasun, & Zemel, 2019)	Spectral graph convolution	–	$O(n^2)$
SGC (Wu et al., 2019)	Spatial graph convolution	Weighted average	$O(m)$
GWNN (Xu, Shen, Cao, Qiu and Cheng, 2019)	Spectral graph convolution	–	$O(m)$
GIN (Xu, Hu, Leskovec and Jegelka, 2018)	Spatial graph convolution	Sum	$O(m)$
GraphAIR (Hu et al., 2020)	Spatial graph convolution	Sum	$O(m)$
PNA (Corso, Cavalleri, Beaini, Liò, & Veličković, 2020)	Spatial graph convolution	Multiple	$O(m)$
S^{2}GC (Zhu & Koniusz, 2021)	Spectral graph convolution	–	$O(m)$
GNNML3 (Balcilar et al., 2021)	Spatial/Spectral	–	$O(m)$
MSGNN (He, Permultter, Reinert, & Cucuringu, 2022)	Spectral graph convolution	–	$O(m)$
EGC (Tailor, Opolka, Lio, & Lane, 2022)	Spatial graph convolution	General	$O(m)$
APPNP (Gasteiger, Bojchevski, & Günnemann, 2018)	Spatial graph convolution	(Approximate) Personalized pagerank	$O(m)$
GCNII (Chen, Wei, Huang, Ding and Li, 2020)	Spatial graph convolution	–	$O(m)$
GATv2 (Brody, Alon, & Yahav, 2021)	Spatial graph convolution	Attentive	$O(m)$

complex graph-structured data, deep graph representation learning needs to meet several characteristics: (1) *topological properties*: Graph representations need to capture the complex topological information of the graph, such as the relationship between nodes and nodes, and other substructure information, such as subgraphs, motif, etc; (2) *feature attributes*: It is necessary for graph representations to describe high-dimensional attribute features in the graph, including the attributes of nodes and edges themselves; (3) *scalability*: Because different real graph data have different characteristics, graph representation learning algorithms should be able to efficiently learn its embedding representation on different graph structure data, making it universal and transferable.

3. Graph convolutions

Graph convolutions have become the basic building blocks in many deep graph representation learning algorithms and graph neural networks developed recently. In this section, we provide a comprehensive review of graph convolutions, which generally fall into two categories: spectral graph convolutions and spatial graph convolutions. Based on the solid mathematical foundations of Graph Signal Processing (GSP) (Hammond, Vandergheynst, & Gribonval, 2011; Sandryhaila & Moura, 2013; Shuman, Narang, Frossard, Ortega, & Vandergheynst, 2013), spectral graph convolutions seek to capture the patterns of the graph in the frequency domain. On the other hand, spatial graph convolutions inherit the idea of message passing from Recurrent Graph Neural Networks (RecGNNs), and they compute node features by aggregating the features of their neighbors. Thus, the computation graph of a node is derived from the local graph structure around it, and the graph topology is naturally incorporated into the way node features are computed. In this section, we first introduce spectral graph convolutions and then spatial graph convolutions, followed by a brief summary. In Table 2, we summarize a number of graph convolutions proposed in recent years.

3.1. Spectral graph convolutions

With the success of Convolutional Neural Networks (CNNs) in computer vision (Krizhevsky, Sutskever, & Hinton, 2017), efforts have been made to transfer the idea of convolution to the graph domain. However, this is not an easy task because of the non-Euclidean nature of graphical data. Graph signal processing (GSP) (Hammond et al., 2011; Sandryhaila & Moura, 2013; Shuman et al., 2013) defines the Fourier Transform on graphs and thus provides a solid theoretical foundation of spectral graph convolutions.

In graph signal processing, a graph signal refers to a set of scalars associated with every node in the graph, *i.e.* $f(v)$, $\forall v \in V$, and it can be written in the n-dimensional vector form $\mathbf{x} \in \mathbb{R}^n$, where n is the number of nodes in the graph. Another core concept of graph signal processing is the symmetric normalized graph Laplacian matrix (or simply, the graph Laplacian), defined as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}$ is the identity matrix, $\mathbf{D}$ is the degree matrix (*i.e.* a diagonal matrix $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$), and $\mathbf{A}$ is the adjacency matrix. In the typical setting of graph signal processing, the graph G is undirected. Therefore, $\mathbf{L}$ is real symmetric and positive semi-definite. This guarantees the eigen decomposition of the graph Laplacian: $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{n-1}]$ is the eigenvectors of the graph Laplacian and the diagonal elements of $\mathbf{\Lambda} = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{n-1})$ are the eigenvalues. With this, the Graph Fourier Transform (GFT) of a graph signal $\mathbf{x}$ is defined as $\tilde{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$, where $\tilde{\mathbf{x}}$ is the graph frequencies of $\mathbf{x}$. Correspondingly, the Inverse Graph Fourier Transform can be written as $\mathbf{x} = \mathbf{U}\tilde{\mathbf{x}}$.

With GFT and the Convolution Theorem, the graph convolution of a graph signal $\mathbf{x}$ and a filter $\mathbf{g}$ can be defined as $\mathbf{g} *_G \mathbf{x} = \mathbf{U}(\mathbf{U}^T\mathbf{g} \odot \mathbf{U}^T\mathbf{x})$. To simplify this, let $\mathbf{g}_\theta = \mathrm{diag}(\mathbf{U}^T g)$, the graph convolution can be written as:

$$\mathbf{g} *_G \mathbf{x} = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x}, \tag{9}$$

which is the general form of most spectral graph convolutions. The key of spectral graph convolutions is to parameterize and learn the filter $\mathbf{g}_\theta$.

Spectral Convolutional Neural Network (Spectral CNN) (Bruna et al., 2013) sets graph filter as a learnable diagonal matrix $\mathbf{W}$. The convolution operation can be written as $\mathbf{y} = \mathbf{U}\mathbf{W}\mathbf{U}^T\mathbf{x}$. In practice, multi-channel signals and activation functions are common, and the graph convolution can be written as

$$\mathbf{Y}_{:,j} = \sigma\left(\mathbf{U}\sum_{i=1}^{c_{in}}\mathbf{W}_{i,j}\mathbf{U}^T\mathbf{X}_{:,i}\right), \quad j = 1, 2, \ldots, c_{out}, \tag{10}$$

where c_{in} is the number of input channel, c_{out} is the number of output channel, $\mathbf{X}$ is a $n \times c_{in}$ matrix representing the input signal, $\mathbf{Y}$ is a $n \times c_{out}$ matrix denoting the output signal, $\mathbf{W}_{i,j}$ is a parameterized diagonal matrix, and $\sigma(\cdot)$ is the activation function. For mathematical convenience we sometimes use single-channel versions of graph convolutions omitting activation functions, and the multi-channel versions are similar to Eq. (10).

Spectral CNN has several limitations. Firstly, the filters are basis-dependent, which means that they cannot be generalized across graphs. Secondly, the algorithm requires eigen decomposition, which is computationally expensive. Thirdly, it has no guarantee of spatial localization of filters. To make filters spatially localized, Henaff et al. (2015)

propose to use a smooth spectral transfer function $\Theta(\mathbf{\Lambda})$ to parameterize the filter, and the convolution operation can be written as:

$$\mathbf{y} = \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x}. \tag{11}$$

Chebyshev Spectral Convolutional Neural Network (ChebNet) (Defferrard et al., 2016) extends this idea by using truncated Chebyshev polynomials to approximate the spectral transfer function. The Chebyshev polynomial is defined as $T_0(x) = 1$, $T_1(x) = x$, $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, and the spectral transfer function $F(\mathbf{\Lambda})$ is approximated to the order of $K - 1$ as

$$F(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{\Lambda}}), \tag{12}$$

where the model parameters θ_k, $k \in \{0, 1, \dots, K-1\}$ are the Chebyshev coefficients, and $\tilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_{max} - \mathbf{I}$ is a diagonal matrix of scaled eigenvalues. Thus, the graph convolution can be written as:

$$\mathbf{g} *_G \mathbf{x} = \mathbf{U}F(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x} = \mathbf{U}\sum_{k=0}^{K-1}\theta_k T_k(\tilde{\mathbf{\Lambda}})\mathbf{U}^T\mathbf{x} = \sum_{k=0}^{K-1}\theta_k T_k(\tilde{\mathbf{L}})\mathbf{x}, \tag{13}$$

where $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I}$.

Graph Convolutional Network (GCN) (Kipf & Welling, 2016a) is proposed as the localized first-order approximation of ChebNet. Assuming $K = 2$ and $\lambda_{max} = 2$, Eq. (13) can be simplified as:

$$\mathbf{g} *_G \mathbf{x} = \theta_0\mathbf{x} + \theta_1(\mathbf{L} - \mathbf{I})\mathbf{x} = \theta_0\mathbf{x} - \theta_1\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{x}. \tag{14}$$

To further constraint the number of parameters, we assume $\theta = \theta_0 = -\theta_1$, which gives a simpler form of graph convolution:

$$\mathbf{g} *_G \mathbf{x} = \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}. \tag{15}$$

As $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ now has the eigenvalues in the range of $[0, 2]$ and repeatedly multiplying this matrix can lead to numerical instabilities, GCN empirically proposes a renormalization trick to solve this problem by using $\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ instead, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}_{ii} = \sum_i \tilde{\mathbf{A}}_{ij}$.

Allowing multi-channel signals and adding activation functions, the more common formula in literature is:

$$\mathbf{Y} = \sigma((\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2})\mathbf{X}\mathbf{\Theta}), \tag{16}$$

where $\mathbf{X}$, $\mathbf{Y}$ have the same shape as in Eq. (10) and $\mathbf{\Theta}$ is a $c_{in} \times c_{out}$ matrix as model's parameters.

Apart from the aforementioned methods, other spectral graph convolutions have been proposed. Levie et al. (2018) propose CayleyNets that utilize Cayley Polynomials to equip the filters with the ability to detect narrow frequency bands. Liao et al. (2019) propose LanczosNets that employ the Lanczos algorithm to construct a low-rank approximation of graph Laplacian to improve the computation efficiency of graph convolutions. The proposed model is able to efficiently utilize the multi-scale information in the graph data. Instead of using Graph Fourier Transform, Xu, Shen et al. (2019) propose a Graph Wavelet Neural Network (GWNN) that uses graph wavelet transform to avoid matrix eigendecomposition. Moreover, graph wavelets are sparse and localized, which provides good interpretations for the convolution operation. Zhu and Koniusz (2021) derive a Simple Spectral Graph Convolution (S^2GC) from a modified Markov Diffusion Kernel, which achieves a trade-off between low-pass and high-pass filter bands.

3.2. Spatial graph convolutions

Inspired by the convolution on Euclidean data (*e.g.* images and texts), which applies data transformation on a small region, spatial graph convolutions compute the central node's feature via transforming and aggregating its neighbors' features. In this way, the graph structure is naturally embedded in the computation graph of node features. Moreover, the idea of sending one node's feature to another node is similar to the message passing used in recurrent graph neural networks. In the

following, we will introduce several seminal spatial graph convolutions as well as some recently proposed promising methods.

Spatial graph convolutions generally follow a three-step paradigm: message generation, feature aggregation and feature update. This can be mathematically written as:

$$\mathbf{y}_i = \text{UPDATE}\left(\mathbf{x}_i, \text{AGGREGATE}\left(\{\text{MESSAGE}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ij}\right), \, j \in \mathcal{N}(i)\}\right)\right), \tag{17}$$

where $\mathbf{x}_i$ and $\mathbf{y}_i$ is the input and output feature vector of node i, $\mathbf{e}_{ij}$ is the feature vector of the edge (or more generally, the relationship) between node i and its neighbor node j, and $\mathcal{N}(i)$ denote the neighbor of node i, which could be more generally defined.

In the previous subsection, we show the spectral interpretation of GCN (Kipf & Welling, 2016a). The model also has its spatial interpretation, which can be mathematically written as:

$$\mathbf{y}_i = \mathbf{\Theta}^T \sum_{j \in \mathcal{N}(i) \cup i} \frac{1}{\sqrt{\hat{d}_i \hat{d}_j}} \mathbf{x}_j, \tag{18}$$

where $\hat{d}_i$ and $\hat{d}_j$ is the ith and jth row sums of $\hat{\mathbf{A}}$ in Eq. (16). For each node, the model takes a weighted sum of its neighbors' features as well as its own features and applies a linear transformation to obtain the result. In practice, multiple GCN layers are often stacked together with non-linear functions after convolution to encode complex and hierarchical features. Nonetheless, Wu, Souza et al. (2019) show that the model still achieves competitive results without non-linearity.

Although GCN as well as other spectral graph convolutions achieve competitive results on a number of benchmarks, these methods assume the presence of all nodes in the graph and fall in the category of transductive learning. Hamilton et al. (2017) propose GraphSAGE that performs graph convolutions in inductive settings, when there are new nodes during inference (*e.g.* newcomers in the social network). For each node, the model samples its K-hop neighbors and uses K graph convolutions to aggregate their features hierarchically. Furthermore, the use of sampling also reduces the computation when a node has too many neighbors.

The attention mechanism has been successfully used in natural language processing (Vaswani et al., 2017), computer vision (Liu et al., 2021) and multi-modal tasks (Chen, Guhur, Schmid and Laptev, 2021; He et al., 2021; Yu, Yu, Cui, Tao, & Tian, 2019; Zhao et al., 2022). Graph Attention Networks (GAT) (Veličković et al., 2017) introduces the idea of attention to graphs. The attention mechanism uses an adaptive, feature-dependent weight (*i.e.* attention coefficient) to aggregate a set of features, which can be mathematically written as:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{\Theta}\mathbf{x}_i \| \mathbf{\Theta}\mathbf{x}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{\Theta}\mathbf{x}_i \| \mathbf{\Theta}\mathbf{x}_j]\right)\right)}, \tag{19}$$

where $\alpha_{i,j}$ is the attention coefficient, $\mathbf{a}$ and $\mathbf{\Theta}$ are model parameters, and $[\cdot \| \cdot]$ means concatenation. After the αs are obtained, the new features are computed as a weighted sum of input node features, which is:

$$\mathbf{y}_i = \alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j. \tag{20}$$

Xu, Hu et al. (2018) explore the representational limitations of graph neural networks. What they discover is that message passing networks like GCN (Kipf & Welling, 2016a) and GraphSAGE (Hamilton et al., 2017) are incapable of distinguishing certain graph structures. To improve the representational power of graph neural networks, they propose the Graph Isomorphism Network (GIN) that gives an adjustable weight to the central node feature, which can be mathematically written as:

$$\mathbf{y}_i = \text{MLP}\left((1 + \epsilon)\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j\right), \tag{21}$$

where ϵ is a learnable parameter.

More recently, efforts have been made to improve the representational power of graph neural networks. For example, Hu, Zhu et al. (2020) propose GraphAIR that explicitly models the neighborhood interaction to better capture complex non-linear features. Specifically, they use the Hadamard product between pairs of nodes in the neighborhood to model the quadratic terms of neighborhood interaction. Balcilar et al. (2021) propose GNNML3 that breaks the limits of the first-order Weisfeiler–Lehman test (1-WL) and reaches the third-order WL test (3-WL) experimentally. They also show that the Hadamard product is required for the model to have more representational power than the first-order Weisfeiler–Lehman test. Other elements in spatial graph convolutions are widely studied. For example, Corso et al. (2020) explore the aggregation operation in GNN and proposes Principal Neighborhood Aggregation (PNA) that uses multiple aggregators with degree-scalers. Tailor et al. (2022) explore the anisotropism and isotropism in the message passing process of graph neural networks, and proposes Efficient Graph Convolution (EGC) that achieves promising results with reduced memory consumption due to isotropism. In order to increase the size of the neighborhood of a node, Gasteiger et al. (2018) propose personalized propagation of neural predictions (PPNP) and its approximation using power iteration (APPNP). To increase the depth of graph neural networks, Chen, Wei et al. (2020) propose GCNII that uses initial residual and identity mapping to mitigate the over-smoothing problem. Brody et al. (2021) propose GATv2 that uses dynamic attention and improves the expressive power of GAT (Veličković et al., 2017).

3.3. Summary

This section introduces graph convolutions. We provide the summary as follows:

- **Techniques.** Graph convolutions mainly fall into two types, *i.e.* spectral graph convolutions and spatial graph convolutions. Spectral graph convolutions have solid mathematical foundations of Graph Signal Processing and therefore their operations have theoretical interpretations. Spatial graph convolutions are inspired by Recurrent Graph Neural Networks and their computation is simple and straightforward, as their computation graph is derived from the local graph structure. Generally, spatial graph convolutions are more common in applications.
- **Challenges and Limitations.** Despite the great success of graph convolutions, their performance is unsatisfactory in more complicated applications. On the one hand, the performance of graph convolutions relies heavily on the construction of the graph. Different constructions of the graph might result in different performances of graph convolutions. On the other hand, graph convolutions are prone to over-smoothing when constructing very deep neural networks.
- **Future Works.** In the future, we expect that more powerful graph convolutions will be developed to mitigate the problem of over-smoothing and we also hope that techniques and methodologies in Graph Structure Learning (GSL) can help learn more meaningful graph structure to benefit the performance of graph convolutions.

4. Graph kernel neural networks

Graph kernels (GKs) are historically the most widely used technique on graph analyzing and representation tasks (Gärtner et al., 2003; Krishnagopal & Ruiz, 2023; Shawe-Taylor, Cristianini, et al., 2004; Zhou, Cui et al., 2020). However, traditional graph kernels rely on hand-crafted patterns or domain knowledge on specific tasks (Kriege, Johansson, & Morris, 2020; Shervashidze et al., 2009). Over the years,

an amount of research has been conducted on graph kernel neural networks (GKNNs), which has yielded promising results. Researchers have explored various aspects of GKNNs, including their theoretical foundations, algorithmic design, and practical applications. These efforts have led to the development of a wide range of GKNN-based models and methods that can be used for graph analysis and representation tasks, such as node classification (Fang, Xu, Song, Long and Zhang, 2022; Ju et al., 2023; Yang et al., 2023), link prediction (Chen, Jacob and Mairal, 2020; Long et al., 2019; Wu, Shi, He, & Jin, 2023), and graph clustering (Ju et al., 2022; Krishnagopal & Ruiz, 2023; Long, Jin, Wu and Song, 2021).

The success of GKNNs can be attributed to their ability to leverage the strengths of both graph kernels and neural networks (Ju et al., 2022; Long, Jin et al., 2021; Wu et al., 2023). By using kernel functions to measure similarity between graphs, GKNNs can capture the structural properties of graphs, while the use of neural networks enables them to learn more complex and abstract representations of graphs (Chen, O'Bray and Borgwardt, 2022; Zang, Zhao, & Tang, 2023). This combination of techniques allows GKNNs to achieve state-of-the-art performance on a wide range of graph-related tasks (Krishnagopal & Ruiz, 2023; Wang, Zhao, Shah and Derr, 2022).

In this section, we begin with introducing the most representative traditional graph kernels. Then we summarize the basic framework for combining GNNs and graph kernels. Finally, we categorize the popular graph kernel Neural networks into several categories and compare their differences.

4.1. Graph kernels

Graph kernels generally evaluate pairwise similarity between nodes or graphs by decomposing them into basic structural units. Random walks (Kang, Tong, & Sun, 2012), subtrees (Shervashidze, Schweitzer, Van Leeuwen, Mehlhorn, & Borgwardt, 2011b), shortest paths (Borgwardt & Kriegel, 2005) and graphlets (Shervashidze et al., 2009) are representative categories.

Given two graphs $G_1 = (V_1, E_1, X_1)$ and $G_2 = (V_2, E_2, X_2)$, a graph kernel function $K(G_1, G_2)$ measures the similarity between G_1 and G_2 through the following formula:

$$K(G_1, G_2) = \sum_{u_1 \in V_1} \sum_{u_2 \in V_2} \kappa_{base} \left(l_{G_1}(u_1), l_{G_2}(u_2) \right), \tag{22}$$

where $l_G(u)$ denotes a set of local substructures centered at node u in graph G, and κ_{base} is a base kernel measuring the similarity between the two sets of substructures. For simplicity, we may rewrite Eq. (22) as:

$$K(G_1, G_2) = \sum_{u_1 \in V_1} \sum_{u_2 \in V_2} \kappa_{base}(u_1, u_2), \tag{23}$$

the uppercase letter $K(G_1, G_2)$ is denoted as graph kernels, $\kappa(u_1, u_2)$ is denoted as node kernels, and lowercase $k(x, y)$ is denoted as general kernel functions.

The kernel mapping of a kernel ψ maps a data point into its corresponding Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. Specifically, given a kernel $k_*(\cdot, \cdot)$, its kernel mapping ψ_* can be formalized as,

$$\forall x_1, x_2, k_*(x_1, x_2) = \langle \psi_*(x_1), \psi_*(x_2) \rangle_{\mathcal{H}_*}, \tag{24}$$

where $\mathcal{H}_*$ is the RKHS of $k_*(\cdot, \cdot)$.

We introduce several representative and popular graph kernels below.

Walk and Path Kernels. A l-walk kernel $K_{walk}^{(l)}$ compares all length l walks starting from each node in two graphs G_1, G_2,

$$\kappa_{walk}^{(l)}(u_1, u_2) = \sum_{w_1 \in \mathcal{W}^l(G_1, u_1)} \sum_{w_2 \in \mathcal{W}^l(G_2, u_2)} \delta(X_1(w_1), X_2(w_2)),$$

$$K_{walk}^{(l)}(G_1, G_2) = \sum_{u_1 \in V_1} \sum_{u_2 \in V_2} \kappa_{walk}^{(l)}(u_1, u_2). \tag{25}$$

Substituting $\mathcal{W}$ with $\mathcal{P}$ is able to get the l-path kernel.

Subtree Kernels. The WL subtree kernel is the most popular one in subtree kernels. It is a finite-depth kernel variant of the 1-WL test. The WL subtree kernel with depth l, $K_{WL}^{(l)}$ compares all subtrees with depth $\leq l$ rooted at each node.

$$
\kappa_{subtree}^{(i)}(u_1, u_2) = \sum_{t_1 \in \mathcal{T}^i(G_1, u_2)} \sum_{t_2 \in \mathcal{T}^i(G_2, u_2)} \delta(t_1, t_2),
$$

$$
K_{subtree}^{(i)}(G_1, G_2) = \sum_{u_1 \in V_1} \sum_{u_2 \in V_2} \kappa_{subtree}^{(i)}(u_1, u_2), \tag{26}
$$

$$
K_{WL}^{(l)}(G_1, G_2) = \sum_{i=0}^{l} K_{subtree}^{(i)}(G_1, G_2),
$$

where $t \in \mathcal{T}^{(i)}(G, u)$ denotes a subtree of depth i rooted at u in G.

4.2. General framework of GKNNs

In this section, we summarize the general framework of GKNNs. For the first step, a kernel that measures similarities of heterogeneous features from heterogeneous nodes and edges $(u_1, e_{\cdot, u_2})$ and $(u_2, e_{\cdot, u_2})$ should be defined. Take the inner product of neighbor tensors as an example, its neighbor kernel is defined as follows,

$$
\kappa((u_1, e_{\cdot, u_1}), (u_2, e_{\cdot, u_2})) = \langle f(u_1), f(u_2) \rangle \cdot \langle f(e_{\cdot, u_1}), f(e_{\cdot, u_2}) \rangle.
$$

Based on the neighbor kernel, a kernel with two l-hop neighborhoods for central node u_1 and u_2 can be defined as $K^{(l)}(u_1, u_2) =$

$$
\begin{cases}
\langle f(u_1), f(u_2) \rangle & l = 0 \\
\langle f(u_1), f(u_2) \rangle \cdot \displaystyle\sum_{v_1 \in N(u_1)} \sum_{v_2 \in N(u_2)} K^{(l-1)}(v_1, v_2) \cdot \langle f(e_{\cdot, v_1}), f(e_{\cdot, v_2}) \rangle & l > 0
\end{cases}, \tag{27}
$$

By regarding the lower-hop kernel $\kappa^{(l-1)}(u_1, u_2)$, as the inner product of the $(l-1)$th hidden representations of u_1 and u_2. Furthermore, by recursively applying the neighborhood kernel, the l-hop graph kernel can be derived as

$$
K^l(G_1, G_2) = \sum_{\boldsymbol{w}_1 \in \mathcal{W}^l(G_1)} \sum_{\boldsymbol{w}_2 \in \mathcal{W}^l(G_2)}
$$
$$
\times \left(\prod_{i=0}^{l-1} \langle f(\boldsymbol{w}_1^{(i)}), f(\boldsymbol{w}_2^{(i)}) \rangle \times \prod_{i=0}^{l-2} \langle f(e_{\boldsymbol{w}_1^{(i)}, \boldsymbol{w}_1^{(i+1)}}), f(e_{\boldsymbol{w}_2^{(i)}, \boldsymbol{w}_2^{(i+1)}}) \rangle \right), \tag{28}
$$

where $\mathcal{W}^l(G)$ denotes the set of all walk sequences with length l in graph G, and $\boldsymbol{w}_1^{(i)}$ denotes the ith node in sequence $\boldsymbol{w}_1$.

As shown in Eq. (24), kernel methods implicitly perform projections from original data spaces to their RKHS $\mathcal{H}$. Hence, as GNNs also project nodes or graphs into vector spaces, connections have been established between GKs and GNNs through the kernel mappings. And several works conducted research on the connections (Lei, Jin, Barzilay, & Jaakkola, 2017; Williams & Seeger, 2001), and found some foundation conclusions. Take the basic rule introduced in Lei et al. (2017) as an example, the proposed graph kernel in Eq. (22) can be derived as the general formulas,

$$
h^{(0)}(v) = \boldsymbol{W}_{t_V(v)}^{(0)} f(v),
$$
$$
h^{(l)}(v) = \boldsymbol{W}_{t_V(v)}^{(l)} f(v) \odot \sum_{u \in N(v)} (\boldsymbol{U}_{t_V(v)}^{(l)} h^{(l-1)}(u) \tag{29}
$$
$$
\odot \boldsymbol{U}_{t_E(e_{u,v})}^{(l)} f(e_{u,v})), \qquad 1 < l \leq L,
$$

where $\odot$ is the element-wise product and $h^{(l)}(v)$ is the cell state vector of node v. The parameter matrices $\boldsymbol{W}_{t_V(v)}^{(l)}$, $\boldsymbol{U}_{t_V(v)}^{(l)}$ and $\boldsymbol{U}_{t_E(e_{u,v})}^{(l)}$ are learnable parameters related to types of nodes and edges.

Then mean embeddings of all nodes are usually used to represent the graph-level embedding, let $|G_i|$ denote the number of nodes in the ith graph, then the graph-level embeddings are generated as,

$$
\Phi(G_i) = \sum_{u \in G_i} \frac{1}{|G_i|} h^{(L)}(v). \tag{30}
$$

For semi-supervised multiclass classification, the cross entropy is used as the objective function over all training examples (Cao et al., 2015, 2016),

$$
\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}, \tag{31}
$$

where $\mathcal{Y}_L$ is the set of node indices that have labels in node classification tasks, or the set of graph indices in graph classification tasks. Z_{lf} denotes the prediction of labels, which are outputs of a linear layer with an activation function, inputting $h^{(l)}(v)$ in the node classification task, and $\Phi(G_i)$ in the graph classification task.

4.3. Popular variants of GKNNs

We summarize the popular variants of GKNNs and compare their differences in Table 3. Specifically, we conclude their basic graph kernels, whether designed for heterogeneous graphs, experimental datasets, etc. As the originally designed graph-kernel based GNNs have high complexity, they usually by acceleration strategies, such as sampling strategies, simplification and approximation, etc. In this section, We select four typical and popular GKNNs to introduce their well-designed graph kernels and corresponding GNN frameworks.

k-dimensional Graph Neural Networks (*k-GNN*) (Morris et al., 2019) is the pioneer of GKNNs, it incorporates the WL-subtree graph kernel and graph neural networks. For better scalability, the paper considers a set-based version of the k-WL. Let $h_k^{(l)}(s)$ and $h_{k,L}^{(l)}(s)$ denote the *global* and *local* hidden representation for node s in the lth layer respectively. k-GNN defined the end-to-end hierarchical trainable framework as,

$$
h_k^{(0)}(s) = \sigma\left(\left[h^{iso}(s), \sum_{u \in s} h^{(T_k-1)}(u) \right] \cdot \boldsymbol{W}_{k-1}^{(0)} \right),
$$
$$
h_k^{(l)}(s) = \sigma\left(h_k^{(l-1)} \cdot \boldsymbol{W}_1^{(l)} + \sum_{u \in N_L(s) \cup N_G(s)} h_k^{(l-1)}(u) \cdot \boldsymbol{W}_2^{(l)} \right), \qquad 1 < l \leq L,
$$
$$
h_{k,L}^{(l)}(s) = \sigma\left(h_{k,L}^{(l-1)}(s) \cdot \boldsymbol{W}_1^{(l)} + \sum_{u \in N_L(s)} h_{k,L}^{(l-1)}(u) \cdot \boldsymbol{W}_2^{(l)} \right), \qquad 1 < l \leq L.
$$
$$\tag{32}$$

where $h^{iso}(s)$ is the one-hot encoding of the isomorphism type of $G[s]$, $N_L(s)$ is the local neighborhood, $N_G(s)$ is the global neighborhood, $h_{k,L}^{(l)}(s)$ is designed for better scalability and running speed, and $h_k^{(l)}(s)$ has better performance due to its larger neighbor sets.

Graph Convolutional Kernel Networks (GCKN) (Chen, Jacob et al., 2020). GCKN is a representative random walk and path-based GKNN. The Gaussian kernel k can be written as,

$$
k(z_1, z_2) = e^{-\frac{\alpha_1}{2} \|z_1 - z_2\|^2} = e^{\alpha(z_1^T z_2 - k - 1)} = \sigma(z_1^T z_2), \tag{33}
$$

then the GNN architecture can be written as

$$
h^{(l)}(u) = \sum_{z \in \mathbb{Z}} K(z_1, z_2) \cdot \sigma\left(Z^T h^{(l-1)}(p) \right)
$$
$$
= \sigma\left(ZZ^T \right)^{-\frac{1}{2}} \cdot \sum_{p \in \mathcal{P}_k(G, u)} \sigma\left(Z^T h^{(l-1)}(p) \right), \tag{34}
$$

where Z is the matrix of prototype path attributes.

Furthermore, the paper analyzes the relationship between GCKN and the WL-subtree based k-GNN. Theorem 1 in paper (Chen, Jacob et al., 2020) shows that WL-subtree based GKNNs can be seen as a special case in GCKN.

Graph Neural Tangent Kernel (GNTK) (Du et al., 2019). Different from the above two works, GNTK proposed a new class of graph kernels. GNTK is a general recipe which translates a GNN architecture to its corresponding GNTK.

Table 3
Summary of popular GKNNs.

Method	Type	Related GK	Adaptive	Datasets
k-GNN (Morris et al., 2019)	Isomorphic	WL subtree	✓	Biochemical network
RetGK (Zhang, Wang, Xiang, Huang and Nehorai, 2018)	Isomorphic	Random walk	✓	Biochemical network, Social network
GNTK (Du et al., 2019)	Isomorphic	Neural tangent kernel		Biochemical network, Social network
DDGK (Al-Rfou, Perozzi, & Zelle, 2019)	Isomorphic	Random walk		Biochemical network
GCKN (Chen, Jacob et al., 2020)	Isomorphic	Random walk	✓	Biochemical network, Social network
GSKN (Long, Jin et al., 2021)	Isomorphic	Random walk, Anonymous walk	✓	Biochemical network, Social network
KerGNNs (Feng, You, Wang, & Tassiulas, 2022)	Isomorphic	Random walk	✓	Biochemical network, Academic network
TurboMGNN (Wu et al., 2023)	Isomorphic	Random walk	✓	Social network
GCN-LASE (Li, Zhang and Song, 2019)	Heterogeneous	Random walk		Social network Academic network
HGK-GNN (Long, Xu, Fang and Song, 2021)	Heterogeneous	Random walk	✓	Social network Academic network

The neighborhood aggregation operation in GNTK is defined as,

$$
\left[\sum_{(0)}^{(l)}(G,G')\right]_{uu'} = c_u c_{u'} \sum_{u \in N(v)\cup\{u\}} \sum_{v' \in N(u')\cup\{u'\}} \left[\sum_{R}^{(l-1)}(G,G')\right]_{vv'},
$$
$$
\left[\Theta_{(0)}^{(l)}(G,G')\right]_{uu'} = c_u c_{u'} \sum_{u \in N(v)\cup\{u\}} \sum_{v' \in N(u')\cup\{u'\}} \left[\Theta_R^{(l-1)}(G,G')\right]_{vv'}, \tag{35}
$$

where $\sum_R^{(0)}(G,G')$ and $\Theta_R^{(0)}(G,G')$ are defined as $\sum^{(0)}(G,G')$. Then GNTK performed R transformations to capture high-order structural information.

$$
\left[A_{(r)}^{(l)}(G,G')\right]_{uu'} = \begin{pmatrix} \left[\sum_{(r-1)}^{(l)}(G,G)\right]_{uu'} & , & \left[\sum_{(r-1)}^{(l)}(G,G')\right]_{uu'} \\ \left[\sum_{(r-1)}^{(l)}(G',G)\right]_{uu'} & , & \left[\sum_{(r-1)}^{(l)}(G',G')\right]_{uu'} \end{pmatrix}. \tag{36}
$$

$$
\left[\sum_{(r)}^{(l)}(G,G')\right]_{uu'} = \mathbb{E}_{(a,b)\sim\mathcal{N}(0,[A_{(r)}^{(l)}(G,G')]_{uu'})}[\sigma(a)\cdot\sigma(b)],
$$
$$
\left[\sum_{(r)}^{(l)}(G,G')\right]_{uu'} = \mathbb{E}_{(a,b)\sim\mathcal{N}(0,[A_{(r)}^{(l)}(G,G')]_{uu'})}[\sigma(a)\cdot\sigma(b)], \tag{37}
$$

then the r order can be calculated as,

$$
\left[\Theta_{(r)}^{(l)}(G,G')\right]_{uu'} = \left[\Theta_{(r-1)}^{(l)}(G,G')\right]_{uu'} \left[\sum(r)^{(l)}(G,G')\right]_{uu'} + \left[\sum(r)^{(l)}(G,G')\right]_{uu'}. \tag{38}
$$

Finally, GNTK calculates the final output as,

$$
\Theta(G,G') = \sum_{u \in V, u' \in V'} \left[\sum_{l=0}^{L} \Theta_{(R)}^{(l)}(G,G')\right]_{u,u'}. \tag{39}
$$

Heterogeneous Graph Kernel based Graph Neural Network (HGK-GNN) (Long, Xu et al., 2021). HGK-GNN first proposed GKNN for heterogeneous graphs. It adopted $\langle f(u_1), f(u_2)\rangle_M$ as graph kernel based on the Mahalanobis Distance to build connections among heterogeneous nodes and edges,

$$
\langle f(u_1), f(u_2)\rangle_{M_1} = f(u_1)^T \mathbf{M}_1 f(u_2),
$$

$$
\langle f(e_{\cdot,u_1}), f(e_{\cdot,u_2})\rangle_{M_2} = f(e_{\cdot,u_1})^T \mathbf{M}_2 f(e_{\cdot,u_2}).
$$

Following the route introduced in Lei et al. (2017), the corresponding neural network architecture of heterogeneous graph kernel can be

derived as

$$
h^{(0)}(v) = \mathbf{W}_{t_V(v)}^{(0)} f(v),
$$
$$
h^{(l)}(v) = \mathbf{W}_{t_V(v)}^{(l)} f(v) \odot \sum_{u \in N(v)} (\mathbf{U}_{t_V(v)}^{(l)} h^{(l-1)}(u)
$$
$$
\odot \mathbf{U}_{t_E(e_{u,v})}^{(l)} f(e_{u,v})), \qquad 1 < l \le L, \tag{40}
$$

where $h^{(l)}(v)$ is the cell state vector of node v, and $\mathbf{W}_{t_V(v)}^{(l)}, \mathbf{U}_{t_V(v)}^{(l)} \mathbf{U}_{t_E(e_{u,v})}^{(l)}$ are learnable parameters.

4.4. Summary

This section introduces graph kernel neural networks. We provide the summary as follows:

- **Techniques.** Graph kernel neural networks (GKNNs) are a recent popular research area that combines the advantages of graph kernels and GNNs to learn more effective graph representations. Researchers have studied GKNNs in various aspects, such as theoretical foundations, algorithmic design, and practical applications. As a result, a wide range of GKNN-based models and methods have been developed for graph analysis and representation tasks, including node classification, link prediction, and graph clustering.
- **Challenges and Limitations.** Although GKNNs have shown great potential in graph-related tasks, they also have several limitations that need to be addressed. Scalability is a significant challenge, particularly when dealing with large-scale graphs and networks. As the size of the graph increases, the computational cost of GKNNs grows exponentially, which can limit their ability to handle large and complex real-world applications.
- **Future Works.** For future works, we expect the GKNNs can integrate more domain-specific knowledge into the designed kernels. Domain-specific knowledge has been shown to significantly improve the performance of many applications, such as drug discovery (Zeng et al., 2023), knowledge graph-based information retrieval systems, and molecular analysis (Feinberg et al., 2018; Wang et al., 2019; Zang et al., 2023). Incorporating domain-specific knowledge into GKNNs can enhance their ability to handle complex and diverse data structures, leading to more accurate and interpretable models.

Table 4
Summary of graph pooling methods.

Method	Type	TopK-based	Cluster-based	Attention mechanism
Mean/Sum/Max Pooling	Global			
GGS-NN (Li et al., 2016)	Global			✓
SortPool (Zhang, Cui et al., 2018)	Global			
SSRead (Lee et al., 2021)	Global			
gPool (Gao & Ji, 2019b)	Hierarchical	✓		
SAGPool (Lee, Lee, & Kang, 2019)	Hierarchical	✓		✓
HGP-SL (Zhang et al., 2019)	Hierarchical	✓		✓
TAPool (Gao, Liu, & Ji, 2021)	Hierarchical	✓		
DiffPool (Ying et al., 2018)	Hierarchical		✓	
MinCutPool (Bianchi, Grattarola, & Alippi, 2020)	Hierarchical		✓	
SEP (Wu, Chen, Xu, & Li, 2022)	Hierarchical		✓	
ASAP (Ranjan, Sanyal, & Talukdar, 2020)	Hierarchical	✓	✓	✓
MuchPool (Du, Wang, Miao, & Zhang, 2021)	Hierarchical	✓	✓	

5. Graph pooling

When it comes to graph-level tasks, such as graph classification and graph regression, graph pooling is an essential component for generating the whole graph representation from the learned node embeddings. To ensure isomorphic graphs have the same representation, the graph pooling operations should be invariant to the permutations of nodes. In this section, we give a systematic review of existing graph pooling algorithms and generally classify them into two categories: global pooling algorithms and hierarchical pooling algorithms. The global pooling algorithms aggregate the node embeddings to the final graph representation directly, while the hierarchical pooling algorithms reduce the graph size and generate the immediate representations gradually to capture the hierarchical structure and characteristics of the input graph. A summary is provided in Table 4.

5.1. Global pooling

Global pooling operations generate a holistic graph representation from the learned node embeddings in one step, which are referred to as readout functions in some literature (Corso et al., 2020; Xu, Hu et al., 2018) as well. Several simple permutation-invariant operations, such as mean, sum, and max, are widely employed on the node embeddings to output the graph-level representation. To enhance the adaptiveness of global pooling operators, GGS-NN (Li, Zemel, Brockschmidt, & Tarlow, 2016) introduces a soft attention mechanism to evaluate the importance of nodes for a particular task and then takes a weighted sum of the node embeddings. SortPool (Zhang, Cui, Neumann and Chen, 2018) exploits Weisfeiler–Lehman methods (Weisfeiler & Leman, 1968) to sort nodes based on their structural positions in the graph topology, and produces the graph representation from the sorted node embeddings by traditional convolutional neural networks. Recently, SSRead (Lee, Kim, Lee, Park, & Yu, 2021) proposes a structural-semantic pooling method, which first aligns nodes and learnable structural prototypes semantically, and then aggregates node representations in groups based on matching structural prototypes.

5.2. Hierarchical pooling

Different from global pooling methods, hierarchical pooling methods coarsen the graph gradually, to preserve the structural information of the graph better. To adaptively coarse the graph and learn optimal hierarchical representations according to a particular task, many learnable hierarchical pooling operators have been proposed in the past few years, which can be integrated with multifarious graph convolution layers. There are two common approaches to coarsening the graph, one is selecting important nodes and dropping the others by TopK selection, and the other one is merging nodes and generating the coarsened graph by clustering methods. We call the former *TopK-based pooling*, and

the latter *cluster-based pooling* in this survey. In addition, some works combine these two types of pooling methods, which will be reviewed in the *hybrid pooling* section.

5.2.1. TopK-based pooling

Typically, TopK-based pooling methods first learn a scoring function to evaluate the importance of nodes of the original graph. Based on importance score $\mathbf{Z} \in \mathbb{R}^{|V| \times 1}$ generated, they select the top K nodes out of all nodes,

$$idx = \text{TOP}_k(\mathbf{Z}), \tag{41}$$

where idx denotes the index of the top K nodes. Based on these selected nodes, most methods directly employ the induced subgraph as the pooled graph,

$$\mathbf{A}_{\text{pool}} = \mathbf{A}_{idx,idx}, \tag{42}$$

where $A_{idx,idx}$ denotes the adjacency matrix indexed by the selected rows and columns. Moreover, to make the scoring function learnable, they further use score Z as a gate for the selected node features,

$$\mathbf{X}_{\text{pool}} = \mathbf{X}_{idx,:} \odot \mathbf{Z}_{idx}, \tag{43}$$

where $X_{idx,:}$ denotes the feature matrix indexed by the selected nodes, and $\odot$ denotes the broadcasted element-wise product. With the help of the gate mechanism, the scoring function can be trained by back-propagation, to adaptively evaluate the importance of nodes according to a certain task. Several representative TopK-based pooling methods are reviewed in detail in the following.

gPool (Gao & Ji, 2019b). gPool is one of the first works to select the most important node subset from the original graph to construct the coarsened graph by Top K operation. The key idea of gPool is to evaluate the importance of all nodes by learning a projection vector $\mathbf{p}$, which projects node features to a scalar score,

$$\mathbf{Z}_j = \mathbf{X}_{j,:} \mathbf{p} / \|\mathbf{p}\|, \tag{44}$$

and then select nodes with K-highest scores to form the pooled graph.

Self-Attention Graph Pooling (SAGPool) (Lee et al., 2019). Unlike gPool (Gao & Ji, 2019b), which only uses node features to generate projection scores, SAGPool captures both graph topology and node features to obtain self-attention scores by graph convolution. The various formulas of graph convolution can be employed to compute the self-attention score $\mathbf{Z}$,

$$\mathbf{Z} = \sigma(\text{GNN}(\mathbf{X}, \mathbf{A})), \tag{45}$$

where σ denotes the activation function, and GNN denotes various graph convolutional layers or stacks of them, whose output dimension is one.

Hierarchical Graph Pooling with Structure Learning (HGP-SL) (Zhang, Bu et al., 2019) . HGP-SL evaluates the importance score of a node according to the information it contains given its neighbors. It supposes

that a node which can be easily represented by its neighborhood carries relatively little information. Specifically, the importance score can be defined by the Manhattan distance between the original node representation and the reconstructed one aggregated from its neighbors' representation,

$$\mathbf{Z} = \left\| \left(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \right) \mathbf{X} \right\|_1, \tag{46}$$

where $\mathbf{I}$ denotes the identity matrix, $\mathbf{D}$ denotes the diagonal degree matrix of $\mathbf{A}$, and $\| \cdot \|_1$ means ℓ_1 norm. Furthermore, to reduce the loss of topology information, HGP-SL leverages structure learning to learn a refined graph topology for the reserved nodes. Specifically, it utilizes the attention mechanism to compute the similarity of two nodes as the weight of the reconstructed edge,

$$\widetilde{\boldsymbol{A}}_{ij}^{\text{pool}} = \text{sparsemax} \left(\sigma \left(\mathbf{w} \left[\boldsymbol{X}_{i,:}^{\text{pool}} \parallel \boldsymbol{X}_{j,:}^{\text{pool}} \right]^\top \right) + \lambda \cdot \boldsymbol{A}_{ij}^{\text{pool}} \right), \tag{47}$$

where $\widetilde{\boldsymbol{A}}^{\text{pool}}$ denotes the refined adjacency matrix of the pooled graph, $\text{sparsemax}(\cdot)$ truncates the values below a threshold to zeros, $\mathbf{w}$ denotes a learnable weight vector, and λ is a weight parameter between the original edges and the reconstructed edges. These reconstructed edges may capture the underlying relationship between nodes disconnected due to node dropping.

Topology-Aware Graph Pooling (TAPool) (Gao et al., 2021). TAPool takes both the local and global significance of a node into account. On the one hand, it utilizes the average similarity between a node and its neighbors to evaluate its local importance,

$$\hat{\mathbf{R}} = \left(\mathbf{X}\mathbf{X}^T \right) \odot \left(\mathbf{D}^{-1}\mathbf{A} \right), \mathbf{Z}^l = \text{softmax} \left(\frac{1}{n} \hat{\mathbf{R}} \mathbf{1}_n \right), \tag{48}$$

where $\hat{\mathbf{R}}$ denotes the localized similarity matrix, and $\mathbf{Z}^l$ denotes the local importance score. On the other hand, it measures the global importance of a node according to the significance of its one-hop neighborhood in the whole graph,

$$\hat{\mathbf{X}} = \mathbf{D}^{-1}\mathbf{A}\mathbf{X}, \mathbf{Z}^g = \text{softmax} \left(\hat{\mathbf{X}}\mathbf{p} \right), \tag{49}$$

where $\mathbf{p}$ is a learnable and globally shared projector vector, similar to the aforementioned gPool (Gao & Ji, 2019b). However, $\hat{\mathbf{X}}$ here further aggregates the features from the neighborhood, which enables the global importance score $\mathbf{Z}^g$ to capture more topology information such as salient subgraphs. Moreover, TAPool encourages connectivity in the coarsened graph with the help of a degree-based connectivity term, then obtaining the final importance score $\mathbf{Z} = \mathbf{Z}^l + \mathbf{Z}^g + \lambda \mathbf{D}/|V|$, where λ is a trade-off hyperparameter.

5.2.2. Cluster-based pooling

Pooling the graph by clustering and merging nodes is the main concept behind cluster-based pooling methods. Typically, they allocate nodes to a collection of clusters by learning a cluster assignment matrix $\mathbf{S} \in \mathbb{R}^{|V| \times K}$, where K is the number of the clusters. After that, they merge the nodes within each cluster to generate a new node in the pooled graph. The feature (embedding) matrix of the new nodes can be obtained by aggregating the features (embeddings) of nodes within the clusters, according to the cluster assignment matrix,

$$\mathbf{X}^{\text{pool}} = \mathbf{S}^T\mathbf{X}. \tag{50}$$

While the adjacency matrix of the pooled graph can be generated by calculating the connectivity strength between each pair of clusters,

$$\mathbf{A}^{\text{pool}} = \mathbf{S}^T\mathbf{A}\mathbf{S}. \tag{51}$$

Then, we review several representative cluster-based pooling methods in detail.

DiffPool (Ying et al., 2018). DiffPool is one of the first and classic works to hierarchically pool the graph by graph clustering. Specifically, it uses an embedding GNN to generate embeddings of nodes, and a pooling GNN to generate the cluster assignment matrix,

$$\hat{\mathbf{X}} = \text{GNN}_{\text{embed}} \left(\mathbf{X}, \mathbf{A} \right), \mathbf{S} = \text{softmax} \left(\text{GNN}_{\text{pool}} \left(\mathbf{X}, \mathbf{A} \right) \right), \mathbf{X}^{\text{pool}} = \mathbf{S}^T\hat{\mathbf{X}}. \tag{52}$$

Besides, DiffPool leverages an auxiliary link prediction objective $L_{\text{LP}} = \left\| \mathbf{A}, \mathbf{S}\mathbf{S}^T \right\|_F$ to encourage the adjacent nodes to be in the same cluster and avoid fake local minima, where $\| \cdot \|_F$ is the Frobenius norm. And it utilizes an entropy regularization term $L_{\text{E}} = \frac{1}{|V|} \sum_{i=1}^{|V|} H \left(\mathbf{S}_i \right)$ to impel clear cluster assignments, where $H(\cdot)$ represents the entropy.

Graph Pooling with Spectral Clustering (MinCutPool) (Bianchi et al., 2020). MinCutPool takes advantage of the properties of spectral clustering (SC) to provide a better inductive bias and avoid degenerate cluster assignments. It learns to cluster like SC by optimizing the MinCut loss,

$$L_c = -\frac{\text{Tr} \left(\mathbf{S}^T\mathbf{A}\mathbf{S} \right)}{\text{Tr} \left(\mathbf{S}^T\mathbf{D}\mathbf{S} \right)}. \tag{53}$$

In addition, it utilizes an orthogonality loss $L_o = \left\| \frac{\mathbf{S}^T\mathbf{S}}{\|\mathbf{S}^T\mathbf{S}\|_F} - \frac{\mathbf{I}_K}{\sqrt{K}} \right\|_F$ to encourage orthogonal and uniform cluster assignments, and prevent the bad minima of L_c, where K is the number of the clusters. When performing a specific task, it can optimize the weighted sum of the unsupervised loss $L_u = L_c + L_o$ and a task-specific loss to find the optimal balance between the theoretical prior and the task objective.

Structural Entropy Guided Graph Pooling (SEP) (Wu et al., 2022). In order to lessen the local structural harm and suboptimal performance caused by separate pooling layers and predesigned pooling ratios, SEP leverages the concept of structural entropy to generate the global and hierarchical cluster assignments at once. Specifically, SEP treats the nodes of a given graph as the leaf nodes of a coding tree and exploits the hierarchical layers of the coding tree to capture the hierarchical structure of the graph. The optimal code tree T can be obtained by minimizing the structural entropy (Li & Pan, 2016),

$$\mathcal{H}^T(G) = - \sum_{v_i \in T} \frac{g(P_{v_i})}{\text{vol}(V)} \log \frac{\text{vol} \left(P_{v_i} \right)}{\text{vol} \left(P_{v_i^+} \right)}, \tag{54}$$

where v_i^+ represents the father node of node v_i, P_{v_i} denotes the partition of leaf nodes which are descendants of v_i in the coding tree T, $g(P_{v_i})$ denotes the number of edges that have a terminal in the P_{v_i}, and $\text{vol}(\cdot)$ denotes the total degrees of leaf nodes in the given partition. Then, the cluster assignment matrix for each pooling layer can be derived from the edges of each layer in the coding tree. With the help of the one-step joint assignments generation based on structural entropy, it can not only make the best use of the hierarchical relationships of pooling layers, but also reduce the structural noise in the original graph.

5.2.3. Hybrid pooling

Hybrid pooling methods combine TopK-based pooling methods and cluster-based pooling methods, to exert the advantages of the two methods and overcome their respective limitations. Here, we review two representative hybrid pooling methods, Adaptive Structure Aware Pooling (Ranjan et al., 2020) and Multi-channel Pooling (Du et al., 2021).

Adaptive Structure Aware Pooling (ASAP) (Ranjan et al., 2020). Considering that TopK-based pooling methods are not good at capturing the connectivity of the coarsened graph, while cluster-based pooling methods fail to be employed for large graphs because of the dense assignment matrix, ASAP organically combines the two types of pooling methods to overcome the above limitations. Specifically, it regards the h-hop ego-network $c_h(v_i)$ of each node v_i as a cluster. Such local clustering enables the cluster assignment matrix to be sparse. Then, a new self-attention mechanism Master2Token is used to learn the cluster assignment matrix $\mathbf{S}$ and the cluster representations,

$$\mathbf{m}_i = \max_{v_j \in c_h(v_i)} \left(\mathbf{X}'_j \right), \mathbf{S}_{j,i} = \text{softmax} \left(\mathbf{w}^T \sigma \left(\mathbf{W}\mathbf{m}_i \parallel \mathbf{X}'_j \right) \right),$$

$$\mathbf{X}_i^c = \sum_{j=1}^{|c_h(v_i)|} \mathbf{S}_{j,i}\mathbf{X}_j, \tag{55}$$

where $\mathbf{X}'$ is the node embedding matrix after passing GCN, $\mathbf{w}$ and $\mathbf{W}$ denote the trainable vector and matrix respectively, and $\mathbf{X}_i^c$ denotes the representation of the cluster $c_h(v_i)$. Next, it utilizes the graph convolution and TopK selection to choose the top K clusters, whose centers are treated as the nodes of the pooled graph. The adjacency matrix of the pooled graph can be calculated like common cluster-based pooling methods (51), preserving the connectivity of the original graph well.

Multi-channel Pooling (MuchPool) (Du et al., 2021). The key idea of MuchPool is to capture both the local and global structure of a given graph by combining the TopK-based pooling methods and the cluster-based pooling methods. MuchPool has two pooling channels based on TopK selection to yield two fine-grained pooled graphs, whose selection criteria are node degrees and projected scores of node features respectively, so that both the local topology and the node features are considered. Besides, it leverages a channel based on graph clustering to obtain a coarse-grained pooled graph, which captures the global and hierarchical structure of the input graph. To better integrate the information of different channels, a cross-channel convolution is proposed, which fuses the node embeddings of the fine-grained pooled graph $\mathbf{X}^{\text{fine}}$ and the coarse-grained pooled graph $\mathbf{X}^{\text{coarse}}$ with the help of the cluster assignments $\mathbf{S}$ of the cluster-based pooling channel,

$$\widetilde{\mathbf{X}}^{\text{fine}} = \sigma\left(\left[\mathbf{X}^{\text{fine}} + \mathbf{S}\mathbf{X}^{\text{coarse}}\right] \cdot \mathbf{W}\right), \tag{56}$$

where $\mathbf{W}$ denotes the learnable weights. Finally, it merges the node embeddings and the adjacency matrices of the two fine-grained pooled graphs to obtain the eventually pooled graph.

5.3. Summary

This section introduces graph pooling. We provide the summary as follows:

- **Techniques.** Graph pooling methods play a vital role in generating an entire graph representation by aggregating node embeddings. There are mainly two categories of graph pooling methods: global pooling methods and hierarchical pooling methods. While global pooling methods directly aggregate node embeddings in one step, hierarchical pooling methods gradually coarsen a graph to capture hierarchical structure characteristics of the graph based on TopK selection, clustering methods, or hybrid methods.
- **Challenges and Limitations.** Despite the great success of graph pooling methods for learning the whole graph representation, there remain several challenges and limitations unsolved: (1) For hierarchical pooling, most cluster-based methods involve the dense assignment matrix, which limits their application to large graphs, while TopK-based methods are not good at capturing structure information of the graph and may lose information due to node dropping. (2) Most graph pooling methods are designed for simple attributed graphs, while pooling algorithms tailored to other types of graphs, like dynamic graphs and heterogeneous graphs, are largely under-explored.
- **Future Works.** In the future, we expect that more hybrid or other pooling methods can be studied to capture the graph structure information sufficiently as well as be efficient for large graphs. In realistic scenarios, there are various types of graphs involving dynamic, heterogeneous, or spatial–temporal information. It is promising to design graph pooling methods specifically for these graphs, which can be beneficial to more real-world applications, such as traffic analysis and recommendation systems.

6. Graph transformer

Though GNNs based on the message-passing paradigm have achieved impressive performance on multiple well-known tasks

(Gilmer, Schoenholz, Riley, Vinyals, & Dahl, 2017; Li, Xiong, Thabet and Ghanem, 2020; Wang et al., 2019; Xu, Hu et al., 2018), they still face some intrinsic problems due to the iterative neighbor-aggregation operation. Many previous works have demonstrated the two major defects of message-passing GNNs, which are known as the over-smoothing and long-distance modeling problems. And there are lots of explanatory works trying to mine insights from these two issues. The over-smoothing problem can be explained in terms of various GNNs focusing only on low-frequency information (Bo, Wang, Shi, & Shen, 2021), mixing information between different kinds of nodes destroying model performance (Chen et al., 2020), GCN is equivalent to Laplacian smoothing (Li, Han, & Wu, 2018), isotropic aggregation among neighbors leading to the same influence distribution as random walk (Xu et al., 2018), etc. The inability to model long-distance dependencies of GNNs is partially due to the over-smoothing problem, because in the context of conventional neighbor-aggregation GNNs, node information can be passed over long distances only through multiple GNN layers. Recently, Alon and Yahav (2020) found that this problem may also be caused by over-squashing, which means the exponential growth of computation paths with increasing distance. Though the two basic performance bottlenecks can be tackled with elaborate message passing and aggregation strategies, the representational power of GNNs is inherently bounded by the Weisfeiler–Lehman isomorphism hierarchy (Morris et al., 2019). Worse still, most GNNs (Gilmer et al., 2017; Kipf & Welling, 2016a; Veličković et al., 2017) are bounded by the simplest first-order Weisfeiler–Lehman test (1-WL). Some efforts have been dedicated to break this limitation, such as hypergraph-based (Feng, You, Zhang, Ji, & Gao, 2019; Huang & Yang, 2021), path-based (Cai & Lam, 2020; Ying et al., 2021), and k-WL-based (Balcilar et al., 2021; Morris et al., 2019) approaches.

Among many attempts to solve these fundamental problems, an essential one is the adaptation of Transformer (Vaswani et al., 2017) for graph representation learning. Transformers, both the vanilla version and several variants, have been adopted with impressive results in various deep learning fields including natural language processing (Devlin, Chang, Lee, & Toutanova, 2018; Vaswani et al., 2017), computer vision (Carion et al., 2020; Zhu et al., 2020), etc. Recently, Transformer also shows powerful graph modeling abilities in many researches (Chen, O'Bray et al., 2022; Dwivedi & Bresson, 2020; Kreuzer, Beaini, Hamilton, Létourneau, & Tossou, 2021; Ma et al., 2023; Wu et al., 2021; Ying et al., 2021; Zhao, Ma, Zhang, Deng and Wei, 2023). Extensive empirical results show that some chronic shortcomings in conventional GNNs can be easily overcome in Transformer-based approaches. This section gives an overview of the current progress on this kind of method.

6.1. Transformer

Transformer (Vaswani et al., 2017) was first applied to model machine translation, but two of the key mechanisms adopted in this work, attention operation and positional encoding, are highly compatible with the graph modeling problem.

To be specific, we denote the input of attention layer in Transformer as $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{n-1}]$, $\mathbf{x}_i \in \mathbb{R}^d$, where n is the length of input sequence and d is the dimension of each input embedding $\mathbf{x}_i$. Then the core operation of calculating new embedding $\hat{\mathbf{x}}_i$ for each $\mathbf{x}_i$ in attention layer can be streamlined as:

$$s^h(\mathbf{x}_i, \mathbf{x}_j) = \text{NORM}_j(\underset{\mathbf{x}_k \in \mathbf{X}}{\|} \mathcal{Q}^h(\mathbf{x}_i)^{\text{T}} \mathcal{K}^h(\mathbf{x}_k)),$$

$$\mathbf{x}_i^h = \sum_{\mathbf{x}_j \in \mathbf{X}} s^h(\mathbf{x}_i, \mathbf{x}_j) \mathcal{V}^h(\mathbf{x}_j), \tag{57}$$

$$\hat{\mathbf{x}}_i = \text{MERGE}(\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^H),$$

where $h \in \{0, 1, \dots, H-1\}$ represents the attention head number. $\mathcal{Q}^h$, $\mathcal{K}^h$ and $\mathcal{V}^h$ are projection functions mapping a vector to the query space, key space and value space respectively. $s^h(\mathbf{x}_i, \mathbf{x}_j)$ is score function

Table 5
Summary of graph transformer methods.

Method	Technique		Capacity		
	Attention modification	Encoding enhancement	Heterogeneous	Long distance	>1-WL
GGT (Dwivedi & Bresson, 2020)	✓	✓		Structure only	✓
GTSA (Kreuzer et al., 2021)	✓	✓		✓	✓
HGT (Hu, Dong, Wang and Sun, 2020)	✓		✓		
G2SHGT (Yao, Wang, & Wan, 2020)	✓		✓	✓	
HINormer (Mao, Liu, Liu and Sun, 2023)	✓		✓	✓	
GRUGT (Cai & Lam, 2020)	✓			✓	✓
GRIT (Ma et al., 2023)	✓			✓	✓
Graphormer-GD (Zhang, Luo, Wang and He, 2023)	✓			✓	✓
Graphormer (Ying et al., 2021)	✓	✓		✓	✓
GSGT (Hussain, Zaki, & Subramanian, 2021)		✓		✓	✓
TMDG (Geisler et al., 2023)		✓		✓	✓
Graph-BERT (Zhang, Zhang, Xia and Sun, 2020)		✓		✓	✓
LRGT (Wu et al., 2021)		✓		✓	
SAT (Chen, O'Bray et al., 2022)		✓		✓	✓

measuring the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. NORM is the normalization operation ensuring $\sum_{\mathbf{x}_j \in \mathbf{X}} s^h(\mathbf{x}_i, \mathbf{x}_j) \equiv 1$ to propel the stability of the output generated by a stack of attention layers, it is usually performed as scaled softmax: $\text{NORM}(\cdot) = \text{SoftMax}(\cdot/\sqrt{d})$. And MERGE function is designed to combine the information extracted from multiple attention heads. Here, we omit further implementation details that do not affect our understanding of attention operation.

The attention process cannot encode the position information of each $\mathbf{x}_i$, which is essential in machine translation problems. So positional encoding is introduced to remedy this deficiency, and it is calculated as:

$$\mathbf{X}_{i,2j}^{pos} = \sin(i/10000^{2j/d}), \quad \mathbf{X}_{i,2j+1}^{pos} = \cos(i/10000^{2j/d}), \tag{58}$$

where i is the position and j is the dimension. The positional encoding is added to the input before it is fed to the Transformer.

6.2. Overview

From the simplified process shown in Eq. (57), we can see that the core of the attention operation is to accomplish information transfer based on the similarity between the source and the target to be updated. It is quite similar to the message-passing process on a fully-connected graph. However, the direct application of this architecture to arbitrary graphs does not make use of structural information, so it may lead to poor performance when graph topology is important. On the other hand, the definition of positional encoding in graphs is not a trivial problem because the order or coordinates of graph nodes are underdefined.

According to these two challenges, Transformer-based methods for graph representation learning can be classified into two major categories, one considering graph structure during the attention process, and the other encoding the topological information of the graph into initial node features. We name the first one as *Attention Modification* and the second one as *Encoding Enhancement*. A summarization is provided in Table 5. In the following discussion, if both methods are used in one paper, we will list them in different subsections, and we will ignore the multi-head trick in attention operation.

6.3. Attention modification

This group of works attempts to modify the full attention operation to capture structure information. The most prevalent approach is changing the score function, which is denoted as $s(\cdot, \cdot)$ in Eq. (57). GGT (Dwivedi & Bresson, 2020) constrains each node feature can only attend to neighbors and enables the model to represent edge feature

information by rewrite $s(\cdot, \cdot)$ as:

$$\tilde{s}_1(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} (\mathbf{W}^Q \mathbf{x}_i)^{\mathrm{T}}(\mathbf{W}^K \mathbf{x}_j \odot \mathbf{W}^E \mathbf{e}_{ji}), & \langle j, i \rangle \in E \\ -\infty, & \text{otherwise} \end{cases},$$
$$s_1(\mathbf{x}_i, \mathbf{x}_j) = \text{SoftMax}_j(\underset{\mathbf{x}_k \in \mathbf{X}}{\|} \tilde{s}_1(\mathbf{x}_i, \mathbf{x}_k)), \tag{59}$$

where $\odot$ is Hadamard product and $\mathbf{W}^{Q,K,E}$ represents trainable parameter matrix. This approach is not efficient yet to model long-distance dependencies since only 1st-neighbors are considered. Though it adopts Laplacian eigenvectors to gather global information (see Section 6.4), but only long-distance structure information is remedied while the node and edge features are not. GTSA (Kreuzer et al., 2021) improves this approach by combining the original graph and the full graph. Specifically, it extends $s_1(\cdot, \cdot)$ to:

$$\tilde{s}_2(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} (\mathbf{W}_1^Q \mathbf{x}_i)^{\mathrm{T}}(\mathbf{W}_1^K \mathbf{x}_j \odot \mathbf{W}_1^E \mathbf{e}_{ji}), & \langle j, i \rangle \in E \\ (\mathbf{W}_0^Q \mathbf{x}_i)^{\mathrm{T}}(\mathbf{W}_0^K \mathbf{x}_j \odot \mathbf{W}_0^E \mathbf{e}_{ji}), & \text{otherwise} \end{cases},$$
$$s_2(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \dfrac{1}{1+\lambda} \text{SoftMax}_j(\underset{\langle k,i \rangle \in E}{\|} \tilde{s}_2(\mathbf{x}_i, \mathbf{x}_k)), & \langle j, i \rangle \in E \\ \dfrac{\lambda}{1+\lambda} \text{SoftMax}_j(\underset{\langle k,i \rangle \notin E}{\|} \tilde{s}_2(\mathbf{x}_i, \mathbf{x}_k)), & \text{otherwise} \end{cases}, \tag{60}$$

where λ is a hyperparameter representing the strength of the full connection.

Some works try to reduce information-mixing problems (Chen, Lin et al., 2020) in heterogeneous graphs. HGT (Hu, Dong, Wang and Sun, 2020) disentangles the attention of different node types and edge types by adopting additional attention heads. It defines $\mathbf{W}_{Q,K,V}^{\tau(v)}$ for each node type $\tau(v)$ and $\mathbf{W}_E^{\phi(e)}$ for each edge type $\phi(e)$, $\tau(\cdot)$ and $\phi(\cdot)$ are type indicating function. G2SHGT (Yao et al., 2020) defines four types of subgraphs, *fully-connected*, *connected*, *default* and *reverse*, to capture global, undirected, forward and backward information respectively. Each subgraph is homogeneous, so it can reduce interactions between different classes.

Path features between nodes are always treated as inductive bias added to the original score function. Let $\text{SP}_{ij} = (e_1, e_2, \dots, e_N)$ denote the shortest path between node pair (v_i, v_j). GRUGT (Cai & Lam, 2020) uses GRU (Chung, Gulcehre, Cho, & Bengio, 2014) to encode forward and backward features as: $\mathbf{r}_{ij} = \text{GRU}(\text{SP}_{ij})$, $\mathbf{r}_{ji} = \text{GRU}(\text{SP}_{ji})$. Then, the final attention score is calculated by adding up four components:

$$\tilde{s}_3(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{W}^Q \mathbf{x}_i)^{\mathrm{T}} \mathbf{W}^K \mathbf{x}_j + (\mathbf{W}^Q \mathbf{x}_i)^{\mathrm{T}} \mathbf{W}^K \mathbf{r}_{ji} + (\mathbf{W}^Q \mathbf{r}_{ij})^{\mathrm{T}} \mathbf{W}^K \mathbf{x}_j$$
$$+ (\mathbf{W}^Q \mathbf{r}_{ij})^{\mathrm{T}} \mathbf{W}^K \mathbf{r}_{ji}, \tag{61}$$

from front to back, which represent content-based score, source-dependent bias, target-dependent bias and universal bias respectively. Graphormer (Ying et al., 2021) uses both path length and path

embedding to introduce structural bias as:

$$\tilde{s}_4(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{W}^Q \mathbf{x}_i)^{\mathrm{T}} \mathbf{W}^K \mathbf{x}_j / \sqrt{d} + b_N + c_{ij},$$

$$c_{ij} = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{e}_k)^{\mathrm{T}} \mathbf{w}_k^E, \tag{62}$$

$$s_4(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{SoftMax}_j(\underset{\mathbf{x}_k \in \mathbf{X}}{\|} \tilde{s}_4(\mathbf{x}_i, \mathbf{x}_k)),$$

where b_N is a trainable scalar indexed by N, the length of SP_{ij}. $\mathbf{e}_k$ is the embedding of the edge e_k, and $\mathbf{w}_k^E \in \mathbb{R}^d$ is the kth edge parameter. If SP_{ij} does not exist, then b_N and c_{ij} are set to be special values. GRIT (Ma, Lin et al., 2023) utilizes relative random walk probabilities as an inductive bias to encode relative path information. Graphormer-GD (Zhang, Luo et al., 2023) also incorporates relative distance as bias, and rigorously proves that this bias is crucial for determining the biconnectivity of a graph.

6.4. Encoding enhancement

This kind of method intends to enhance initial node representations to enable the Transformer to encode structure information. They can be further divided into two categories, position-analogy methods and structure-aware methods.

6.4.1. Position-analogy methods

In Euclidean space, the Laplacian operator corresponds to the divergence of the gradient, whose eigenfunctions are sine/cosine functions. For the graph, the Laplacian operator is the Laplacian matrix, whose eigenvectors can be considered as eigenfunctions. Hence, inspired by Eq. (58), position-analogy methods utilize Laplacian eigenvectors to simulate positional encoding $\mathbf{X}^{pos}$ as they are the equivalents of sine/cosine functions.

Laplacian eigenvectors can be calculated via the eigendecomposition of normalized graph Laplacian matrix $\tilde{\mathbf{L}}$:

$$\tilde{\mathbf{L}} \triangleq \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^{\mathrm{T}}, \tag{63}$$

where $\mathbf{A}$ is the adjacency matrix, $\mathbf{D}$ is the degree matrix, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n-1}]$ are eigenvectors and $\mathbf{\Lambda} = diag(\lambda_0, \lambda_1, \ldots, \lambda_{n-1})$ are eigenvalues. With $\mathbf{U}$ and $\mathbf{\Lambda}$, GGT (Dwivedi & Bresson, 2020) uses eigenvectors of the k smallest non-trivial eigenvalues to denote the intermediate embedding $\mathbf{X}^{mid} \in \mathbb{R}^{n \times k}$, and maps it to d-dimensional space and gets the position encoding $\mathbf{X}^{pos} \in \mathbb{R}^{n \times d}$. This process can be formalized as:

$$index = \mathrm{argmin}_k(\{\lambda_i | 0 \leq i < n \wedge \lambda_i > 0\}),$$

$$\mathbf{X}^{mid} = [\mathbf{u}_{index_0}, \mathbf{u}_{index_1}, \ldots, \mathbf{u}_{index_{k-1}}]^{\mathrm{T}}, \tag{64}$$

$$\mathbf{X}^{pos} = \mathbf{X}^{mid} \mathbf{W}^{k \times d},$$

where $index$ is the subscript of the selected eigenvectors. GTSA (Kreuzer et al., 2021) puts eigenvector $\mathbf{u}_i$ on the frequency axis at λ_i and uses sequence modeling methods to generate positional encoding. Specifically, it extends $\mathbf{X}^{mid}$ in Eq. (64) to $\tilde{\mathbf{X}}^{mid} \in \mathbb{R}^{n \times k \times 2}$ by concatenating each value in eigenvectors with corresponding eigenvalue, and then positional encoding $\mathbf{X}^{pos} \in \mathbb{R}^{n \times d}$ are generated as:

$$\mathbf{X}^{input} = \tilde{\mathbf{X}}^{mid} \mathbf{W}^{2 \times d},$$
$$\mathbf{X}^{pos} = \mathrm{SumPooling}(\mathrm{Transformer}(\mathbf{X}^{input}), \dim = 1). \tag{65}$$

Here, $\mathbf{X}_{input} \in \mathbb{R}^{n \times k \times d}$ is equivalent to the input matrix in sequence modeling with shape $(batch\_size, length, dim)$, and can be naturally processed by Transformer. Since the Laplacian eigenvectors can be complex-valued for directed graph, GSGT (Hussain et al., 2021) proposes to utilize SVD of adjacency matrix $\mathbf{A}$, which is denoted as $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$, and uses the largest k singular values $\mathbf{\Sigma}_k$ and associated left and right singular vectors $\mathbf{U}_k$ and $\mathbf{V}_k^{\mathrm{T}}$ to output $\mathbf{X}^{pos}$ as $\mathbf{X}^{pos} = [\mathbf{U}_k \mathbf{\Sigma}_k^{1/2} \|$ $\mathbf{V}_k \mathbf{\Sigma}_k^{1/2}]$, where $\|$ is the concatenation operation. In addition to SVD, TMDG (Geisler et al., 2023) processes directed graphs by utilizing the Magnetic Laplacian. All these methods above randomly flip the signs of

eigenvectors or singular vectors during the training phase to promote the invariance of the models to the sign ambiguity.

6.4.2. Structure-aware methods

In contrast to position-analogy methods, structure-aware methods do not attempt to mathematically rigorously simulate sequence positional encoding. They use some additional mechanisms to directly calculate structure-related encoding.

Some approaches compute extra encoding $\mathbf{X}^{add}$ and add it to the initial node representation. Graphormer (Ying et al., 2021) proposes to leverage node centrality as an additional signal to address the importance of each node. Concretely, $\mathbf{x}_i^{add}$ is determined by the in-degree $\deg_i^-$ and outdegree $\deg_i^+$:

$$\mathbf{x}_i^{add} = \mathcal{P}^-(\deg_i^-) + \mathcal{P}^+(\deg_i^+), \tag{66}$$

where $\mathcal{P}^-$ and $\mathcal{P}^+$ are learnable embedding function. Graph-BERT (Zhang, Zhang et al., 2020) employs Weisfeiler–Lehman algorithm to label node v_i to a number $\mathrm{WL}(v_i) \in \mathbb{N}$ and defines $\mathbf{x}_i^{add}$ as:

$$\mathbf{x}_{i,2j}^{add} = \sin(\mathrm{WL}(v_i)/10000^{2j/d}), \ \mathbf{x}_{i,2j+1}^{add} = \cos(\mathrm{WL}(v_i)/10000^{2j/d}). \tag{67}$$

The other approaches try to leverage GNNs to initialize inputs to the Transformer. LRGT (Wu et al., 2021) applies GNN to get intermediate vectors as $\mathbf{X}' = \mathrm{GNN}(\mathbf{X})$, and passes the concatenation of $\mathbf{X}'$ and a special vector $\mathbf{x}_{\mathrm{CLS}}$ to Transformer layer as: $\hat{\mathbf{X}} = \mathrm{Transformer}([\mathbf{X}' \|$ $\mathbf{x}_{\mathrm{CLS}}])$. Then $\hat{\mathbf{x}}_{\mathrm{CLS}}$ can be used as the representation of the entire graph for downstream tasks. This method cannot break the 1-WL bottleneck because it uses GCN (Kipf & Welling, 2016a) and GIN (Xu, Hu et al., 2018) as graph encoders in the first step, which are intrinsically limited by 1-WL test. SAT (Chen, O'Bray et al., 2022) improves this deficiency by using subgraph-GNN NGNN (Zhang & Li, 2021) for initialization, and achieves outstanding performance.

6.5. Summary

This section introduces Transformer-based approaches for graph representation learning and we provide the summary as follows:

- **Techniques.** Graph Transformer methods modify two fundamental techniques in Transformer, attention operation and positional encoding, to enhance its ability to encode graph data. Typically, they introduce fully connected attention to model long-distance relationships, utilize shortest path and Laplacian eigenvectors to break 1-WL bottleneck, and separate points and edges belonging to different classes to avoid over-mixing problems.

- **Challenges and Limitations.** Though Graph Transformers achieve encouraging performance, they still face two major challenges. The first challenge is the computational cost of the quadratic attention mechanism and shortest path calculation. These operations require significant computing resources and can be a bottleneck, particularly for large graphs. The second is the reliance of Transformer-based models on large amounts of data for stable performance. It poses a challenge when dealing with problems that lack sufficient data, especially for few-shot and zero-shot settings.

- **Future Works.** We expect efficiency improvement for Graph Transformer should be further explored. Additionally, there are some works using pre-training and fine-tuning frameworks to balance performance and complexity in downstream tasks (Ying et al., 2021), this may be a promising solution to address the aforementioned two challenges.

7. Semi-supervised learning on graphs

We have investigated various architectures of graph neural networks in which the parameters should be tuned by a learning objective.

Table 6
Summary of methods for semi-supervised learning on graphs. Contrastive learning can be considered as a specific kind of consistency learning.

	Approach	Pseudo-labeling	Consistency learning	Knowledge distillation	Active learning
Node-level	CoGNet (Li, Yang et al., 2022)	✓			
	DSGCN (Zhou et al., 2019)	✓			
	GRAND (Feng et al., 2020)		✓		
	AugGCR (Park et al., 2021)		✓		
	HCPL (Luo et al., 2023)	✓			
Graph-level	SEAL (Li et al., 2019)	✓			✓
	InfoGraph (Sun, Hoffmann, Verma and Tang, 2020)		✓	✓	
	DSGC (Yang et al., 2022)		✓		
	ASGN (Hao et al., 2020)			✓	✓
	TGNN (Ju et al., 2022)		✓		
	KGNN (Ju, Yang et al., 2022)	✓			
	HGMI (Li, Huang, Chang and Rong, 2022)	✓	✓		
	ASGNN (Xie, Lv, Qian, Wen and Liang, 2022)	✓			✓
	DualGraph (Luo et al., 2022)	✓	✓		
	GLA (Yue, Zhang, Zhang, & Liu, 2022)	✓	✓		
	SS (Xie et al., 2022)	✓			

The most prevalent optimization approach is supervised learning on graph data. Due to the label deficiency, semi-supervised learning has attracted increasing attention in the data mining community. In detail, these methods attempt to combine graph representation learning with current semi-supervised techniques including pseudo-labeling, consistency learning, knowledge distillation and active learning. These works can be further subdivided into node-level representation learning and graph-level representation learning. We would introduce both parts in detail as in Sections 7.1 and 7.2, respectively. A summarization is provided in Table 6.

7.1. Node representation learning

Typically, node representation learning follows the concept of transductive learning, which has access to test unlabeled data. We first review the simplest loss objective, i.e., node-level supervised loss. This loss exploits the ground truth of labeled nodes on graphs. The standard cross-entropy is usually adopted for optimization. In formulation,

$$\mathcal{L}_{NSL} = -\frac{1}{|\mathcal{Y}^L|} \sum_{i \in \mathcal{Y}^L} \mathbf{y}_i^T \log \mathbf{p}_i, \tag{68}$$

where $\mathcal{Y}^L$ denotes the set of labeled nodes. Additionally, there are a variety of unlabeled nodes that can be used to offer semantic information. To fully utilize these nodes, a range of methods attempt to combine semi-supervised approaches with graph neural networks. Pseudo-labeling (Lee et al., 2013) is a fundamental semi-supervised technique that uses the classifier to produce the label distribution of unlabeled examples and then adds appropriately labeled examples to the training set (Li, Yang, Pagnucco and Song, 2022; Zhou, Zhang, & Huang, 2019). Another line of semi-supervised learning is consistency regularization (Laine & Aila, 2016) that requires two examples to have identical predictions under perturbation. This regularization is based on the assumption that each instance has a distinct label that is resistant to random perturbations (Feng et al., 2020; Park et al., 2021). Then, we show several representative works in detail.

Cooperative Graph Neural Networks (CoGNet) (Li, Yang et al., 2022). CoGNet is a representative pseudo-label-based GNN approach for semi-supervised node classification. It employs two GNN classifiers to jointly annotate unlabeled nodes. In particular, it calculates the confidence of each node as follows:

$$CV(\mathbf{p}_i) = \mathbf{p}_i^T \log \mathbf{p}_i, \tag{69}$$

where $\mathbf{p}_i$ denotes the output label distribution. Then it selects the pseudo-labels with high confidence generated from one model to supervise the optimization of the other model. In particular, the objective for unlabeled nodes is written as follows:

$$\mathcal{L}_{CoGNet} = \sum_{i \in \mathcal{V}^U} \mathbf{1}_{CV(\mathbf{p}_i) > \tau} \hat{\mathbf{y}}_i^T \log \mathbf{q}_i, \tag{70}$$

where $\hat{\mathbf{y}}_i$ denotes the one-hot formulation of the pseudo-label $\hat{y}_i = argmax\mathbf{p}_i$ and $\mathbf{q}_i$ denotes the label distribution predicted by the other classifier. τ is a pre-defined temperature coefficient. This cross supervision has been demonstrated effective in Chen, Yuan, Zeng and Wang (2021) and Luo et al. (2021) to prevent the provision of biased pseudo-labels. Moreover, it employs GNNExplainer (Ying, Bourgeois, You, Zitnik, & Leskovec, 2019) to provide additional information from a dual perspective. Here it measures the minimal subgraphs where GNN classifiers can still generate the same prediction. In this way, CoGNet can illustrate the entire optimization process to enhance our understanding. HCPL (Luo, Ju, Gu, Qin et al., 2023) incorporates curriculum learning into pseudo-labeling in semi-supervised node classification, which can generate dynamics thresholds for reliable nodes.

Dynamic Self-training Graph Neural Network (DSGCN) (Zhou et al., 2019). DSGCN develops an adaptive manner to utilize reliable pseudo-labels for unlabeled nodes. In particular, it allocates smaller weights to samples with lower confidence with the additional consideration of class balance. The weight is formulated as:

$$\omega_i = \frac{1}{n_{c^i}} \max \left(\text{RELU} \left(\mathbf{p}_i - \beta \cdot \mathbf{1} \right) \right), \tag{71}$$

where n_{c^i} denotes the number of unlabeled samples assigned to the class c^i. This technique will decrease the impact of wrong pseudo-labels during iterative training.

Graph Random Neural Networks (GRAND) (Feng et al., 2020). GRAND is a representative consistency learning-based method. It first adds a variety of perturbations to the input graph to generate a list of graph views. Each graph view G^r is sent to a GNN classifier to produce a prediction matrix $\mathbf{P}^r = [\mathbf{p}_1^r, \dots, \mathbf{p}_N^r]$. Then it summarizes these matrices as:

$$\mathbf{P} = \frac{1}{R} \mathbf{P}^r. \tag{72}$$

To provide more discriminative information and ensure that the matrix is row-normalized, GRAND sharpens the summarized label matrix into $\mathbf{P}^{SA}$ as:

$$\mathbf{P}_{ij}^{SA} = \frac{\mathbf{P}_{ij}^{1/T}}{\sum_{j'=0} \mathbf{P}_{ij'}^{1/T}}, \tag{73}$$

where T is a given temperature parameter. Finally, consistency learning is performed by comparing the sharpened summarized matrix with the matrix of each graph view. Formally, the objective is:

$$\mathcal{L}_{GRAND} = \frac{1}{R} \sum_{r=1}^{R} \sum_{i \in V} \|\mathbf{P}_i^{SA} - \mathbf{P}_i\|, \tag{74}$$

here $\mathcal{L}_{GRAND}$ serves as a regularization which is combined with the standard supervised loss.

Augmentation for GNNs with the Consistency Regularization (AugGCR) (Park et al., 2021). AugGCR begins with the generation of augmented

graphs by random dropout and mixup of different order features. To enhance the model generalization, it borrows the idea of meta-learning to partition the training data, which improves the quality of graph augmentation. In addition, it utilizes consistency regularization to enhance the semi-supervised node classification.

7.2. Graph representation learning

The objective of graph classification is to predict the property of the whole graph example. Assuming that the training set comprises N^l and N^u graph samples $\mathcal{G}^l = \{G^1, \ldots, G^{N^l}\}$ and $\mathcal{G}^u = \{G^{N^l+1}, \ldots, G^{N^l+N^u}\}$, the graph-level supervised loss for labeled data can be expressed as follows:

$$\mathcal{L}_{GSL} = -\frac{1}{|\mathcal{G}^u|} \sum_{G_j \in \mathcal{G}^L} \mathbf{y}^{j^T} log \mathbf{p}^j, \tag{75}$$

where $\mathbf{y}^j$ denotes the one-hot label vector for the jth sample while $\mathbf{p}^j$ denotes the predicted distribution of G^j. When $N^u = 0$, this objective can be utilized to optimize supervised methods. However, due to the shortage of labels in graph data, supervised methods cannot reach exceptional performance in real-world applications (Hao et al., 2020; Liu, Zhao, Inae, Luo and Jiang, 2023; Mao, Ju, Qin, Luo and Zhang, 2023; Yi et al., 2023). To tackle this, semi-supervised graph classification has been developed extensively. These approaches can be categorized into pseudo-labeling-based methods, knowledge distillation-based methods and contrastive learning-based methods. Pseudo-labeling methods annotate graph instances and utilize well-classified graph examples to update the training set (Ju et al., 2024; Li, Huang et al., 2022; Li, Rong et al., 2019). Knowledge distillation-based methods usually utilize a teacher-student architecture, where the teacher model conducts graph representation learning without label information to extract generalized knowledge while the student model focuses on the downstream task. Due to the restricted number of labeled instances, the student model transfers knowledge from the teacher model to prevent overfitting (Hao et al., 2020; Sun, Hoffmann et al., 2020). Another line of this topic is to utilize graph contrastive learning, which is frequently used in unsupervised learning. Typically, these methods extract topological information from two perspectives (i.e., different perturbation strategies and graph encoders), and maximize the similarity of their representations compared with those from other examples (Ju, Luo, Ma et al., 2022; Ju, Luo, Qu et al., 2022; Luo, Ju, Qu, Chen et al., 2022). Active learning, as a prevalent technique to improve the efficiency of data annotation, has also been utilized for semi-supervised methods (Hao et al., 2020; Xie, Lv et al., 2022). Then, we review these methods in detail.

SEmi-supervised grAph cLassification (SEAL) (Li, Rong et al., 2019). SEAL treats each graph example as a node in a hierarchical graph. It builds two graph classifiers which generate graph representations and conduct semi-supervised graph classification respectively. SEAL employs a self-attention module to encode each graph into a graph-level representation, and then conducts message passing from a graph level for final classification. SEAL can also be combined with cautious iteration and active iteration. The former merely utilizes partial graph samples to optimize the parameters in the first classifier due to the potential erroneous pseudo-labels. The second combines active learning with the model, which increases the annotation efficiency in semi-supervised scenarios.

InfoGraph (Sun, Hoffmann et al., 2020). Infograph is the first contrastive learning-based method. It maximizes the similarity between summarized graph representations and their node representations. In particular, it generates node representations using the message passing mechanism and summarizes these node representations into a graph representation. Let $\Phi(\cdot, \cdot)$ denote a discriminator to distinguish whether

a node belongs to the graph, and we have:

$$\mathcal{L}_{InfoGraph} = \sum_{j=1}^{|\mathcal{G}^l|+|\mathcal{G}^u|} \sum_{i \in \mathcal{G}_j} \left[-\operatorname{sp}\left(-\Phi\left(\mathbf{h}_i^j, \mathbf{z}^j\right)\right) \right]$$
$$- \frac{1}{|N_i^j|} \sum_{i',j' \in N_i^j} \left[\operatorname{sp}\left(\Phi\left(\mathbf{h}_{i'}^{j'}, \mathbf{z}^j\right)\right) \right], \tag{76}$$

where $\operatorname{sp}(\cdot)$ denotes the softplus function. N_i^j denotes the negative node set where nodes are not in G^j. This mutual information maximization formulation is originally developed for unsupervised learning and it can be simply extended for semi-supervised graph classification. In particular, InfoGraph utilizes a teacher-student architecture that compares the representation across the teacher and student networks. The contrastive learning objective serves as a regularization by combining with supervised loss.

Dual Space Graph Contrastive Learning (DSGC) (Yang, Chen et al., 2022). DSGC is a representative contrastive learning-based method. It utilizes two graph encoders. The first is a standard GNN encoder in the Euclidean space and the second is the hyperbolic GNN encoder. The hyperbolic GNN encoder first converts graph embeddings into hyperbolic space and then measures the distance based on the length of geodesics. DSGC compares graph embeddings in the Euclidean space and hyperbolic space. Assuming the two GNNs are named as $f_1(\cdot)$ and $f_2(\cdot)$, the positive pair is denoted as:

$$\mathbf{z}_{E \to H}^j = \exp_{\mathbf{o}}^c(f_1(G^j)),$$
$$\mathbf{z}_H^j = \exp_{\mathbf{o}}^c\left(f_2(G^j)\right). \tag{77}$$

Then it selects one labeled sample and N_B unlabeled sample G^j for graph contrastive learning in the hyperbolic space. In formulation,

$$\mathcal{L}_{DSGC} = -\log \frac{e^{d^H\left(\mathbf{h}_H^i, \mathbf{z}_{E \to H}^i\right)/\tau}}{e^{d^H\left(\mathbf{z}_H^j, \mathbf{z}_{E \to H}^i\right)/\tau} + \sum_{i=1}^N e^{d_{\mathbb{D}}\left(\mathbf{z}_{E \to H}^j, \mathbf{z}_H^j\right)/\tau}}$$
$$- \frac{\lambda_u}{N} \sum_{i=1}^N \log \frac{e^{d_{\mathbb{D}}^u\left(\mathbf{z}_H^j, \mathbf{z}_{E \to H}^i\right)/\tau}}{e^{d_{\mathbb{D}}^u\left(\mathbf{z}_H^j, \mathbf{z}_{E \to H}^j\right)/\tau} + e^{d_{\mathbb{D}}\left(\mathbf{z}_H^i, \mathbf{z}_{E \to H}^i\right)/\tau}}, \tag{78}$$

where $\mathbf{z}_{E \to H}^i$ and $\mathbf{z}_H^i$ denote the embeddings for labeled graph sample G^i and $d^H(\cdot)$ denotes a distance metric in the hyperbolic space. This contrastive learning objective maximizes the similarity between embeddings learned from two encoders compared with other samples. Finally, the contrastive learning objective can be combined with the supervised loss to achieve effective semi-supervised contrastive learning.

Active Semi-supervised Graph Neural Network (ASGN) (Hao et al., 2020). ASGN utilizes a teacher-student architecture with the teacher model focusing on representation learning and the student model targeting at molecular property prediction. In the teacher model, ASGN first employs a message passing neural network to learn node representations under the reconstruction task and then borrows the idea of balanced clustering to learn graph-level representations in a self-supervised fashion. In the student model, ASGN utilizes label information to monitor the model training based on the weights of the teacher model. In addition, active learning is also used to minimize the annotation cost while maintaining sufficient performance. Typically, the teacher model seeks to provide discriminative graph-level representations without labels, which transfer knowledge to the student model to overcome the potential overfitting in the presence of label scarcity.

Twin Graph Neural Networks (TGNN) (Ju, Luo, Qu et al., 2022). TGNN also uses two graph neural networks to give different perspectives to learn graph representations. Differently, it adopts a graph kernel neural network to learn graph-level representations in virtue of random walk kernels. Rather than directly enforcing representation from two modules to be similar, TGNN exchanges information by contrasting the similarity structure of the two modules. In particular, it constructs a list of anchor graphs, $G^{a_1}, G^{a_2}, \ldots, G^{a_M}$, and utilizes two graph encoders to produce their embeddings, i.e., $\{z^{a_m}\}_{m=1}^M$, $\{w^{a_m}\}_{m=1}^M$.

Then it calculates the similarity distribution between each unlabeled graph and anchor graphs for two modules. Formally,

$$p_m^j = \frac{\exp\left(\cos\left(z^j, z^{a_m}\right)/\tau\right)}{\sum_{m'=1}^M \exp\left(\cos\left(z^j, z^{a_{m'}}\right)/\tau\right)}, \tag{79}$$

$$q_m^j = \frac{\exp\left(\cos\left(\mathbf{w}^j, \mathbf{w}^{a_m}\right)/\tau\right)}{\sum_{m'=1}^M \exp\left(\cos\left(\mathbf{w}^j, \mathbf{w}^{a_{m'}}\right)/\tau\right)}. \tag{80}$$

Then, TGNN minimizes the distance between distributions from different modules as follows:

$$\mathcal{L}_{TGNN} = \frac{1}{|\mathcal{G}^U|} \sum_{G^j \in \mathcal{G}^u} \frac{1}{2}\left(D_{KL}\left(\mathbf{p}^j \parallel \mathbf{q}^j\right) + D_{KL}\left(\mathbf{q}^j \parallel \mathbf{p}^j\right)\right), \tag{81}$$

which serves as a regularization term to combine with the supervised loss.

7.3. Summary

This section introduces semi-supervised learning for graph representation learning and we provide the summary as follows:

- **Techniques.** Classic node classification aims to conduct transductive learning on graphs with access to unlabeled data, which is a natural semi-supervised problem. Semi-supervised graph classification aims to relieve the requirement of abundant labeled graphs. Here, a variety of semi-supervised methods have been put forward to achieve better performance under the label scarcity. Typically, they try to integrate semi-supervised techniques such as active learning, pseudo-labeling, consistency learning, and consistency learning with graph representation learning.
- **Challenges and Limitations.** Despite their great success, the performance of these methods is still unsatisfactory, especially in graph-level representation learning. For example, DSGC can only achieve an accuracy of 57% in a binary classification dataset REDDIT-BINARY. Even worse, label scarcity is often accompanied by unbalanced datasets and potential domain shifts, which provides more challenges from real-world applications.
- **Future Works.** In the future, we expect that these methods can be applied to different problems such as molecular property predictions. There are also works to extend graph representation learning in more realistic scenarios like few-shot learning (Chauhan, Nathani, & Kaul, 2020; Ma et al., 2020). A higher accuracy is always anticipated for more advanced and effective semi-supervised techniques.

8. Graph self-supervised learning

Besides supervised or semi-supervised methods, self-supervised learning (SSL) also has shown its powerful capability in data mining and representation embedding in recent years. In this section, we investigated Graph Neural Networks based on SSL and provided a detailed introduction to a few typical models. Graph SSL methods usually have a unified pipeline, which includes pretext tasks and downstream tasks. Pretext tasks help the model encoder to learn better representation, as a premise of better performance in downstream tasks. So a delicate design of pretext task is crucial for Graph SSL. We would firstly introduce the overall framework of Graph SSL in Section 8.1, then introduce the two kinds of pretext task design, generation-based methods and contrast-based methods respectively in Sections 8.2 and 8.3. A summarization is provided in Table 7.

8.1. Overall framework

Consider a featured graph $\mathcal{G}$, we denote a graph encoder f to learn the representation of the graph, and a pretext decoder g with specific architecture in different pretext tasks. Then the pretext self-supervised learning loss can be formulated as:

$$\mathcal{L}_{total} = E_{\mathcal{G} \sim \mathcal{D}}[\mathcal{L}_{ssl}(g, f, \mathcal{G})], \tag{82}$$

where $\mathcal{D}$ denotes the distribution of featured graph $\mathcal{G}$. By minimizing $\mathcal{L}_{overall}$, we can learn encoder f with capacity to produce high-quality embedding. As for downstream tasks, we denote a graph decoder d which transforms the output of graph encoder f into model prediction. The loss of downstream tasks can be formulated as:

$$\mathcal{L}_{sup} = \mathcal{L}_{sup}(d, f, \mathcal{G}; y), \tag{83}$$

where y is the ground truth in downstream tasks. We can obverse that $\mathcal{L}_{sup}$ is a typical supervised loss. To ensure the model achieves wise graph representation extraction and optimistic prediction performance, $\mathcal{L}_{ssl}$ and $\mathcal{L}_{sup}$ have to be minimized simultaneously. We introduce 3 different ways to minimize the two loss functions:

Pre-training. This strategy has two steps. In pre-training step, the $\mathcal{L}_{ssl}$ is minimized to get g^* and f^*:

$$g^*, f^* = \arg\min_{g,f} \mathcal{L}_{ssl}(g, f, \mathcal{D}). \tag{84}$$

Then the parameter of f^* is kept to continue training in pretext supervised learning progress. The supervised loss is minimized to get the final parameters of f and d.

$$\min_{d,f} \mathcal{L}_{ssl}(d, f|_{f_0=f}^*, \mathcal{G}; y). \tag{85}$$

Collaborative Train. In this strategy, $\mathcal{L}_{ssl}$ and $\mathcal{L}_{sup}$ are optimized simultaneously. A hyperparameter α is used to balance the contribution of pretext task loss and downstream task loss. The overall minimization strategy is like the traditional supervised strategy with a pretext task regularization:

$$\min_{g,f,d}[\mathcal{L}_{ssl}(g, f, \mathcal{G}) + \alpha\mathcal{L}_{sup}(d, f, \mathcal{G}; y)]. \tag{86}$$

Unsupervised Feature Extracting. This strategy is similar to the Pre-training and Fine-tuning strategy in the first step to minimize pretext task loss $\mathcal{L}_{ssl}$ and get f^*. However, when minimizing downstream loss $\mathcal{L}_{sup}$, the encoder f^* is fixed. Also, the training graph data are on the same dataset, which differs from the Pre-training and Fine-tuning strategy. The formulation is defined as:

$$g^*, f^* = \arg\min_{g,f} \mathcal{L}_{ssl}(g, f, \mathcal{D}), \tag{87}$$

$$\min_d \mathcal{L}_{sup}(d, f^*, \mathcal{G}; y). \tag{88}$$

8.2. Generation-based pretext task design

If a model with an encoder–decoder structure can reproduce certain graph features from an incomplete or perturbed graph, it indicates the encoder has the ability to extract useful graph representation. This motivation is derived from Autoencoder (Hinton & Salakhutdinov, 2006) which originally learns on image dataset. In such a case, Eq. (84) can be rewritten as:

$$\min_{g,f} \mathcal{L}_{ssl}(g(f(\hat{\mathcal{G}})), \mathcal{G}), \tag{89}$$

where $f(\cdot)$ and $g(\cdot)$ stand for the representation encoder and rebuilding decoder. However, feature information and structure information are both important compositions suitable to be rebuilt for graph datasets. So generation-based pretext can be divided into two categories: feature rebuilding and structure rebuilding. We introduce several outstanding models in the following part.

Graph Completion (You et al., 2020) is one of the representative methods of feature rebuilding. They mask some node features to generate an incomplete graph. Then the pretext task is set as predicting the removed node features. As shown in Eq. (90), this method can be formulated as a special case of Eq. (90), letting $\hat{\mathcal{G}} = (A, \hat{X})$ and replacing

Table 7
Summary of methods for self-supervised learning on graphs. "PT", "CT" and "UFE" mean "Pre-training", "Collaborative Train" and "Unsupervised Feature Extracting" respectively.

	Approach	Augmentation scheme	Training scheme	Generation target	Objective function
Generation-based	Graph Completion (You, Chen, Wang, & Shen, 2020)	Feature mask	PT/CT	Node feature	–
	AttributeMask (Jin et al., 2020)	Feature mask	PT/CT	PCA Node feature	–
	AttrMasking (Hu et al., 2019)	Feature mask	PT	Node/Edge feature	–
	MGAE (Wang, Pan, Long, Zhu, & Jiang, 2017)	No augmentation	CT	Node feature	–
	GAE (Kipf & Welling, 2016b)	Feature noise	UFE	Adjacency matrix	–
Contrast-based	DeepWalk (Perozzi et al., 2014)	Random walk	UFE	–	SkipGram
	LINE (Tang, Qu, Wang et al., 2015)	Random walk	UFE	–	Jensen–Shannon
	GCC (Qiu et al., 2020)	Random walk	PT/URL	–	InfoNCE
	SimGCL (Yu et al., 2022)	Embedding noise	UFE	–	InfoNCE
	SimGRACE (Xia, Wu, Chen, Hu and Li, 2022)	Model noise	UFE	–	InfoNCE
	GCA (Zhu et al., 2021)	Feature masking & Structure adjustment	URL	–	InfoNCE
	BGRL (Grill et al., 2020)	Feature masking & Structure adjustment	URL	–	BYOL

$\mathcal{G} \to X$. The loss function is often Mean Squared Error or Cross Entropy, depending on whether the feature is continuous or binary.

$$\min_{g,f} \mathbf{MSE}(g(f(\hat{\mathcal{G}})), \mathbf{X}). \qquad (90)$$

Other works make some changes to the feature settings. For example, AttrMasking (Hu et al., 2019) aims to rebuild both node representation and edge representation, AttributeMask (Jin, Derr et al., 2020) preprocess X firstly by PCA to reduce the complexity of rebuilding features.

On the other hand, MGAE (Wang et al., 2017) modifies the original graph by adding noise in node representation, motivated by denoising autoencoder (Vincent et al., 2010). As shown in Eq. (90), we can also consider MGAE as an implement of Eq. (84) where $\hat{\mathcal{G}} = (A, \hat{X})$ and $\mathcal{G} \to X$. $\hat{X}$ stands for perturbed node representation. Since the noise is independent and random, the encoder is more robust to feature input.

$$\min_{g,f} \mathbf{BCE}(g(f(\hat{\mathcal{G}})), \mathbf{A}). \qquad (91)$$

As for structure rebuilding methods, GAE (Kipf & Welling, 2016b) is the simplest instance, which can be regarded as an implement of Eq. (84) where $\hat{\mathcal{G}} = \mathcal{G}$ and $\mathcal{G} \to A$. A is the adjacency matrix of the graph. Similar to feature rebuilding methods, GAE compresses raw node representation vectors into low-dimensional embedding with its encoder. Then the adjacency matrix is rebuilt by computing node embedding similarity. The loss function is set to the error between the ground-truth adjacency matrix and the recovered one, to help the model rebuild the correct graph structure. Other feature rebuilding methods (Zang et al., 2023) and structure rebuilding methods (Tan et al., 2023; Wen et al., 2023) are also increasingly being developed across numerous related publications.

8.3. Contrast-based pretext task design

The mutual information maximization principle, which implements self-supervising by predicting the similarity between the two augmented views, forms the foundation of contrast-based approaches. Since mutual information represents the degree of correlation between two samples, we can maximize it in augmented pairs and minimize it in random-selected pairs.

The contrast-based graph SSL taxonomy can be formulated as Eq. (92). The discriminator that calculates the similarity of sample pairs is indicated by pretext decoder g. $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ are two variants of G that have been augmented. Since graph contrastive learning methods differ from each other in (1) view generation, (2) MI estimation method we introduce this methodology in these two perspectives.

$$\min_{g,f} \mathcal{L}_{ssl}(g[f(\hat{\mathcal{G}}^{(1)}), f(\hat{\mathcal{G}}^{(2)})]). \qquad (92)$$

The domain of contrastive-based graph SSL is witnessing an expanding body of work in a growing number of methods (Ho & Armanfard, 2023; Ji et al., 2023; Yang, Huang, Xia and Huang, 2023; Zhao, Wen, Ju, Zhang and Ye, 2023) and applications (Farhat & Ben-Hamadou, 2023; Gao, Wang, He, Feng, & Zhang, 2023; Xia et al., 2023).

8.3.1. View generation

The traditional pipeline of contrastive learning-based models first involves augmenting the graph using well-crafted empirical methods, and then maximizing the consistency between different augmentations. Drawing from methods in the computer vision domain and considering the non-Euclidean structure of graph data, typical graph augmentation methods aim to modify the graph topologically or representationally.

Given graph $\mathcal{G} = (A, X)$, the topologically augmentation methods usually modify the adjacency matrix A, which can be formulated as:

$$\hat{A} = \mathcal{T}_A(A), \qquad (93)$$

where $\mathcal{T}_A(\cdot)$ is the transform function of adjacency matrix. Topology augmentation methods have many variants, in which the most popular one is edge modification, given by $\mathcal{T}_A(A) = P \circ A + Q \circ (1 - A)$. P and Q are two matrices representing edge dropping and adding respectively. Another method, graph diffusion, connect nodes with their k-hop neighbors with specific weight, defined as: $\mathcal{T}_A(A) = \sum_{k=0}^{\infty} \alpha_k T^k$. where α and T are coefficient and transition matrix. Graph diffusion method can integrate broad topological information with local structure.

On the other hand, the representative augmentation modifies the node representation directly, which can be formulated as:

$$\hat{X} = \mathcal{T}_X(X), \qquad (94)$$

usually $\mathcal{T}_X(\cdot)$ can be a simple masking operator, a.k.a. $\mathcal{T}_X(X) = M \circ X$ and $M \in \{0,1\}^{N \times D}$. Based on such mask strategy, some methods propose ways to improve performance. GCA (Zhu et al., 2021) preserves critical nodes while giving less significant nodes a larger masking probability, where significance is determined by node centrality.

As introduced before, the paradigm of augmentation has been proven to be effective in contrastive learning view generation. However, given the variety of graph data, it is challenging to maintain semantics properly during augmentations. To preserve the valuable nature of specific graph datasets, There are currently three mainly used methods: picking by trial-and-errors, trying laborious search, or seeking domain-specific information as guidance (Ju et al., 2023; Luo et al., 2023, 2022). Such complicated augmentation methods constrain the effectiveness and widespread application of graph contrastive learning. So many newest works question the necessity of augmentation and seek other contrastive view generation methods.

SimGCL (Yu et al., 2022) is one of the outstanding works challenging the effectiveness of graph augmentation. The author finds that

noise can be a substitution to augmentation to produce graph views in specific tasks such as recommendation. After doing an ablation study about augmentation and InfoNCE (Xie, Xu, Zhang, Wang and Ji, 2022), they find that the InfoNCE loss, not the augmentation of the graph, is what makes the difference. It can be further explained by the importance of distribution uniformity. Contrastive learning enhances model representation ability by intensifying two characteristics: The alignment of features from positive samples and the uniformity of the normalized feature distribution. SimGCL directly adds random noises to node embeddings as augmentation, to control the uniformity of the representation distribution more effectively:

$$\mathbf{e}_i^{(1)} = \mathbf{e}_i + \epsilon^{(1)} * \tau_i^{(1)}, \quad \mathbf{e}_i^{(2)} = \mathbf{e}_i + \epsilon^{(2)} * \tau_i^{(2)},$$
$$\epsilon \sim \mathcal{N}(0, \sigma^2), \tag{95}$$

where $\mathbf{e}_i$ is a node representation in embedding space, $\tau_i^{(1)}$ and $\tau_i^{(2)}$ are two random sampled unit vector. The experiment results indicate that SimGCL performs better than its graph augmentation-based competitors, while training time is significantly decreased.

SimGRACE (Xia, Wu et al., 2022) is another graph contrastive learning framework without data augmentation. Motivated by the observation that despite encoder disruption, graph data can effectively maintain their semantics, SimGRACE takes GNN with its modified version as an encoder to produce two contrastive embedding views by the same graph input. For GNN encoder $f(\cdot; \theta)$, the two contrastive embedding views $\mathbf{e}, \mathbf{e}'$ can be computed by:

$$\mathbf{e}^{(1)} = f(\mathcal{G}; \theta), \quad \mathbf{e}^{(2)} = f(\mathcal{G}; \theta + \epsilon \cdot \Delta\theta),$$
$$\Delta\theta_l \sim \mathcal{N}(0, \sigma_l^2), \tag{96}$$

where $\Delta\theta_l$ represents GNN parameter perturbation $\Delta\theta$ in the lth layer. SimGRACE can improve alignment and uniformity simultaneously, proving its capacity to produce high-quality embedding.

8.3.2. MI estimation method.

The mutual information $I(x, y)$ measures the information that x and y share, given a pair of random variables (x, y). As discussed before, mutual information is a significant component of the contrast-based method by formulating the loss function. Mathematically rigorous MI is defined on the probability space, we can formulate mutual information between a pair of instances (x_i, x_j) as:

$$I(x, y) = D_{KL}(p(x, y) \| p(x)p(y))$$
$$= E_{p(x,y)}[\log \frac{p(x, y)}{p(x)p(y)}]. \tag{97}$$

However, directly computing Eq. (97) is quite difficult, so we introduce several different types of estimation for MI:

InfoNCE. Noise-contrastive estimator is a widely used lower bound MI estimator. Given a positive sample y and several negative sample y_i', a noise-contrastive estimator can be formulated as (Qiu, Chen et al., 2020; Zhu et al., 2020):

$$\mathcal{L} = -I(x, y) = -E_{p(x,y)}[\log \frac{e^{g(x,y)}}{e^{g(x,y)} + \sum_i e^{g(x,y_i')}}], \tag{98}$$

usually the kernel function $g(\cdot)$ can be cosine similarity or dot product.

Triplet Loss. Intuitively, we can aim to create a distinct separation in the degree of similarity, ensuring that positive samples are closer together and negative samples are further apart by a certain distance. So we can define the loss function in the following manner (Jiao et al., 2020):

$$\mathcal{L} = E_{p(x,y)}[\max(g(x, y) - g(x, y') + \epsilon, 0)], \tag{99}$$

where ϵ is a hyperparameter. This function is straightforward to compute.

BYOL Loss. Estimation without negative samples is investigated by BYOL (Grill et al., 2020). The estimator is Asymmetrically structured:

$$\mathcal{L} = E_{p(x,y)}[2 - 2 \frac{g(x) \cdot y}{\|g(x)\|\|y\|}], \tag{100}$$

note that encoder g should keep the dimension of input and output the same.

8.4. Summary

This section introduces graph self-supervised learning and we provide the summary as follows:

- **Techniques.** Differing from classic supervised and semi-supervised learning, self-supervised learning increases a model's generalization ability and robustness while decreasing reliance on labels. Graph SSL utilizes pretext tasks to extract inherent information from representation distributions. Typical Graph SSL methods can be divided into generation-based and contrast-based approaches. Generation-based methods learn an encoder with the ability to reconstruct a graph as precisely as possible, motivated by the principles of Autoencoder. Contrast-based methods, which have recently attracted significant interest, involve learning an encoder to minimize mutual information between relevant instances and maximize mutual information between unrelated instances.
- **Challenges and Limitations.** Although graph SSL has achieved superior performance in many tasks, its theoretical basis is not so solid. Many well-known methods are validated only through experiments, without providing theoretical explanations or mathematical proofs. It is imperative to establish a strong theoretical foundation for graph SSL.
- **Future Works.** In the future we expect more graph ssl methods designed essentially by theoretical proof, without dedicated designed augment process or pretext tasks by intuition. This will bring us more definite mathematical properties and a less ambiguous empirical sense. Also, graphs are a prevalent form of data representation across diverse domains, yet obtaining manual labels can be prohibitively expensive. Expanding the applications of graph SSL to broader fields is a promising avenue for future research.

9. Graph structure learning

Graph structure determines how node features propagate and affect each other, playing a crucial role in graph representation learning. In some scenarios the provided graph is incomplete, noisy, or even has no structure information at all. Recent research also finds that graph adversarial attacks (i.e., modifying a small number of node features or edges), can degrade learned representations significantly. These issues motivate graph structure learning (GSL), which aims to learn a new graph structure to produce optimal graph representations. According to how edge connectivity is modeled, there are three different approaches in GSL, namely metric-based approaches, model-based approaches, and direct approaches. Besides edge modeling, regularization is also a common trick to make the learned graph satisfy some desired properties. We first present the basic framework and regularization methods for GSL in Sections 9.1 and 9.2, respectively, and then introduce different categories of GSL in Sections 9.3, 9.4 and 9.5. We summarize GSL approaches in Table 8.

9.1. Overall framework

We denote a graph by $\mathcal{G} = (\mathbf{A}, \mathbf{X})$, where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix and $\mathbf{X} \in \mathbb{R}^{N \times M}$ is the node feature matrix with M being the dimension of each node feature. A graph encoder f_θ learns to represent the graph based on node features and graph structure for task-specific objective $\mathcal{L}_t(f_\theta(\mathbf{A}, \mathbf{X}))$. In the GSL setting, there is also a graph structure learner which aims to build a new graph adjacency matrix $\mathbf{A}^*$ to optimize the learned representation. Besides the task-specific objective,

Table 8
Summary of graph structure learning methods.

	Method	Structure learning	Regularization		
			Sparsity	Low-rank	Smoothness
Metric-based	AGCN (Li, Wang, Zhu and Huang, 2018)	Mahalanobis distance			
	GRCN (Yu, Zhang, Jiang, Wu and Yang, 2021)	Inner product	✓		
	CAGCN (Zhu, Xu, Yu, Wu and Wang, 2020)	Inner product	✓		
	GNNGUARD (Zhang & Zitnik, 2020)	Cosine similarity			
	IDGL (Chen, Wu and Zaki, 2020)	Cosine similarity	✓	✓	✓
	HGSL (Zhao et al., 2021)	Cosine similarity	✓		
	GDC (Gasteiger, Weißenberger, & Günnemann, 2019)	Graph diffusion	✓		
Model-based	GLN (Pilco & Rivera, 2019)	Recurrent blocks			
	GLCN (Jiang, Zhang, Lin, Tang and Luo, 2019)	One-layer neural network	✓		✓
	NeuralSparse (Zheng et al., 2020)	Multi-layer neural network	✓		
	GAT (Veličković et al., 2017)	Self-attention			
	GaAN (Zhang et al., 2018)	Gated attention			
	hGAO (Gao & Ji, 2019a)	Hard attention	✓		
	VIB-GSL (Sun et al., 2022)	Dot-product attention	✓		
	MAGNA (Wang, Ying, Huang and Leskovec, 2020)	Graph attention diffusion			
Direct	GLNN (Gao, Hu, & Guo, 2020)	MAP estimation	✓		✓
	Pro-GNN (Jin et al., 2020)	Direct optimization	✓	✓	✓
	GSML (Wan & Kokel, 2021)	Bilevel optimization	✓		
	LSD-GNN (Franceschi, Niepert, Pontil, & He, 2019)	Bilevel optimization			
	BGCNN (Zhang, Pal, Coates and Ustebay, 2019)	Bayesian optimization			
	VGCN (Elinas, Bonilla, & Tiao, 2020)	Stochastic variational inference			

a regularization term can be added to constrain the learned structure. So the overall objective function of GSL can be formulated as

$$\min_{\theta, \mathbf{A}^*} \mathcal{L} = \mathcal{L}_t(f_\theta(\mathbf{A}^*, \mathbf{X})) + \lambda \mathcal{L}_r(\mathbf{A}^*, \mathbf{A}, \mathbf{X}), \qquad (101)$$

where $\mathcal{L}_t$ is the task-specific objective, $\mathcal{L}_r$ is the regularization term and λ is a hyperparameter for the weight of regularization.

9.2. Regularization

The goal of regularization is to constrain the learned graph to satisfy some properties by adding some penalties to the learned structure. The most common properties used in GSL are sparsity, low lank, and smoothness.

9.2.1. Sparsity
Noise or adversarial attacks will introduce redundant edges into graphs and degrade the quality of graph representation. An effective technique to remove unnecessary edges is sparsity regularization, i.e., adding a penalty on the number of nonzero entries of the adjacency matrix (ℓ_0-norm) (Wan & Kokel, 2021; Yu, Zhang et al., 2021; Zhao, Wang et al., 2021; Zheng et al., 2020):

$$\mathcal{L}_{sp} = \|\mathbf{A}\|_0, \qquad (102)$$

however, ℓ_0-norm is not differentiable so optimizing it is difficult, and in many cases ℓ_1-norm is used instead as a convex relaxation. Other methods to impose sparsity include pruning and discretization (Franceschi et al., 2019; Zhu, Xu, Yu, Wu et al., 2020). These processes are also called postprocessing since they usually happen after the adjacency matrix is learned. Pruning removes part of the edges according to some criteria (Zhu, Xu, Yu, Wu et al., 2020). For example, edges with weights lower than a threshold, or those not in the top-K edges of nodes or graphs. Discretization is applied to generate graph structure by sampling from some distribution (Franceschi et al., 2019). Compared to directly learning edge weights, sampling enjoys the advantage of controlling the generated graph, but has issues during optimizing since sampling itself is discrete and hard to optimize. Reparameterization and Gumbel-softmax are two useful techniques to overcome such issues, and are widely adopted in GSL.

9.2.2. Low rank
In real-world graphs, similar nodes are likely to group together and form communities, which should lead to a low-rank adjacency matrix.

Recent work also finds that adversarial attacks tend to increase the rank of the adjacency matrix quickly (Chen, Wu et al., 2020; Jin, Ma et al., 2020). Therefore, low-rank regularization is also a useful tool to make graph representation learning more robust:

$$\mathcal{L}_{lr} = Rank(\mathbf{A}). \qquad (103)$$

It is hard to minimize matrix rank directly. A common technique is to optimize the nuclear norm, which is a convex envelope of the matrix rank:

$$\mathcal{L}_{nc} = \|\mathbf{A}\|_* = \sum_i^N \sigma_i, \qquad (104)$$

where σ_i are singular values of $\mathbf{A}$. Entezari et al. replaces the learned adjacency matrix with rank-r approximation by singular value decomposition (SVD) to achieve robust graph learning against adversarial attacks.

9.2.3. Smoothness
A common assumption is that connected nodes share similar features, or in other words, the graph is "smooth" as the difference between local neighbors is small (Chen, Wu et al., 2020; Gao et al., 2020; Jiang, Zhang et al., 2019; Jin, Ma et al., 2020). The following metric is a natural way to measure graph smoothness:

$$\mathcal{L}_{sm} = \frac{1}{2} \sum_{i,j=1}^N A_{ij}(x_i - x_j)^2 = tr(\mathbf{X}^\top (\mathbf{D} - \mathbf{A})\mathbf{X}) = tr(\mathbf{X}^\top \mathbf{L}\mathbf{X}), \qquad (105)$$

where $\mathbf{D}$ is the degree matrix of $\mathbf{A}$ and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is called graph Laplacian. A variant is to use the normalized graph Laplacian $\widehat{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}}$.

9.3. Metric-based methods

Metric-based methods measure the similarity between nodes as the edge weights. They follow the basic assumption that similar nodes tend to have connections with each other. We show some representative works

Adaptive Graph Convolutional Neural Networks (AGCN) (Li, Wang et al., 2018). AGCN learns a task-driven adaptive graph during training to enable a more generalized and flexible graph representation model. After parameterizing the distance metric between nodes, AGCN is able

to adapt graph topology to the given task. It proposes a generalized Mahalanobis distance between two nodes with the following formula:

$$\mathbb{D}(x_i, x_j) = \sqrt{(x_i - x_j)^{\top} M(x_i - x_j)}, \tag{106}$$

where $M = W_d W_d^{\top}$ and W_d is the trainable weights to minimize task-specific objective. Then the Gaussian kernel is used to obtain the adjacency matrix:

$$\mathbb{G}_{ij} = \exp(-\mathbb{D}(x_i, x_j)/(2\sigma^2)), \tag{107}$$

$$\hat{A} = normalize(\mathbb{G}). \tag{108}$$

Graph-Revised Convolutional Network (GRCN) (Yu, Zhang et al., 2021). GRCN uses a graph revision module to predict missing edges and revise edge weights through joint optimization on downstream tasks. It first learns the node embedding with GCN and then calculates pair-wise node similarity with the dot product as the kernel function.

$$Z = GCN_g(A, X), \tag{109}$$

$$S_{ij} = \langle z_i, z_j \rangle. \tag{110}$$

The revised adjacency matrix is the residual summation of the original adjacency matrix $\hat{A} = A + S$. GRCN also applies a sparsification technique on the similarity matrix S to reduce computation cost:

$$S_{ij}^{(K)} = \begin{cases} S_{ij}, & S_{ij} \in topK(S_i) \\ 0, & S_{ij} \notin topK(S_i) \end{cases}. \tag{111}$$

Threshold pruning is also a common strategy for sparsification. For example, CAGCN (Zhu, Xu, Yu, Wu et al., 2020) uses dot product to measure node similarity, and refines the graph structure by removing edges between nodes whose similarity is less than a threshold τ_r and adding edges between nodes whose similarity is greater than another threshold τ_a.

Defending Graph Neural Networks against Adversarial Attacks (GNN-Guard) (Zhang & Zitnik, 2020). GNNGuard measures similarity between a node u and its neighbor v in the kth layer by cosine similarity and normalizes node similarity at the node level within the neighborhood as follows:

$$s_{uv}^k = \frac{h_u^k \odot h_v^k}{\|h_u^k\|_2 \|h_v^k\|_2}, \tag{112}$$

$$\alpha_{uv}^k = \begin{cases} s_{uv}^k / \sum_{v \in \mathcal{N}_u} s_{uv}^k \times \hat{N}_u^k / (\hat{N}_u^k + 1), & if \quad u \neq v \\ 1/(\hat{N}_u^k + 1), & if \quad u = v \end{cases}, \tag{113}$$

where $\mathcal{N}_u$ denotes the neighborhood of node u and $\hat{N}_u^k = \sum_{v \in \mathcal{N}_u} \|s_{uv}^k\|_0$. To stabilize GNN training, it also proposes a layer-wise graph memory by keeping part of the information from the previous layer in the current layer. Similar to GNNGuard, IDGL (Chen, Wu et al., 2020) uses multi-head cosine similarity and mask edges with node similarity smaller than a non-negative threshold, and HGSL (Zhao, Wang et al., 2021) generalizes this idea to heterogeneous graphs.

Graph Diffusion Convolution (GDC) (Gasteiger et al., 2019). GDC replaces the original adjacency matrix with generalized graph diffusion matrix $\mathbf{S}$:

$$\mathbf{S} = \sum_{k=0}^{\infty} \theta_k \mathbf{T}^k, \tag{114}$$

where θ_k is the weighting coefficient and $\mathbf{T}$ is the generalized transition matrix. To ensure convergence, GDC further requires that $\sum_{k=0}^{\infty} \theta_k = 1$ and the eigenvalues of $\mathbf{T}$ lie in $[0, 1]$. The random walk transition matrix $\mathbf{T}_{rw} = \mathbf{A}\mathbf{D}^{-1}$ and the symmetric transition matrix $\mathbf{T}_{sym} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ are two examples. This new graph structure allows graph convolution to aggregate information from a larger neighborhood. The graph diffusion acts as a smoothing operator to filter out underlying noise. However, in most cases graph diffusion will result in a dense adjacency matrix S, so sparsification technology like top-k filtering

and threshold filtering will be applied to graph diffusion. Following GDC, there are some other graph diffusion proposed. For example, AdaCAD (Lim, Um, Chang, Jo, & Choi, 2021) proposes Class-Attentive Diffusion, which further considers node features and aggregates nodes probably of the same class among K-hop neighbors. Adaptive diffusion convolution (ADC) (Zhao, Dong, Ding, Kharlamov and Tang, 2021) learns the optimal neighborhood size via optimizing a bi-level problem.

9.4. Model-based methods

Model-based methods parameterize edge weights with more complex models like deep neural networks. Compared to metric-based methods, model-based methods offer greater flexibility and expressive power.

Graph Learning Network (GLN) (Pilco & Rivera, 2019). GLN proposes a recurrent block to first produce intermediate node embeddings and then merge them with adjacency information as the output of this layer to predict the adjacency matrix for the next layer. Specifically, it uses convolutional graph operations to extract node features, and creates a local-context embedding based on node features and the current adjacency matrix:

$$H_{int}^{(l)} = \sum_{i=1}^k \sigma_l(\tau(A^{(l)})H^{(l)}W_i^{(l)}), \tag{115}$$

$$H_{local}^{(l)} = \sigma_l(\tau(A^{(l)})H_{int}^{(l)}U^{(l)}), \tag{116}$$

where $W_i^{(l)}$ and $U^{(l)}$ are the learnable weights. GLN then predicts the next adjacency matrix as follows:

$$A^{(l+1)} = \sigma_l(M^{(l)}\alpha_l(H_{local}^{(l)})M^{(l)\top}). \tag{117}$$

Similarly, GLCN (Jiang, Zhang et al., 2019) models graph structure with a softmax layer over the inner product between the difference of node features and a learnable vector. NeuralSparse (Zheng et al., 2020) uses a multi-layer neural network to generate a learnable distribution from which a sparse graph structure is sampled. PTDNet (Luo et al., 2021) prunes graph edges with a multi-layer neural network and penalizes the number of non-zero elements to encourage sparsity.

Graph Attention Networks (GAT) (Veličković et al., 2017). Besides constructing a new graph to guide the message passing and aggregation process of GNNs, many recent researchers also leverage the attention mechanism to adaptively model the relationship between nodes. GAT is the first work to introduce the self-attention strategy into graph learning. In each attention layer, the attention weight between two nodes is calculated as the Softmax output on the combination of linear and non-linear transform of node features:

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j), \tag{118}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp(e_{ik})}, \tag{119}$$

where $\mathcal{N}_i$ denotes the neighborhood of node i, $\mathbf{W}$ is learnable linear transform and a is pre-defined attention function. In the original implementation of GAT, a is a single-layer neural network with LeakyReLU:

$$a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j) = \text{LeakyReLU}(\vec{a}^{\top}[\mathbf{W}\vec{h}_i \parallel \mathbf{W}\vec{h}_j]). \tag{120}$$

The attention weights are then used to guide the message-passing phase of GNNs:

$$\vec{h}_i' = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij}\mathbf{W}\vec{h}_j), \tag{121}$$

where σ is a nonlinear function. It is beneficial to concatenate multiple heads of attention together to get a more stable and generalizable model, so-called multi-head attention. The attention mechanism serves as a soft graph structure learner which captures important connections

within node neighborhoods. Following GAT, many recent works propose more effective and efficient graph attention operators to improve performance. GaAN (Zhang, Shi et al., 2018) adds a soft gate at each attention head to adjust its importance. MAGNA (Wang, Ying et al., 2020) proposes a novel graph attention diffusion layer to incorporate multihop information. One drawback of graph attention is that the time and space complexities are both $O(N^3)$. hGAO (Gao & Ji, 2019a) performs hard graph attention by limiting node attention to its neighborhood. VIB-GSL (Sun et al., 2022) adopts the information bottleneck principle to guide feature masking in order to drop task-irrelevant information and preserve actionable information for the downstream task.

9.5. Direct methods

Direct methods treat edge weights as free learnable parameters. These methods enjoy more flexibility but are also more difficult to train. The optimization is usually carried out in an alternating way, i.e., iteratively updating the adjacency matrix $\mathbf{A}$ and the GNN encoder parameters θ.

GLNN (Gao et al., 2020). GLNN uses MAP estimation to learn an optimal adjacency matrix for a joint objective function including sparsity and smoothness. Specifically, it targets at finding the most probable adjacency matrix $\hat{A}$ given graph node features x:

$$\tilde{A}_{MAP}(x) = \underset{\hat{A}}{\mathrm{argmax}}\, f(x|\hat{A})g(\hat{A}), \tag{122}$$

where $f(x|\hat{A})$ measures the likelihood of observing x given $\hat{A}$, and $g(\hat{A})$ is the prior distribution of $\hat{A}$. GLNN uses sparsity and property constraint as prior, and defines the likelihood function f as:

$$f(x|\hat{A}) = exp(-\lambda_0 x^{\top} L x) \tag{123}$$
$$= exp(-\lambda_0 x^{\top}(I - \hat{A})x), \tag{124}$$

where λ_0 is a parameter. This likelihood imposed a smoothness assumption on the learned graph structure. Some other works also model the adjacency matrix in a probabilistic manner. Bayesian GCNN (Zhang, Pal et al., 2019) adopts a Bayesian framework and treats the observed graph as a realization from a family of random graphs. Then it estimates the posterior probability of labels given the observed graph adjacency matrix and features with Monte Carlo approximation. VGCN (Elinas et al., 2020) follows a similar formulation and estimates the graph posterior through stochastic variational inference. Pro-GNN (Jin, Ma et al., 2020) learns a clean graph structure from perturbed data and optimizes parameters for a robust GNN, leveraging properties like sparsity, low rank, and feature smoothness.

Graph Sparsification via Meta-Learning (GSML) (Wan & Kokel, 2021). GSML formulates GSL as a meta-learning problem and uses bi-level optimization to find the optimal graph structure. The goal is to find a sparse graph structure that leads to high node classification accuracy at the same time given labeled and unlabeled nodes. To achieve this, GSML makes the inner optimization as training on the node classification task, and targets the outer optimization at the sparsity of the graph structure, which formulates the following bi-level optimization problem:

$$\hat{G}^* = \underset{\hat{G} \in \Phi(G)}{\min}\, L_{sps}(f_{\theta^*}(\hat{G}), Y_U), \tag{125}$$

$$s.t. \quad \theta^* = \underset{\theta}{\mathrm{argmin}}\, L_{train}(f_{\theta}(\hat{G}), Y_L). \tag{126}$$

In this bi-level optimization problem, $\hat{G} \in \Phi(G)$ are the meta-parameters and optimized directly without parameterization. Similarly, LSD-GNN (Franceschi et al., 2019) also uses bi-level optimization. It models graph structure with a probability distribution over the graph and reformulates the bi-level program in terms of the continuous distribution parameters.

9.6. Summary

In this section, we provide the summary as follows:

- **Techniques.** GSL aims to learn an optimized graph structure for better graph representations. It is also used for more robust graph representation against adversarial attacks. According to the way of edge modeling, we categorize GSL into three groups: metric-based methods, model-based methods, and direct methods. Regularization is also a commonly used principle to make the learned graph structure satisfy specific properties including sparsity, low-rank and smoothness.
- **Challenges and Limitations.** Since there is no way to access the ground truth or optimal graph structure as training data, the learning objective of GSL is either indirect (e.g., performance on downstream tasks) or manually designed (e.g., sparsity and smoothness). Therefore, the optimization of GSL is difficult and the performance is not satisfying. In addition, many GSL methods are based on homophily assumption, i.e., similar nodes are more likely to connect with each other. However, many other types of connection exist in the real world which impose great challenges for GSL.
- **Future Works.** In the future we expect more efficient and generalizable GSL methods to be applied to large-scale and heterogeneous graphs. Most existing GSL methods focus on pair-wise node similarities and thus struggle to scale to large graphs. Besides, they often learn homogeneous graph structure, but in many scenarios graphs are heterogeneous.

10. Social analysis

In the real world, there usually exist complex relations and interactions between people and multiple entities. Taking people, concrete things, and abstract concepts in society as nodes and taking the diverse, changeable, and large-scale connections between data as links, we can form massive and complex social information as social networks (Camacho, Panizo-LLedot, Bello-Orgaz, Gonzalez-Pardo, & Cambria, 2020; Tabassum, Pereira, Fernandes, & Gama, 2018). Compared with traditional data structures such as texts and forms, modeling social data as graphs has many benefits. Especially with the arrival of the "big data" era, more and more heterogeneous information is interconnected and integrated, and it is difficult and uneconomical to model this information with a traditional data structure. The graph is an effective implementation for information integration, as it can naturally incorporate different types of objects and their interactions from heterogeneous data sources (Moscato & Sperlì, 2021; Shi, Li, Zhang, Sun, & Philip, 2016). A summarization of social analysis applications is provided in Table 9.

10.1. Concepts of social networks

A social network is usually composed of multiple types of nodes, link relationships, and node attributes, which inherently include rich structural and semantic information. Specifically, a social network can be homogeneous or heterogeneous and directed or undirected in different scenarios. Without loss of generality, we define the social network as a directed heterogeneous graph $G = \{V, E, \mathcal{T}, \mathcal{R}\}$, where $V = \{n_i\}_{i=1}^{|V|}$ is the node set, $E = \{e_i\}_{i=1}^{|E|}$ is the edge set, $\mathcal{T} = \{t_i\}_{i=1}^{|\mathcal{T}|}$ is the node type set, and $\mathcal{R} = \{r_i\}_{i=1}^{|\mathcal{R}|}$ is the edge type set. Each node $n_i \in V$ is associated with a node type mapping: $\phi_n(n_i) = t_j : V \longrightarrow \mathcal{T}$ and each edge $e_i \in E$ is associated with a node type mapping: $\phi_e(e_i) = r_j : E \longrightarrow \mathcal{R}$. A node n_i may have a feature set, where the feature space is specific for the node type. An edge e_i is also represented by node pairs (n_j, n_k) at both ends and can be directed or undirected with relation-type-specific attributes. If $|\mathcal{T}| = 1$ and $|\mathcal{R}| = 1$, the social network is a homogeneous graph; otherwise, it is a heterogeneous graph.

Table 9
A summarization of social analysis applications.

Social networks	Node type	Edge type	Applications	References
Academic social network	*Author, Publication, Venue, Organization, Keyword*	*Authorship, Co-Author, Advisor-advisee, Citing, Cited, Co-Citing, Publishing*	Classification/ Clustering	Paper/author classification (Dong, Chawla, & Swami, 2017; Qiao et al., 2020; Wang et al., 2019; Zhang, Song, Huang, Swami and Chawla, 2019), name disambiguation (Chen et al., 2020; Ma & Xia, 2023; Qiao, Du, Fu, Wang, & Zhou, 2019; Zhang, Zhang, Yao and Tang, 2018)
			Relationship prediction	Co-authorship (Cho & Yu, 2018; Chuan et al., 2018; Zhu, Quan, Chen, Kim, & Che, 2023), citation relationship (Jiang, Yin, Gao, Lu, & Liu, 2018; Wang, Zhu, Dai and Wang, 2020; Yu, Gu, Zhou, & Han, 2012), advisor-advisee relationship (Liu et al., 2019; Luo et al., 2023; Zhao et al., 2018)
			Recommendation	Collaborator recommendation (Kong et al., 2017, 2016; Liu, Xie and Chen, 2018), paper recommendation (Bai et al., 2019; Dai et al., 2023; Sugiyama & Kan, 2010), venue recommendation (Margaris, Vassilakis, & Spiliotopoulos, 2019; Yu et al., 2018)
Social media network	*User, Blog, Article, Image, Video*	*Following, Like, Unlike, Clicked, Viewed, Commented, Reposted*	Anomaly detection	Malicious attacks (Liu et al., 2018; Sadhasivam, Valarmathie, & Dinakaran, 2023; Sun, Wang, Yang and Liu, 2020), emergency detection (Bian et al., 2020; Dahou, Mabrouk, Ewees, Gaheen, & Abd Elaziz, 2023; Li, Li, Zhang, Zhao and Wang, 2023), and robot discovery (Feng, Wan, Wang, & Luo, 2021; Lu, Gong, Li, Liu, & Liu, 2023)
			Sentiment analysis	Customer feedback (Rosa, Schwartz, Ruggiero, & Rodríguez, 2018; Uma Maheswari & Dhenakaran, 2023; Zhang et al., 2014), public events (Bouadjenek, Sanner, & Wu, 2023; Manguri, Ramadhan, & Amin, 2020; Unankard, Li, Sharaf, Zhong, & Li, 2014)
			Influence analysis	Important node finding (Domingos & Richardson, 2001; Richardson & Domingos, 2002), information diffusion modeling (Keikha, Rahgozar, Asadpour, & Abdollahi, 2020; Kumar, Mallik, Khetarpal, & Panda, 2022; Panagopoulos, Malliaros, & Vazirgiannis, 2020; Zhang, Li, Wei, Liu and Li, 2022)
Location-based social network	*Restaurant, Cinema, Mall, Parking*	*Friendship, Check-in*	POI recommendation	Spatial/temporal influence (Si, Zhang, & Liu, 2019; Wang et al., 2022; Zhao et al., 2020), social relationship (Long, Chen, Nguyen, & Yin, 2023; Xu, Ding and Zhao, 2021), textual information (Wang, Fukumoto, Li, Yu and Sun, 2023; Wang, Zeng, Wen, Gao and Zhou, 2023; Xu, Liu and Mei, 2021)
			Urban computing	Traffic congestion prediction (Jiang & Luo, 2022; Xiong, Vahedian, Zhou, Li, & Luo, 2018), urban mobility analysis (Cao et al., 2021; Yildirimoglu & Kim, 2018), event detection (Sofuoglu & Aviyente, 2022; Yu et al., 2021)

Almost any data produced by social activities can be modeled as social networks, for example, the academic social network produced by academic activities such as collaboration and citation, the online social network produced by user following and followed on social media, and the location-based social network produced by human activities on different locations. Based on constructing social networks, researchers have new paths to data mining, knowledge discovery, and multiple application tasks on social data. Exploring social networks also brings new challenges. One of the critical challenges is how to succinctly represent the network from the massive and heterogeneous raw graph data, that is, how to learn continuous and low-dimensional social network representations, so as to researchers can efficiently perform advanced machine learning techniques on the social network data for multiple application tasks, such as analysis, clustering, prediction, and knowledge discovery. Thus, graph representation learning on the social network becomes the foundational technique for social analysis.

10.2. Academic social network

Academic collaboration is a common and important behavior in academic society, and also a major way for scientists and researchers to innovate and breakthrough scientific research, which leads to social relationship between scholars. The academic data generated by academic collaboration usually contains a large number of interconnected entities with complex relationships (Kong, Shi, Yu, Liu, & Xia, 2019; Zhou, Wang, Qiao, Xiao and Du, 2020). Normally, in an academic social network, the node type set consists of *Author, Publication, Venue, Organization, Keyword*, etc., and the relation set consists of *Authorship,*

Co-Author, Advisor-advisee, Citing, Cited, Co-Citing, Publishing, Co-Word, etc. Note that in most social networks, each relation type always connects two fixed node types with a fixed direction. For example, the relation *Authorship* points from the node type *Author* to *Publication*, and the *Co-Author* is an undirected relation between two nodes with type *Author*. Based on the node and relation types in an academic social network, one can divide it into multiple categories. For example, the co-author network with nodes of *Author* and relations of *Co-Author*, the citation network with nodes of *Publication* and relation of *Citing*, and the academic heterogeneous information graph with multiple academic node and relation types. Many research institutes and academic search engines, such as Aminer,[1] DBLP,[2] Microsoft Academic Graph (MAG),[3] have provided open academic social network datasets for research purposes.

There are multiple applications of graph representation learning on the academic social network. Roughly, they can be divided into three categories–academic entity classification/clustering, academic relationship prediction, and academic resource recommendation.

- Academic entities usually belong to different classes of research areas. Research of academic entity classification and clustering aims to categorize these entities, such as papers and authors,

---

[1] https://www.aminer.cn/.

[2] https://dblp.uni-trier.de/.

[3] https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/.

into different classes (Dong et al., 2017; Ju et al., 2023; Qiao, Wang et al., 2020; Wang, Ji et al., 2019; Yi et al., 2023; Zhang, Song et al., 2019). In literature, academic networks such as Cora, citepSeer, and Pubmed (Sen et al., 2008) have become the most widely used benchmark datasets for examining the performance of graph representation learning models on paper classification. Also, the author name disambiguation problem (Chen, Zhang et al., 2020; Ma & Xia, 2023; Qiao et al., 2019; Zhang, Zhang et al., 2018) is also essentially a node clustering task on co-author networks and is usually solved by the graph representation learning technique.

- Academic relationship prediction represents the link prediction task on various academic relations. Typical applications are co-authorship prediction (Cho & Yu, 2018; Chuan et al., 2018; Zhu et al., 2023) and citation relationship prediction (Jiang et al., 2018; Wang, Zhu et al., 2020; Yu et al., 2012). Existing methods learn representations of authors and papers and use the similarity between two nodes to predict the link probability. Besides, some work (Liu et al., 2019; Luo, Shi et al., 2023; Zhao et al., 2018) studies the problem of advisor-advisee relationship prediction in the collaboration network.

- Various academic recommendation systems have been introduced to retrieve academic resources for users from large amounts of academic data in recent years. For example, collaborator recommendation (Kong et al., 2017, 2016; Liu, Xie et al., 2018) benefit researchers by finding suitable collaborators under particular topics; paper recommendation (Bai et al., 2019; Dai et al., 2023; Sugiyama & Kan, 2010) help researchers find relevant papers on given topics; venue recommendation (Margaris et al., 2019; Yu et al., 2018) help researchers choose appropriate venues when they submit papers.

10.3. Social media network

With the development of the Internet in decades, various online social media have emerged in large numbers and greatly changed people's traditional social models. People can establish friendships with others beyond the distance limit and share interests, hobbies, status, activities, and other information among friends. These abundant interactions on the Internet form large-scale complex social media networks, also named online social networks. Usually, in an academic social network, the node type set consists of *User, Blog, Article, Image, Video*, etc., and the relation type set consists of *Following, Like, Unlike, Clicked, Viewed, Commented, Reposted*, etc. The main property of a social media network is that it usually contains multi-mode information on the nodes, such as video, image, and text. Also, the relations are more complex and multiplex, including the explicit relations such as *Like* and *Unlike* and the implicit relations such as *Clicked*. The social media network can be categorized into multiple types based on their media categories. For example, the friendship network, the movie review network, and the music interacting network are extracted from different social media platforms. In a broad sense, the user–item networks in online shopping system can also be viewed as social media networks as they also exist on the Internet and contains rich interactions by people. There are many widely used data sources for social media network analysis, such as Twitter, Facebook, Weibo, YouTube, and Instagram.

The mainstream application research on social media networks via graph representation learning techniques mainly includes anomaly detection, sentiment analysis, and influence analysis.

- Anomaly detection aims to find strange or unusual patterns in social networks, which has a wide range of application scenarios, such as malicious attacks (Liu, Chen et al., 2018; Sadhasivam et al., 2023; Sun, Wang et al., 2020), emergency detection (Bian et al., 2020; Dahou et al., 2023), and robot discovery (Feng et al., 2021; Lu et al., 2023) in social networks. Unsupervised

anomaly detection usually learns a reconstructed graph to detect those nodes with higher reconstructed error as the anomaly nodes (Ahmed, Galoppo, Hu, & Ding, 2021; Zhao, Sawlani, Srinivasan and Akoglu, 2022); Supervised methods model the problem as a binary classification task on the learned graph representations (Meng et al., 2021; Zheng, Li, Li, Li, & Gao, 2019).

- Sentiment analysis, also named as opinion mining, is to mine the sentiment, opinions, and attitudes, which can help enterprises understand customer feedback on products (Rosa et al., 2018; Uma Maheswari & Dhenakaran, 2023; Zhang et al., 2014) and help the government analyze the public emotion and make rapid response to public events (Bouadjenek et al., 2023; Manguri et al., 2020; Unankard et al., 2014). The graph representation learning model is usually combined with RNN-based (Chen, Tian and Song, 2020; Zhang, Li and Song, 2019) or Transformer-based (AlBadani, Shi, Dong, Al-Sabri, & Moctard, 2022; Tang, Ji, Li and Zhou, 2020) text encoders to incorporate both the user relationship and textual semantic information.

- Influence analysis usually aims to find several nodes in a social network to initially spread information such as advertisements, so as to maximize the final spread of information (Domingos & Richardson, 2001; Richardson & Domingos, 2002). The core challenge is to model the information diffusion process in the social network. Deep learning methods (Keikha et al., 2020; Kumar et al., 2022; Panagopoulos et al., 2020; Zhang, Li et al., 2022) usually leverage graph neural networks to learn node embeddings and diffusion probabilities between nodes.

10.4. Location-based social network

Locations are the fundamental information of human social activities. With the wide availability of mobile Internet and GPS positioning technology, people can easily acquire their precise locations and socialize with their friends by sharing their historical check-ins on the Internet. This opens up a new avenue of research on location-based social network analysis, which gathered significant attention from the user, business, and government perspectives. Usually, in a location-based social network, the node type set consists of *User*, and *Location*, also named *Point of Interest* (POI) in the recommendation scenario containing multiple categories such as *Restaurant, Cinema, Mall, Parking*, etc. The relation type set consists of *Friendship, Check-in*. Also, those node and relation types that exist in traditional social media networks can be included in a location-based social network. The difference with other social networks, the main location-based social networks are spatial and temporal, making the graph representation learning more challenging. For example, in a typical social network constructed for the POI recommendation, the user nodes are connected with each other by their friendship. The location nodes are connected by user nodes with the relations feature of timestamps. The location nodes also have a spatial relationship with each other and own have complex features, including categories, tags, check-in counts, number of users check-in, etc. There are many location-based social network datasets, such as Foursquare,[4] Gowalla,[5] and Waze.[6] Also, many social media such as Twitter, Instagram, and Facebook can provide location information.

The research of graph representation learning on location-based social networks can be divided into two categories: POI recommendation for business benefits and urban computing for public management.

- POI recommendation is one of the research hotspots in the field of location-based social networks and recommendation systems in recent years (Islam, Mohammad, Das, & Ali, 2022; Ju et al., 2022;

---

[4] https://foursquare.com/.
[5] https://www.gowalla.com/.
[6] https://www.waze.com/live-map/.

Werneck et al., 2020), which aim to utilize historical check-ins of users and auxiliary information to recommend potential favor places for users from a large of location points. Existing researches mainly integrate four essential characteristics, including spatial influence, temporal influence (Si et al., 2019; Wang, Zhu et al., 2022; Zhao, Luo et al., 2020), social relationship (Long et al., 2023; Xu, Ding et al., 2021), and textual information (Wang, Fukumoto et al., 2023; Wang, Zeng et al., 2023; Xu, Liu et al., 2021).

- Urban computing is defined as a process of analysis of the large-scale connected urban data created from city activities of vehicles, human beings, and sensors (Paulos, Anderson, & Townsend, 2004; Paulos & Goodman, 2004; Silva et al., 2019). Besides the local-based social network, the urban data also includes physical sensors, city infrastructure, traffic roads, and so on. Urban computing aims to improve the quality of public management and life quality of people living in city environments. Typical applications including traffic congestion prediction (Jiang & Luo, 2022; Xiong et al., 2018), urban mobility analysis (Cao, Li, Tu et al., 2021; Yildirimoglu & Kim, 2018), event detection (Sofuoglu & Aviyente, 2022; Yu, Du et al., 2021).

10.5. Summary

This section introduces social analysis by graph representation learning and we provide the summary as follows:

- **Techniques.** Social networks, generated by human social activities, such as communication, collaboration, and social interactions, typically involve massive and heterogeneous data, with different types of attributes and properties that can change over time. Thus, social network analysis is a field of study that explores the techniques to understand and analyze the complex attributes, heterogeneous structures, and dynamic information of social networks. Social network analysis typically learns low-dimensional graph representations that capture the essential properties and patterns of the social network data, which can be used for various downstream tasks, such as classification, clustering, link prediction, and recommendation.
- **Challenges and Limitations.** Despite the structural heterogeneity in social networks (nodes and relations have different types), with the technological advances in social media, the node attributes have become more heterogeneous now, containing text, video, and images. Also, the large-scale problem is a pending issue in social network analysis. The data in the social network has increased exponentially in past decades, containing a high density of topological links and a large amount of node attribute information, which brings new challenges to the efficiency and effectiveness of traditional network representation learning on the social network. Lastly, social networks are often dynamic, which means the network information usually changes over time, and this temporal information plays a significant role in many downstream tasks, such as recommendations. This brings new challenges to representation learning on social networks in incorporating temporal information.
- **Future Works.** Recently, multi-modal big pre-training models that can fuse information from different modalities have gained increasing attention (Qiao et al., 2022; Radford et al., 2021). These models can obtain valuable information from a large amount of unlabeled data and transfer it to various downstream analysis tasks. Moreover, Transformer-based models have demonstrated better effectiveness than RNNs in capturing temporal information. In the future, there is potential for introducing multi-modal big pre-training models in social network analysis. Also, it is important to make the models more efficient for network information extraction and use lightweight techniques

like knowledge distillation to further enhance the applicability of the models. These advancements can lead to more effective social network analysis and enable the development of more sophisticated applications in various domains.

11. Molecular property prediction

Molecular Property Prediction is an essential task in computational drug discovery and cheminformatics. Traditional quantitative structure property/activity relationship (QSPR/QSAR) approaches are based on either SMILES or fingerprints (Mikolov, Sutskever, Chen, Corrado and Dean, 2013; Xu, Wang, Zhu, & Huang, 2017; Zhang et al., 2018), largely overlooking the topological features of the molecules. To address this problem, graph representation learning has been widely applied to molecular property prediction. A molecule can be represented as a graph where nodes stand for atoms and edges stand for atom-bonds (ABs). Graph-level molecular representations are learned via the message passing mechanism to incorporate the topological information. The representations are then utilized for the molecular property prediction tasks.

Specifically, a molecule is denoted as a topological graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i | i = 1, \ldots, |\mathcal{G}|\}$ is the set of nodes representing atoms. A feature vector $\mathbf{x}_i$ is associated with each node v_i indicating its type such as Carbon, Nitrogen. $\mathcal{E} = \{e_{ij} | i, j = 1, \ldots, |\mathcal{G}|\}$ is the set of edges connecting two nodes (atoms) v_i and v_j representing atom bonds. Graph representation learning methods are used to obtain the molecular representation $\mathbf{h}_{\mathcal{G}}$. Then downstream classification or regression layers $f(\cdot)$ are applied to predict the probability of target property of each molecule $y = f(\mathbf{h}_{\mathcal{G}})$.

In Section 11.1, we introduce 4 types of molecular properties graph representation learning can be treated and their corresponding datasets. Section 11.2 reviews the graph representation learning backbones applied to molecular property prediction. Strategies for training the molecular property prediction methods are listed in Section 11.3.

11.1. Molecular property categorization

Plenty of molecular properties can be predicted by graph-based methods. We follow Wieder et al. (2020) to categorize them into 4 types: quantum chemistry, physicochemical properties, biophysics, and biological effect.

Quantum chemistry is a branch of physical chemistry focused on the application of quantum mechanics to chemical systems, including conformation, partial charges and energies. QM7, QM8, QM9 (Wu et al., 2018), COD (Ruddigkeit, Van Deursen, Blum, & Reymond, 2012) and CSD (Groom, Bruno, Lightfoot, & Ward, 2016) are datasets for quantum chemistry prediction.

Physicochemical properties are the intrinsic physical and chemical characteristics of a substance, such as bioavailability, octanol solubility, aqueous solubility and hydrophobicity. ESOL, Lipophilicity and Freesolv (Wu et al., 2018) are datasets for physicochemical properties prediction.

Biophysics properties are about the physical underpinnings of biomolecular phenomena, such as affinity, efficacy and activity. PDB-bind (Wang, Fang, Lu, Yang, & Wang, 2005), MUV, and HIV (Wu et al., 2018) are biophysics property prediction datasets.

Biological effect properties are generally defined as the response of an organism, a population, or a community to changes in its environment, such as side effects, toxicity and ADMET. Tox21, toxcast (Wu et al., 2018) and PTC (Toivonen, Srinivasan, King, Kramer, & Helma, 2003) are biological effect prediction datasets.

Moleculenet (Wu et al., 2018) is a widely-used benchmark dataset for molecule property prediction. It contains over 700,000 compounds tested on different properties. For each dataset, they provide a metric and a splitting pattern. Among the datasets, QM7, OM7b, QM8, QM9, ESOL, FreeSolv, Lipophilicity and PDBbind are regression tasks, using

Table 10

Summary of GNNs in molecular property prediction.

Type	Spatial/Spectral	Method	Application
Recurrent GNN	–	R-GNN	Biological effect (Scarselli, Gori, Tsoi, Hagenbuchner, & Monfardini, 2008)
Recurrent GNN	–	GGNN	Quantum chemistry (Mansimov, Mahmood, Kang, & Cho, 2019), Biological effect (Altae-Tran, Ramsundar, Pappu, & Pande, 2017; Feinberg et al., 2018; Withnall, Lindelöf, Engkvist, & Chen, 2020)
Recurrent GNN	–	IterRefLSTM	Biophysics (Altae-Tran et al., 2017), Biological effect (Altae-Tran et al., 2017)
Convolutional GNN	Spatial/Spectral	GCN	Quantum chemistry (Liao et al., 2019; Withnall et al., 2020; Yang et al., 2019), physicochemical properties (Coley, Barzilay, Green, Jaakkola, & Jensen, 2017; Duvenaud et al., 2015; Ryu, Lim, Hong, & Kim, 2018), Biophysics (Bouritsas, Frasca, Zafeiriou, & Bronstein, 2022; Duvenaud et al., 2015; Yang et al., 2019) Biological effect (Li, Cai and He, 2017; Wu et al., 2018)
Convolutional GNN	Spectral	LanczosNet	Quantum chemistry (Liao et al., 2019)
Convolutional GNN	Spectral	ChebNet	Physicochemical properties, Biophysics, Biological effect (Li, Wang et al., 2018)
Convolutional GNN	Spatial	GraphSAGE	Physicochemical properties (Hu et al., 2019), Biophysics (Chen, Chen, Villar and Bruna, 2020; Errica, Podda, Bacciu, & Micheli, 2019; Liang, Zhang, Gao, & Xu, 2020), Biological effect (Hu et al., 2019; Ma, Wang, Aggarwal, & Tang, 2019)
Convolutional GNN	Spatial	GAT	Physicochemical properties (Ahmad, Tayara, & Chong, 2023; Hu et al., 2019), Biophysics (Bouritsas et al., 2022; Chen, Chen, Villar et al., 2020), Biological effect (Hu et al., 2019)
Convolutional GNN	Spatial	DGCNN	Biophysics (Chen, Bian, & Sun, 2019), Biological effect (Zhang, Cui et al., 2018)
Convolutional GNN	Spatial	GIN	Physicochemical properties (Bouritsas et al., 2022; Hu et al., 2019), Biophysics (Hu et al., 2020, 2019), Biological effect (Hu et al., 2019)
Convolutional GNN	Spatial	MPNN	Physicochemical (Ma et al., 2020)
Transformer	–	MAT	Physicochemical, Biophysics (Maziarka et al., 2020)

MAE or RMSE as the evaluation metrics. Other tasks such as tox21 and toxcast are classification tasks, using AUC as evaluation metric.

11.2. Molecular graph representation learning backbones

Since node attributes and edge attributes are crucial to molecular representation, most works use GNN instead of traditional graph representation learning methods as backbones, since many GNN methods consider edge information. Existing GNNs designed for the general domain can be applied to molecular graphs. Table 10 summarizes the GNNs used for molecular property prediction and the types of properties they can be applied to predict.

Furthermore, many works customize their GNN structure by considering the chemical domain knowledge.

- First, the **chemical bonds** and molecule interaction are taken into consideration carefully. For example, Ma, Bian et al. (2020) use an additional edge GNN to model the chemical bonds separately. Specifically, given an edge (v, w), they formulate an Edge-based GNN as:

$$\mathbf{m}_{vw}^{(k)} = \text{AGG}_{\text{edge}}(\{\mathbf{h}_{vw}^{(k-1)}, \mathbf{h}_{uv}^{(k-1)}, \mathbf{x}_u | u \in \mathcal{N}_v \setminus w\}),$$
$$\mathbf{h}_{vw}^{(k)} = \text{MLP}_{\text{edge}}(\{\mathbf{m}_{vw}^{(k-1)}, \mathbf{h}_{vw}^{(0)}\}), \tag{127}$$

where $\mathbf{h}_{vw}^{(0)} = \sigma(\mathbf{W}_{\text{ein}}\mathbf{e}_{vw})$ is the input state of the Edge-based GNN, $\mathbf{W}_{\text{ein}} \in \mathbb{R}^{d_{\text{hid}} \times d_e}$ is the input weight matrix. PotentialNet (Feinberg et al., 2018) further uses different message passing operations for different edge types. DGNN-DDI (Ma & Lei, 2023) leverage dual graph neural networks to model the interaction between two molecules.

- Second, **motifs** in molecular graphs play an important role in molecular property prediction. GSN (Bouritsas et al., 2022) leverage substructure encoding to construct a topologically-aware message-passing method. Each node v updates its state $\mathbf{h}_v^t$ by combining its previous state with the aggregated messages:

$$\mathbf{h}_v^{t+1} = \text{UP}^{t+1}(\mathbf{h}_v^t, \mathbf{m}_v^{t+1}), \tag{128}$$

$$\mathbf{m}_v^{t+1} = \begin{cases} M^{t+1}([\mathbf{h}_v^t, \mathbf{h}_u^t, \mathbf{x}_v^V, \mathbf{x}_u^V, \mathbf{e}_{u,v}]_{u \in \mathcal{N}(v)}) \ (\textbf{GSN-v}) \\ \qquad\qquad\qquad \text{or} \\ M^{t+1}([\mathbf{h}_v^t, \mathbf{h}_u^t, \mathbf{x}_{u,v}^E, \mathbf{e}_{u,v}]_{u \in \mathcal{N}(v)}) \ (\textbf{GSN-e}) \end{cases}, \tag{129}$$

where $\mathbf{x}_v^V, \mathbf{x}_u^V, \mathbf{x}_{u,v}^E, \mathbf{e}_{u,v}$ contains the substructure information associated with nodes and edges, [] denotes a multiset. Yu and Gao (2022) constructs a heterogeneous graph using motifs and molecules. Motifs and molecules are both treated as nodes and the edges model the relationship between motifs and graphs, for example, if a graph contains a motif, there will be an edge between them. MGSSL (Zhang, Liu, Wang, Lu & Lee, 2021) leverages a retrosynthesis-based algorithm BRICS and additional rules to find the motifs and combines motif layers with atom layers. It is a hierarchical framework jointly modeling atom-level information and motif-level information. Aouichaoui, Fan, Mansouri, Abildskov, and Sin (2023) introduce group-contribution-based attention to highlight the most substructures in molecules.

- Third, **different feature modalities** have been used to improve molecular graph embedding. Lin, Xu, Woicik, Ma and Wang (2022) combine SMILES modality and graph modality with contrastive learning. Zhu et al. (2022) encode 2D molecular graph and 3D molecular conformation with a unified Transformer. It uses a unified model to learn 3D conformation generation given 2D graph and 2D graph generation given 3D conformation. Cremer, Medrano Sandonas, Tkatchenko, Clevert, and De Fabritiis (2023) use a Equivariant Graph Neural Networks to represent the 3D information of molecules. Liu et al. (2023) consider molecular chirality and design a chirality-aware molecular convolution module.

- Finally, **knowledge graph and literature** can provide additional knowledge for molecular property prediction. Fang et al. (2022) introduce a chemical element knowledge graph to summarize microscopic associations between elements and augment the molecular graph based on the knowledge graph, and a knowledge-aware message-passing network is used to encode the augmented graph. MuMo (Su et al., 2022) introduces biomedical literature

to guide molecular property prediction. It pretrains a GNN and a language model on paired data of molecules and literature mentions via contrastive learning:

$$\ell_i^{(\mathbf{z}_i^G, \mathbf{z}_i^T)} = -\log \frac{\exp\left(sim(\mathbf{z}_i^G, \mathbf{z}_i^T)/\tau\right)}{\sum_{j=1}^N \exp\left(sim(\mathbf{z}_i^G, \mathbf{z}_j^T)/\tau\right)}, \tag{130}$$

where $\mathbf{z}_i^G, \mathbf{z}_i^T$ are the representation of molecule and its corresponding literature. Zhao et al. (2023) propose a unified Transformer architecture to jointly model molecule graph and the corresponding bioassay description.

11.3. Training strategies

Despite the encouraging performance achieved by GNNs, the traditional supervised training scheme of GNNs faces a severe limitation: The scarcity of available molecules with desired properties. Although there are a large number of molecular graphs in public databases such as PubChem, labeled molecules are hard to acquire due to the high cost of wet-lab experiments and quantum chemistry calculations. Directly training GNNs on such limited molecules in a supervised way is prone to over-fitting and lack of generalization. To address this issue, few-shot learning and self-supervised learning are widely used in molecular property prediction.

Few-shot learning. Few-shot learning aims at generalizing to a task with a small labeled data set. The prediction of each property is treated as a single task. Metric-based and optimization-based few-shot learning have been adopted for molecular property prediction. Metric-based few-shot learning is similar to nearest neighbors and kernel density estimation, which learns a metric or distance function over objects. IterRefLSTM (Altae-Tran et al., 2017) leverages matching network (Vinyals, Blundell, Lillicrap, Wierstra, et al., 2016) as the few-shot learning framework, calculating the similarity between support samples and query samples. Optimization-based few-shot learning optimizes a meta-learner for parameter initialization which can be fast adapted to new tasks. Meta-MGNN (Guo et al., 2021) adopts MAML (Finn, Abbeel, & Levine, 2017) to train a parameter initialization to adapt to different tasks and use self-attentive task weights for each task. PAR (Wang, Abuduweili, Yao and Dou, 2021) also uses MAML framework and learns an adaptive relation graph among molecules for each task.

Self-supervised learning. Self-supervised learning can pre-train a GNN model with plenty of unlabeled molecular graphs and transfer it to specific molecular property prediction tasks. Self-supervised learning contains generative methods and predictive methods. Predictive methods design prediction tasks to capture the intrinsic data features. Pre-GNN (Hu et al., 2019) exploits both node-level and graph-level prediction tasks including context prediction, attribute masking, graph-level property prediction and structural similarity prediction. MGSSL (Zhang, Liu et al., 2021) provides a motif-based generative pre-training framework making topology prediction and motif generation iteratively. Contrastive methods learn graph representations by pulling views from the same graph close and pushing views from different graphs apart. Different views of the same graph are constructed by graph augmentation or leveraging the 1D SMILES and 3D structure. MolCLR (Wang, Wang, Cao and Barati Farimani, 2022) augments molecular graphs by atom masking, bond deletion and subgraph removal and maximizes the agreement between the original molecular graph and augmented graphs. Fang, Zhang et al. (2022) uses a chemical knowledge graph to guide the graph augmentation. SMICLR (Pinheiro, Da Silva, & Quiles, 2022) uses contrastive learning across SMILES and 2D molecular graphs. GeomGCL (Li, Zhou, Xu, Dou and Xiong, 2022) leverages graph contrastive learning to capture the geometry of the molecule across 2D and 3D views. Fang et al. (2023) and Jiang (2023) integrate molecule graphs with chemical knowledge graph and fuse the two modalities with contrastive learning. Self-supervised learning can also be combined with few-shot learning to fully leverage the hierarchical information in the training set (Ju et al., 2023).

11.4. Summary

This section introduces graph representation learning in molecular property prediction and we provide the summary as follows:

- **Techniques.** For molecular property prediction, a molecule is represented as a graph whose nodes are atoms and edges are atom-bonds (ABs). GNNs such as GCN, GAT, and GraphSAGE are adopted to learn the graph-level representation. The representations are then fed into a classification or regression head for the molecular property prediction tasks. Many works guide the model structure design with medical domain knowledge including chemical bond features, motif features, different modalities of molecular representation, chemical knowledge graph and literature. Due to the scarcity of available molecules with desired properties, few-shot learning and contrastive learning are used to train molecular property prediction models, so that the model can leverage the information in large unlabeled dataset and can be adapted to new tasks with a few examples.
- **Challenges and Limitations.** Despite the great success of graph representation learning in molecular property prediction, the methods still have limitations: (1) Few-shot molecular property prediction are not fully explored. (2) Most methods depend on training with labeled data, but neglect the chemical domain knowledge.
- **Future Works.** In the future, we expect that: (1) More few-shot learning and zero-shot learning methods are studied for molecular property prediction to solve the data scarcity problem. (2) Heterogeneous data can be fused for molecular property prediction. There are a large amount of heterogeneous data about molecules such as knowledge graphs, molecule descriptions and property descriptions. They can be considered to assist molecular property prediction. (3) Chemical domain knowledge can be leveraged for the prediction model. For example, when we perform affinity prediction, we can consider molecular dynamics knowledge.

12. Molecular generation

Molecular generation is pivotal to drug discovery, where it serves a fundamental role in downstream tasks like molecular docking (Meng, Zhang, Mezei, & Cui, 2011) and virtual screening (Walters, Stahl, & Murcko, 1998). The goal of molecular generation is to produce chemical structures that satisfy a specific molecular profile, e.g., novelty, binding affinity, and SA scores. Traditional methods have relied on 1D string formats like SMILES (Gómez-Bombarelli et al., 2018) and SELFIES (Krenn, Häse, Nigam, Friederich, & Aspuru-Guzik, 2020). With the recent advances in graph representation learning, numerous graph-based methods have also emerged, where molecular graph $\mathcal{G}$ can naturally embody both 2D topology and 3D geometry. While recent literature reviews (Du, Fu, Sun and Liu, 2022; Meyers, Fabian, & Brown, 2021) have covered the general topics of molecular design, this chapter is dedicated to the applications of graph representation learning in the molecular generation task. Molecular generation is intrinsically a *de novo* task, where molecular structures are generated from scratch to navigate and sample from the vast chemical space. Therefore, this chapter does not discuss tasks that restrict chemical structures *a priori*, such as docking (Ganea et al., 2021; Stärk, Ganea, Pattanaik, Barzilay, & Jaakkola, 2022) and conformation generation (Shi, Luo, Xu, & Tang, 2021; Zhu et al., 2022).

12.1. Taxonomy for molecular featurization methods

This section categorizes the different methods to feature molecules. The taxonomy presented here is unique to the task of molecular generation, owing to the various modalities of molecular entities, complex interactions with other bio-molecular systems and formal knowledge from the laws of chemistry and physics.

2D topology *vs.* 3D geometry. Molecular data are multi-modal by nature. For one thing, a molecule can be unambiguously represented by its 2D topological graph $\mathcal{G}_{2D}$, where atoms are nodes and bonds are edges. $\mathcal{G}_{2D}$ can be encoded by canonical MPNN models like GCN (Kipf & Welling, 2016a), GAT (Veličković et al., 2017), and R-GCN (Schlichtkrull et al., 2018), in ways similar to tasks like social networks and knowledge graphs. A typical example of this line of work is GCPN (You, Liu, Ying, Pande, & Leskovec, 2018), a graph convolutional policy network generating molecules with desired properties such as synthetic accessibility and drug-likeness.

For another, the 3D conformation of a molecule can be accurately depicted by its 3D geometric graph $\mathcal{G}_{3D}$, which incorporates 3D atom coordinates. In 3D-GNNs like SchNet (Schütt, Sauceda, Kindermans, Tkatchenko, & Müller, 2018) and OrbNet (Qiao, Welborn, Anandkumar, Manby and Miller III, 2020), $\mathcal{G}_{3D}$ is organized into a k-NN graph or a radius graph according to the Euclidean distance between atoms. It is justifiable to approximate $\mathcal{G}_{3D}$ as a 3D extension to $\mathcal{G}_{2D}$, since covalent atoms are closest to each other in most cases. However, $\mathcal{G}_{3D}$ can also find a more long-standing origin in the realm of computational chemistry (Frisch et al., 2016), where both covalent and non-covalent atomistic interactions are considered to optimize the potential surface and simulate molecular dynamics. Therefore, $\mathcal{G}_{3D}$ more realistically represents the molecular geometry, which makes a good fit for protein pocket binding and 3D-QSAR optimization (Verma, Khedkar, & Coutinho, 2010).

Molecules can rotate and translate, affecting their position in the 3D space. Therefore, it is ideal to encode these molecules with GNNs equivariant/invariant to roto-translations, which can be $\sim 10^3$ times more efficient than data augmentation (Geiger & Smidt, 2022). Equivariant GNNs can be based on irreducible representation (Anderson, Hy, & Kondor, 2019; Batzner et al., 2021; Brandstetter, Hesselink, van der Pol, Bekkers, & Welling, 2022; Fuchs, Worrall, Fischer, & Welling, 2020; Thomas et al., 2018), regular representation (Finzi, Stanton, Izmailov, & Wilson, 2020; Hutchinson et al., 2021), or scalarization (Huang et al., 2022; Jing, Eismann, Suriana, Townshend, & Dror, 2021; Klicpera, Becker, & Günnemann, 2021; Klicpera, Groß, & Günnemann, 2020; Köhler, Klein, & Noe, 2020; Liu et al., 2022; Satorras, Hoogeboom, & Welling, 2021; Schütt et al., 2018; Schütt, Unke, & Gastegger, 2021; Thölke & Fabritiis, 2022), which are explained in more detail in Han, Rong, Xu, and Huang (2022). Recent works like GraphVF (Sun, Zhan, Guo, Zhang, & Tang, 2023) and MolCode (Zhang, Liu, Lee, Hsieh and Chen, 2023) have been incorporating $\mathcal{G}_{2D}$ and $\mathcal{G}_{3D}$ to accurately capture the relationship between structure and properties in molecular design in a unified way.

Unbounded *vs.* binding-based. Earlier works have aimed to generate **unbounded** molecules in either 2D or 3D space, striving to learn good molecular representations through this task. In the 2D scenario, GraphNVP (Madhawa, Ishiguro, Nakago, & Abe, 2019) first introduces a flow-based model to learn an invertible transformation between the 2D chemical space and the latent space. GraphAF (Shi et al., 2020) further adopts an autoregressive generation scheme to check the valence of the generated atoms and bonds. In the 3D scenario, G-SchNet (Gebauer, Gastegger, & Schütt, 2019) first proposes to utilize $\mathcal{G}_{3D}$ (instead of 3D density grids) as the generation backbone. It encodes $\mathcal{G}_{3D}$ via SchNet, and uses an auxiliary token to generate atoms on the discretized 3D space autoregressively. G-SphereNet (Luo & Ji, 2022)

uses symmetry-invariant representations in a spherical coordinate system (SCS) to generate atoms in the continuous 3D space and preserve equivariance.

Unbounded models adopt certain techniques to optimize specific properties of the generated molecules. GCPN and GraphAF use scores like logP, QED, and chemical validity to tune the model via reinforcement learning. EDM (Hoogeboom, Satorras, Vignac, & Welling, 2022) can generate 3D molecules with property c by re-training the diffusion model with c's feature vector concatenated to the E(n) equivariant dynamics function $\hat{\epsilon}_t = \phi\left(z_t, [t, c]\right)$. cG-SchNet (Gebauer, Gastegger, Hessmann, Müller, & Schütt, 2022) adopts a conditioning network architecture to jointly target multiple electronic properties during conditional generation without the need to re-train the model. RetMol (Wang et al., 2022) uses a retrieval-based model for controllable generation.

On the other hand, **binding-based** methods generate drug-like molecules (aka. ligands) according to the binding site (aka. binding pocket) of a protein receptor. Drawing inspirations from the lock-and-key model for enzyme action (Fischer, 1894), works like LiGAN (Ragoza, Masuda, & Koes, 2022) and DESERT (Long, Zhou, Dai, & Zhou, 2022) uses 3D density grids to fit the density surface between the ligand and the receptor, encoded by 3D-CNNs. Meanwhile, a growing amount of literature has adopted $\mathcal{G}_{3D}$ for representing ligand and receptor molecules, because $\mathcal{G}_{3D}$ more accurately depicts molecular structures and atomistic interactions both within and between the ligand and the receptor. Representative works include 3D-SBDD (Luo, Guan, Ma and Peng, 2021), GraphBP (Liu, Luo, Uchino, Maruhashi and Ji, 2022), Pocket2Mol (Peng et al., 2022), and DiffSBDD (Schneuing et al., 2022). GraphBP shares a similar workflow with G-SphereNet, except that the receptor atoms are also incorporated into $\mathcal{G}_{3D}$ to depict the 3D geometry at the binding pocket.

Atom-based *vs.* fragment-based. Molecules are inherently hierarchical structures. At the atomistic level, molecules are represented by encoding atoms and bonds. At a coarser level, molecules can also be represented as molecular fragments like functional groups or chemical sub-structures. Both the composition and the geometry are fixed within a given fragment, e.g., the planar peptide-bond (–CO–NH–) structure. Fragment-based generation effectively reduces the degree of freedom (DOF) of chemical structures, and injects well-established knowledge about molecular patterns and reactivity. JT-VAE (Jin, Barzilay, & Jaakkola, 2018) decomposes 2D molecular graph $\mathcal{G}_{2D}$ into a junction-tree structure $\mathcal{T}$, which is further encoded via tree message-passing. DeepScaffold (Li et al., 2019) expands the provided molecular scaffold into 3D molecules. L-Net (Li, Pei, & Lai, 2021a) adopts a graph U-Net architecture and devises a custom three-level node clustering scheme for pooling and unpooling operations in molecular graphs. A number of works have also emerged lately for fragment-based generation in the binding-based setting, including FLAG (Zhang, Min, Zheng and Liu, 2022) and FragDiff (Peng, Guan, Peng, & Ma, 2023). FLAG uses a regression-based approach to sequentially decide the type and torsion angle of the next fragment to be placed at the binding site, and finally optimizes the molecule conformation via a pseudo-force field. FragDiff also adopts a sequential generation process but uses a diffusion model to determine the type and pose of each fragment in one go.

12.2. Generative methods for molecular graphs

For a molecular graph generation process, the model first learns a latent distribution $P(Z|\mathcal{G})$ characterizing the input molecular graphs. A new molecular graph $\hat{\mathcal{G}}$ is then generated by sampling and decoding from this learned distribution. Various models have been adopted to generate molecular graphs, including generative adversarial network (GAN), variational auto-encoder (VAE), normalizing flow (NF), diffusion model (DM), and autoregressive model (AR).

Generative adversarial network (GAN). GAN (Goodfellow et al., 2020) is trained to discriminate real data x from generated data z, with

the training object formalized as

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))], \quad (131)$$

where $G(\cdot)$ is the generator function and $D(\cdot)$ is the discriminator function. For example, MolGAN (De Cao & Kipf, 2018) encodes $\mathcal{G}_{\text{2D}}$ with R-GCN, trains D and G with improved W-GAN (Arjovsky, Chintala, & Bottou, 2017), and uses reinforcement learning to generate attributed molecules, where the score function is assigned from RDKit (Landrum et al., 2013) and chemical validity.

Variational auto-encoder (VAE). In VAE (Kingma & Welling, 2013), the decoder parameterizes the conditional likelihood distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, and the encoder parameterizes an approximate posterior distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x}) \approx p_\theta(\boldsymbol{z}|\boldsymbol{x})$. The model is optimized by the evidence lower bound (ELBO), consisting of the reconstruction loss term and the distance loss term:

$$\max_{\theta, \phi} \mathcal{L}_{\theta, \phi}(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{z} \sim q_\phi(\cdot|\boldsymbol{x})} \left[\ln \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] = \ln p_\theta(\boldsymbol{x}) - D_{\text{KL}} \left(q_\phi(\cdot|\boldsymbol{x}) \parallel p_\theta(\cdot|\boldsymbol{x}) \right).$$
$$(132)$$

Maximizing ELBO is equivalent to simultaneously maximizing the log-likelihood of the observed data, and minimizing the divergence of the approximate posterior $q_\phi(\cdot|x)$ from the exact posterior $p_\theta(\cdot|x)$. Representative works along this thread include JT-VAE (Jin et al., 2018), GraphVAE (Simonovsky & Komodakis, 2018), and CGVAE (Liu, Allamanis, Brockschmidt and Gaunt, 2018) for the 2D generation task, and 3DMolNet (Nesterov, Wieser, & Roth, 2020) for the 3D generation task.

Autoregressive model (AR). Autoregressive model is an umbrella definition for any model that sequentially generates the components (atoms or fragments) of a molecule. ARs better capture the interdependency within the molecular structure and allows for explicit valency check. For each step in AR, the new component can be produced using different techniques:

- Regression/classification, such is the case with 3D-SBDD (Luo, Guan et al., 2021), Pocket2Mol (Peng et al., 2022), etc.
- Reinforcement learning, such is the case with L-Net (Li et al., 2021a), DeepLigBuilder (Li, Pei, & Lai, 2021b), etc.
- Probabilistic models like normalizing flow and diffusion.

Normalizing flow (NF). Based on the change-of-variable theorem, NF (Rezende & Mohamed, 2015) constructs an invertible mapping f between a complex data distribution $\boldsymbol{x} \sim X$: and a normalized latent distribution $\boldsymbol{z} \sim Z$. Unlike VAE, which has juxtaposed parameters for encoder and decoder, the flow model uses the same set of parameter for encoding and encoding: reverse flow f^{-1} for generation, and forward flow f for training:

$$\max_f \log p(\boldsymbol{x}) = \log p_K(\boldsymbol{z}_K) \quad (133)$$

$$= \log p_{K-1}(\boldsymbol{z}_{K-1}) - \log \left| \det \left(\frac{df_K(\boldsymbol{z}_{K-1})}{d\boldsymbol{z}_{K-1}} \right) \right| \quad (134)$$

$$= \cdots \quad (135)$$

$$= \log p_0(\boldsymbol{z}_0) - \sum_{i=1}^{K} \log \left| \det \left(\frac{df_i(\boldsymbol{z}_{i-1})}{d\boldsymbol{z}_{i-1}} \right) \right|, \quad (136)$$

where $f = f_K \circ f_{K-1} \circ \cdots \circ f_1$ is a composite of K blocks of transformation. While GraphNVP (Madhawa et al., 2019) generates the molecular graph with NF in one go, following works tend to use the autoregressive flow model, including GraphAF (Shi et al., 2020), GraphDF (Luo, Yan and Ji, 2021), GraphBP (Liu, Luo et al., 2022) and SiamFlow (Tan, Gao and Li, 2023).

Diffusion model (DM). Diffusion models (Ho, Jain, & Abbeel, 2020; Sohl-Dickstein, Weiss, Maheswaranathan, & Ganguli, 2015; Song & Ermon, 2019) define a Markov chain of diffusion steps to slowly add random noise to data $\boldsymbol{x}_0 \sim q(\boldsymbol{x})$:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1 - \beta_t} \boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}), \quad (137)$$

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}). \quad (138)$$

They then learn to reverse the diffusion process to construct desired data samples from the noise:

$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad (139)$$

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)), \quad (140)$$

while the models are trained using a variational lower bound. Diffusion models have been applied to generate unbounded 3D molecules in EDM (Hoogeboom et al., 2022) and GCDM (Morehead & Cheng, 2023), and binding-specific ligands in DiffSBDD (Schneuing et al., 2022), DiffBP (Lin et al., 2022) and TargetDiff (Guan et al., 2023). Diffusion can also be applied to generate molecular fragments in autoregressive models, as is the case with FragDiff (Peng et al., 2023).

12.3. Summary and prospects

We wrap up this chapter with Table 11, which profiles existing molecular generation models according to their taxonomy for molecular featurization, the GNN backbone, and the generative method. This chapter covers the critical topics of molecular generation, which also elicit valuable insights into the promising directions for future research. We summarize these important aspects as follows.

Techniques. Graph neural networks can be flexibly leveraged to encode molecular features on different representation levels and across different problem settings. Canonical GNNs like GCN (Kipf & Welling, 2016a), GAT (Veličković et al., 2017), and R-GCN (Schlichtkrull et al., 2018) have been widely adopted to model 2D molecular graphs, while 3D equivariant GNNs have also been effective in modeling 3D molecular graphs. In particular, this 3D approach can be readily extended to binding-based scenarios, where the 3D geometry of the binding protein receptor is considered alongside the ligand geometry *per se*. Fragment-based models like JT-VAE (Jin et al., 2018) and L-Net (Li et al., 2021a) can also effectively capture the hierarchical molecular structure. Various generative methods have also been effectively incorporated into the molecular setting, including generative adversarial network (GAN), variational auto-encoder (VAE), autoregressive model (AR), normalizing flow (NF), and diffusion model (DM). These models have been able to generate valid 2D molecular topologies and realistic 3D molecular geometries, greatly accelerating the search for drug candidates.

Challenges and Limitations. While there has been an abundant supply of unlabeled molecular structural and geometric data (Francoeur et al., 2020; Irwin, Sterling, Mysinger, Bolstad, & Coleman, 2012; Spackman, Jayatilaka, & Karton, 2016), the number of labeled molecular data over certain critical biochemical properties like toxicity (Gayvert, Madhukar, & Elemento, 2016) and solubility (Delaney, 2004) remain very limited. On the other hand, existing models have heavily relied on expert-crafted metrics to evaluate the quality of the generated molecules, such as QED and Vina (Eberhardt, Santos-Martins, Tillack, & Forli, 2021), rather than actual wet lab experiments.

Future Works. Besides the structural and geometric attributes described in this chapter, an even more extensive array of data can be applied to aid molecular generation, including chemical reactions and medical ontology. These data can be organized into a heterogeneous knowledge graph to aid the extraction of high-quality molecular representations. Furthermore, high throughput experimentation (HTE) should be adopted to realistically evaluate the synthesizablity and druggability of the generated molecules in the wet lab. Concrete case studies, such as the design of potential inhibitors to SARS-CoV-2 (Li et al., 2021b), would be even more encouraging, bringing new insights

Table 11
Summary of molecular generation models.

Model	2D/3D	Binding-based	Fragment-based	GNN backbone	Generative model
GCPN (You et al., 2018)	2D			GCN (Kipf & Welling, 2016a)	GAN
MolGAN (De Cao & Kipf, 2018)	2D			R-GCN (Schlichtkrull et al., 2018)	GAN
DEFactor (Assouel, Ahmed, Segler, Saffari, & Bengio, 2018)	2D			GCN	GAN
GraphVAE (Simonovsky & Komodakis, 2018)	2D			ECC (Simonovsky & Komodakis, 2017)	VAE
MDVAE (Du, Guo, Shehu and Zhao, 2022)	2D			GGNN (Li, Tarlow, Brockschmidt, & Zemel, 2015)	VAE
JT-VAE (Jin et al., 2018)	2D		✓	MPNN (Gilmer et al., 2017)	VAE
CGVAE (Liu, Allamanis et al., 2018)	2D			GGNN	VAE
DeepScaffold (Li, Hu et al., 2019)	2D		✓	GCN	VAE
GraphNVP (Madhawa et al., 2019)	2D			R-GCN	NF
MoFlow (Zang & Wang, 2020)	2D			R-GCN	NF
GraphAF (Shi et al., 2020)	2D			R-GCN	NF + AR
GraphDF (Luo, Yan et al., 2021)	2D			R-GCN	NF + AR
L-Net (Li et al., 2021a)	3D		✓	g-U-Net (Gao & Ji, 2019b)	AR
G-SchNet (Gebauer et al., 2019)	3D			SchNet (Schütt et al., 2018)	AR
GEN3D (Roney, Maragakis, Skopp, & Shaw, 2021)	3D			EGNN (Satorras et al., 2021)	AR
G-SphereNet (Luo & Ji, 2022)	3D			SphereNet (Liu, Wang et al., 2022)	NF + AR
EDM (Hoogeboom et al., 2022)	3D			EGNN	DM
GCDM (Morehead & Cheng, 2023)	3D			GCPNet (Morehead & Cheng, 2022)	DM
3D-SBDD (Luo, Guan et al., 2021)	3D	✓		SchNet	AR
Pocket2Mol (Peng et al., 2022)	3D	✓		GVP (Jing, Eismann, Suriana, Townshend, & Dror, 2020)	AR
FLAG (Zhang, Min et al., 2022)	3D	✓	✓	SchNet	AR
GraphBP (Liu, Luo et al., 2022)	3D	✓		SchNet	NF + AR
SiamFlow (Tan, Gao et al., 2023)	3D	✓		R-GCN	NF
DiffBP (Lin, Huang et al., 2022)	3D	✓		EGNN	DM
DiffSBDD (Schneuing et al., 2022)	3D	✓		EGNN	DM
TargetDiff (Guan et al., 2023)	3D	✓		EGNN	DM
FragDiff (Peng et al., 2023)	2D + 3D	✓	✓	MPNN	DM + AR
GraphVF (Sun et al., 2023)	2D + 3D	✓	✓	SchNet	NF + AR
MolCode (Zhang, Liu et al., 2023)	2D + 3D	✓		EGNN	NF + AR

into leveraging these molecular generative models to facilitate the design and fabrication of potent and applicable drug molecules in the pharmaceutical industry.

Integrating Large Language Models (LLMs) like GPT-4 (OpenAI, 2023) with graph-based representations offers a promising new direction in molecular generation. Recent studies like those by Guo et al. (2023) and Jablonka, Schwaller, Ortega-Guerrero, and Smit (2023) highlight LLMs' potential in chemistry, especially in low-data scenarios. While current LLM-based approaches in this domain, including those by Bagal, Aggarwal, Vinod, and Priyakumar (2021) and Mazuz, Shtar, Shapira, and Rokach (2023), predominantly utilize textual SMILES strings, their potential is somewhat constrained by the limits of text-only inputs. The emerging trend, exemplified by Liu, Ren and Ren (2023), is to leverage multi-modal data, integrating graph, image, and text, which could more comprehensively capture the intricacies of molecular structures. This approach marks a significant shift toward utilizing graph-based information alongside traditional text, enhancing the capability of LLMs in molecular generation. Such advances suggest that future research should focus more on exploiting the synergy between graph-based molecular representations and the evolving landscape of LLMs to address complex challenges in chemistry and material sciences.

13. Recommender systems

The use of graph representation learning in recommender systems has been drawing increasing attention as one of the key strategies for addressing the issue of information overload. With their strong ability to capture high-order connectivity between graph nodes, deep graph representation learning has been shown to be beneficial in enhancing recommendation performance across a variety of recommendation scenarios.

Typical recommender systems take the observed interactions between users and items and their fixed features as input, and are intended for making proper predictions on which items a specific user is probably interested in. To formulate, given an user set $\mathcal{U}$, an item set $\mathcal{I}$ and the interaction matrix between users and items $X \in \{0, 1\}^{|\mathcal{U}| \times |\mathcal{I}|}$,

where $X_{u,v}$ indicates there is an observed interaction between user u and item i. The target of GNNs on recommender systems is to learn representations $h_u, h_i \in \mathbb{R}^d$ for given u and i. The preference score can further be calculated by a similarity function:

$$\hat{x}_{u,i} = f(h_u, h_i), \tag{141}$$

where $f(\cdot, \cdot)$ is the similarity function, e.g. inner product, cosine similarity, multi-layer perceptrons that takes the representation of u and i and calculate the preference score $\hat{x}_{u,i}$.

When it comes to adapting graph representation learning in recommender systems, a key step is to construct graph-structured data from the interaction set X. Generally, a graph is represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}, \mathcal{E}$ denotes the set of vertices and edges respectively. According to the construction of $\mathcal{G}$, we can categorize the existing works as follows into three parts which are introduced in the following subsections. A summary is provided in Table 12.

13.1. User-item bipartite graph

13.1.1. Graph construction

A undirected bipartite graph where the vertex set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ and the undirected edge set $\mathcal{E} = \{(u, i) | u \in \mathcal{U} \wedge i \in \mathcal{I}\}$. Under this case the graph adjacency can be directly obtained from the interaction matrix, thus the optimization target on the user–item bipartite graph is equivalent to collaborative filtering tasks such as MF (Koren, Bell, & Volinsky, 2009) and SVD++ (Koren, 2008).

There have been plenty of previous works that applied GNNs on the constructed user-item bipartite graphs. GC-MC (Berg et al., 2017) firstly applies graph convolution networks to user–item recommendation and optimizes a graph autoencoder (GAE) to reconstruct interactions between users and items. NGCF (Wang, He et al., 2019) introduces the concept of Collaborative Filtering (CF) into graph-based recommendations by modeling the affinity between neighboring nodes on the interaction graph. MMGCN (Wei et al., 2019) extends the graph-based recommendation to multi-modal scenarios by constructing different subgraphs for each modal. LightGCN (He et al., 2020) improves NGCF

Table 12
Summary of graph models for recommender systems.

Model	Recommendation task	Graph structure	Graph encoder	Representation
GC-MC (Berg, Kipf, & Welling, 2017)	Matrix completion	User-item graph	GCN	Last-layer
NGCF (Wang, He, Wang, Feng & Chua, 2019)	Collaborative filtering	User-item graph	GCN+Affinity	Concatenation
MMGCN (Wei et al., 2019)	Micro-video	Multi-modal graph	GCN	Last-layer
LightGCN (He et al., 2020)	Collaborative filtering	User-item graph	LGC	Mean-pooling
DGCF (Wang et al., 2020)	Collaborative filtering	User-item graph	Dynamic routing	Mean-pooling
CAGCN (Wang, Zhao, Zhang and Derr, 2023)	Collaborative filtering	User-item graph	GCN+CIR	Mean-pooling
SR-GNN (Wu et al., 2019)	Session-based	Transition graph	GGNN	Soft-attention
GC-SAN (Wu, Tang et al., 2019; Xu et al., 2019)	Session-based	Session graph	GGNN	Self-attention
FGNN (Qiu, Li, Huang, & Yin, 2019)	Session-based	Session graph	GAT	Last-layer
GAG (Qiu, Yin, Huang and Chen, 2020)	Session-based	Session graph	GCN	Self-attention
GCE-GNN (Wang et al., 2020)	Session-based	Transition+Global	GAT	Sum-pooling
HyperRec (Wang, Ding, Hong, Liu and Caverlee, 2020)	Sequence-based	Sequential hypergraph	HGCN	Self-attention
DHCF (Ji et al., 2020)	Collaborative filtering	Dual hypergraph	JHConv	Last-layer
MBHT (Yang et al., 2022)	Sequence-based	Learnable hypergraph	Transformer	Cross-view attention
HCCF (Xia et al., 2022)	Collaborative filtering	Learnable hypergraph	HGCN	Last-layer
H³Trans (Xu et al., 2023)	Sequence-based	Hierarchical hypergraph	Message-passing	Last-layer
STHGCN (Yan et al., 2023)	POI recommendation	Spatio-temporal hypergraph	HGCN	Mean-pooling

by removing the non-linear activation functions and simplifying the message function. With the development of disentangled representation learning, there are works like DGCF (Wang, Jin et al., 2020) that introduce disentangled graph representation learning to represent users and items from multiple disentangled perspectives. Additionally, having realized the limitation of the existing message-passing scheme in capturing collaborative signals, CAGCN (Wang, Zhao et al., 2023) proposes Common Interacted Ratio (CIR) as a recommendation-oriented topological metric for GNN-based recommender models.

13.1.2. Graph propagation scheme

A common practice is to follow the traditional message-passing networks (MPNNs) and design the graph propagation method accordingly. GC-MC adopts vanilla GCNs to encode the user–item bipartite graph. NGCF enhances GCNs by considering the affinity between users and items. The message function of NGCF from node j to i is formulated as:

$$\begin{cases} m_{i \leftarrow j} = \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}}(W_1 e_j + W_2(e_i \odot e_j)) \\ m_{i \leftarrow i} = W_1 e_i, \end{cases} \tag{142}$$

where W_1, W_2 are trainable parameters, e_i represents i's representation from previous layer. The matrix form can be further provided by:

$$E^{(l)} = \text{LeakyReLU}((\mathcal{L} + I)E^{(l-1)}W_1^{(l)} + \mathcal{L}E^{(l-1)} \odot E^{(l-1)}W_2^{(l)}), \tag{143}$$

where $\mathcal{L}$ represents the Laplacian matrix of the user–item graph. The element-wise product in Eq. (143) represents the affinity between connected nodes, containing the collaborative signals from interactions.

However, the notable heaviness and burdensome calculation of NGCF's architecture hinder the model from making faster recommendations on larger graphs. LightGCN solves this issue by proposing Light Graph Convolution (LGC), which simplifies the convolution operation with:

$$e_i^{(l+1)} = \sum_{j \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}} e_j^{(l)}. \tag{144}$$

When an interaction takes place, e.g. a user clicks a particular item, there could be multiple intentions behind the observed interaction. Thus it is necessary to consider the various disentangled intentions among users and items. DGCF proposes the cross-intent embedding propagation scheme on the graph, inspired by the dynamic routing algorithm of capsule networks (Sabour, Frosst, & Hinton, 2017). To formulate, the propagation process maintains a set of routing logits $\tilde{S}_k(u, i)$ for each user u. The weighted sum aggregator to get the representation

of u can be defined as:

$$u_k^t = \sum_{i \in \mathcal{N}_u} \mathcal{L}_k^t(u, i) \cdot i_k^0 \tag{145}$$

for tth iteration, where $\mathcal{L}_k^t(u, i)$ denotes the Laplacian matrix of $S_k^t(u, i)$, formulated as:

$$\mathcal{L}_k^t(u, i) = \frac{S_k^t}{\sqrt{[\sum_{i' \in \mathcal{N}_u} S_k^t(u, i')] \cdot [\sum_{u' \in \mathcal{N}_i} S_k^t(u', i)]}}. \tag{146}$$

13.1.3. Node representations

After the graph propagation module outputs node-level representations, there are multiple methods to leverage node representations for recommendation tasks. A plain solution is to apply a readout function on layer outputs like the concatenation operation used by NGCF:

$$e^* = Concat(e^{(0)}, \dots, e^{(L)}) = e^{(0)} \| \dots \| e^{(L)}. \tag{147}$$

However, the readout function among layers would neglect the relationship between the target item and the current user. A general solution is to use the attention mechanism (Vaswani et al., 2017) to reweight and aggregate the node representations. SR-GNN adapts soft-attention mechanism to model the item-item relationship:

$$\begin{aligned} \alpha_i &= \mathbf{q}^T \sigma(W_1 e_t + W_2 e_i + c), \\ s_g &= \sum_{i=1}^{n-1} \alpha_i e_i, \end{aligned} \tag{148}$$

where $\mathbf{q}$, W_1, W_2 are trainable matrices.

Some methods focus on exploiting information from multiple graph structures. A common practice is contrastive learning, which maximizes the mutual information between hidden representations from several views. HCCF leverage InfoNCE loss as the estimator of mutual information, given a pair of representation z_i, Γ_i for node i, controlled by temperature parameter τ:

$$\mathcal{L}_{InfoNCE}(i) = -\log \frac{\exp(cosine(z_i, \Gamma_i))/\tau}{\sum_{i' \neq i} \exp(cosine(z_i, \Gamma_{i'}))/\tau}. \tag{149}$$

Besides InfoNCE, there exist several other ways to combine node representations from different views. For instance, MBHT applies an attention mechanism to fuse multiple semantics, DisenPOI adapts bayesian personalized ranking loss (BPR) (Rendle, Freudenthaler, Gantner, & Schmidt-Thieme, 2012) as a soft estimator for contrastive learning, and KBGNN applies pair-wise similarities to ensure the consistency from two views.

13.2. Transition graph

13.2.1. Transition graph construction

Since sequence-based recommendation (SR) is one of the fundamental problems in recommender systems, some researches focus on modeling the sequential information with GNNs. A commonly applied way is to construct transition graphs based on each given sequence according to the clicking sequence by a user. To formulate, given a user u's clicking sequence $s_u = [i_{u,1}, i_{u,2}, \ldots, i_{u,n}]$ containing n items, noting that there could be duplicated items, the sequential graph is constructed via $\mathcal{G}_s = \{SET(s_u), \mathcal{E}\}$, where $\forall \langle i_j, i_k \rangle \in \mathcal{E}$ indicates there exists a successive transition from i_j to i_k. Since $\mathcal{G}_s$ are directed graphs, a widely used way to depict graph connectivity is by building the connection matrix $A_s \in \mathbb{R}^{n \times 2n}$. A_s is the combination of two adjacency matrices $A_s = [A_s^{(in)}; A_s^{(out)}]$, which denotes the normalized node degrees of incoming and outgoing edges in the session graph respectively.

The proposed transition graphs that obtain user behavior patterns have been demonstrated important to session-based recommendations (Li et al., 2017; Liu, Zeng, Mokhosi and Zhang, 2018). SR-GNN and GC-SAN (Wu, Tang et al., 2019; Xu, Zhao et al., 2019) propose to leverage transition graphs and apply attention-based GNNs to capture the sequential information for session-based recommendation. FGNN (Qiu et al., 2019) formulates the recommendation within a session as a graph classification problem to predict the next item for an anonymous user. GAG (Qiu, Yin et al., 2020) and GCE-GNN (Wang, Wei et al., 2020) further extend the model to capture global embeddings among multiple session graphs.

13.2.2. Session graph propagation

Since the session graphs are directed item graphs, there have been multiple session graph propagation methods to obtain node representations on session graphs.

SR-GNN leverages Gated Graph Neural Networks (GGNNs) to obtain sequential information from a given session graph adjacency $A_s = [A_s^{(in)}; A_s^{(out)}]$ and item embedding set $\{e_i\}$:

$$a_t = A_s[e_1, \ldots, e_{t-1}]^T H + b, \tag{150}$$

$$z_t = \sigma(W_z a_t + U_z e_{t-1}), \tag{151}$$

$$r_t = \sigma(W_r a_t + U_r e_{t-1}), \tag{152}$$

$$\tilde{e}_t = \tanh(W_o a_t + U_o(r_t \odot e_{t-1})), \tag{153}$$

$$e_t = (1 - z_t) \odot e_{t-1} + z_t \tilde{e}_t, \tag{154}$$

where Ws and Us are trainable parameters. GC-SAN extend GGNN by calculating initial state a_t separately to better exploit transition information:

$$a_t = Concat(A_s^{(in)}([e_1, \ldots, e_{t-1}W_a^{(in)}] + b^{(in)}),$$
$$A_s^{(out)}([e_1, \ldots, e_{t-1}W_a^{(out)}] + b^{(out)})). \tag{155}$$

13.3. HyperGraph

13.3.1. Hypergraph topology construction

Motivated by the idea of modeling hyper-structures and high-order correlation among nodes, hypergraphs (Feng et al., 2019) are proposed as extensions of the commonly used graph structures. For graph-based recommender systems, a common practice is to construct hyper structures among the original user–item bipartite graphs. To be specific, an incidence matrix of a graph with vertex set $\mathcal{V}$ is presented as a binary matrix $H \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{E}|}$, where $\mathcal{E}$ represents the set of hyperedges. Each entry $h(v, e)$ of H depicts the connectivity between vertex v and hyperedge e:

$$h(v, e) = \begin{cases} 1 \ if \ v \in e \\ 0 \ if \ v \notin e. \end{cases} \tag{156}$$

Given the formulation of hypergraphs, the degrees of vertices and hyperedges of H can then be defined with two diagonal matrices $D_v \in \mathbb{N}^{|\mathcal{V}| \times |\mathcal{V}|}$ and $D_e \in \mathbb{N}^{|\mathcal{E}| \times |\mathcal{E}|}$, where

$$D_v(i; i) = \sum_{e \in \mathcal{E}} h(v_i, e), \qquad D_e(j; j) = \sum_{v \in \mathcal{V}} h(v, e_j). \tag{157}$$

The development of Hypergraph Neural Networks (HGNNs) (Feng et al., 2019; Huang, Elhoseiny, Elgammal, & Yang, 2015; Zhou, Huang, & Schölkopf, 2006) have shown to be capable of capturing the high-order connectivity between nodes. HyperRec (Wang, Ding et al., 2020) firstly attempts to leverage hypergraph structures for sequential recommendation by connecting items with hyperedges according to the interactions with users during different time periods. DHCF (Ji et al., 2020) proposes to construct hypergraphs for users and items respectively based on certain rules, to explicitly capture the collaborative similarities via HGNNs. MBHT (Yang, Huang et al., 2022) combines hypergraphs with a low-rank self-attention mechanism to capture the dynamic heterogeneous relationships between users and items. HCCF (Xia, Huang et al., 2022) uses the contrastive information between hypergraph and interaction graph to enhance the recommendation performance. To extend the model's ability to multi-domain categories of items, H³Trans (Xu, Wei et al., 2023) incorporates two hyperedge-based modules and leverages hierarchical hypergraph propagation to transfer from domains. STHGCN (Yan et al., 2023) formulates a spatio-temporal hypergraph structure for POI recommendation.

13.3.2. Hyper graph message passing

With the development of HGNNs, previous works have proposed different variants of HGNN to better exploit hypergraph structures. A classic high-order hyper convolution process on a fixed hypergraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with hyper adjacency H is given by:

$$g \star X = D_v^{-1/2} H D_e^{-1} H^T D_v^{-1/2} X \Theta, \tag{158}$$

where D_v, D_e are degree matrices of nodes and hyperedges, Θ denotes the convolution kernel. For hyper adjacency matrix H, DHCF refers to a rule-based hyperstructure via k-order reachable rule, where nodes in the same hyperedge group are k-order reachable to each other:

$$A_u^k = \min(1, \text{power}(A \cdot A^T, k)), \tag{159}$$

where A denotes the graph adjacency matrix. By considering the situations where $k = 1, 2$, the matrix formulation of the hyper connectivity of users and items is calculated with:

$$\begin{cases} H_u = A \parallel (A(A^T A)) \\ H_i = A^T \parallel (A^T(AA^T)), \end{cases} \tag{160}$$

which depicts the dual hypergraphs for users and items.

HCCF proposes to construct a learnable hypergraph to depict the global dependencies between nodes on the interaction graph. To be specific, the hyperstructure is factorized with two low-rank embedding matrices to achieve model efficiency:

$$H_u = E_u \cdot W_u, \ H_v = E_v \cdot W_v. \tag{161}$$

13.4. Other graphs

Since there are a variety of recommendation scenarios, several tailored designed graph structures have been proposed accordingly, to better exploit the domain information from different scenarios. For instance, CKE (Zhang, Yuan, Lian, Xie, & Ma, 2016) and MKR (Wang, Zhang et al., 2019) introduce Knowledge graphs to enhance graph recommendation. GSTN (Wang, Zhu et al., 2022), KBGNN (Ju, Qin et al., 2022), DisenPOI (Qin et al., 2022) and Diff-POI (Qin, Wu, Ju, Luo, & Zhang, 2023) propose to build geographical graphs based on the distance between Point-of-Interests (POIs) to better model the locality of users' visiting patterns. TGSRec (Fan et al., 2021) and DisenCTR (Wang et al., 2022) empower the user–item interaction graphs with

temporal sampling between layers to obtain sequential information from static bipartite graphs.

13.5. Summary

This section introduces the application of different kinds of graph neural networks in recommender systems and can be summarized as follows:

- **Graph Constructions.** There are multiple options for constructing graph-structured data for a variety of recommendation tasks. For instance, the user–item bipartite graphs reveal the high-order collaborative similarity between users and items, and the transition graph is suitable for encoding sequential information in clicking history. These diversified graph structures provide different views for node representation learning on users and items, and can be further used for downstream ranking tasks.
- **Challenges and Limitations.** Though the superiority of graph-structured data and GNNs against traditional methods has been widely illustrated, there are still challenges unsolved. For example, the computational cost of graph methods is normally expensive and thus unacceptable in real-world applications. The data sparsity and cold-started issue in graph recommendation remains to be explored as well.
- **Future Works.** In the future, an efficient solution for applying GNNs in recommendation tasks is expected. There are also some attempts (Fan et al., 2021; Qin, Ju, Wu, Luo, & Zhang, 2024; Wang, Qin et al., 2022) on incorporating temporal information in graph representation learning for sequential recommendation tasks.

14. Traffic analysis

Intelligent Transportation Systems (ITS) are essential for safe, reliable, and efficient transportation in smart cities, serving the daily commuting and traveling needs of millions of people. To support ITS, advanced modeling and analysis techniques are necessary, and Graph Neural Networks (GNNs) are a promising tool for traffic analysis. GNNs can effectively model spatial correlations, making them well-suited for analyzing complex transportation networks. As such, GNNs have garnered significant interest in the traffic domain for their ability to provide insights into traffic patterns and behaviors (Li et al., 2024).

In this section, we first conclude the main GNN research directions in the traffic domain, and then we summarize the typical graph construction processes in different traffic scenes and datasets. Finally, we list the classical GNN workflows for dealing with tasks in traffic networks. A summary is provided in Table 13.

14.1. Research directions in traffic domain

We summarize main GNN research directions in the traffic domain as follows,

- **Traffic Flow Forecasting.** Traffic flow forecasting plays an indispensable role in ITS (Dimitrakopoulos & Demestichas, 2010; Ran & Boyce, 2012), which involves leveraging spatial–temporal data collected by various sensors to gain insights into future traffic patterns and behaviors. Classic methods, like autoregressive integrated moving average (ARIMA) (Box & Pierce, 1970), support vector machine (SVM) (Hearst, Dumais, Osuna, Platt, & Scholkopf, 1998) and recurrent neural networks (RNN) (Connor, Martin, & Atlas, 1994) can only model time series separately without considering their spatial connections. To address this issue, graph neural networks (GNNs) have emerged as a powerful approach for traffic forecasting due to their strong ability of modeling complex graph-structured correlations (Bui, Cho, & Yi,

2021; Jiang & Luo, 2022; Li, Yu, Zhang and Xu, 2023; Oskarsson, Sidén, & Lindsten, 2023; Rao et al., 2022; Xie et al., 2020; Zhao et al., 2023).
- **Trajectory Prediction.** Trajectory prediction is a crucial task in various applications, such as autonomous driving and traffic surveillance, which aims to forecast future positions of agents in the traffic scene. However, accurately predicting trajectories can be challenging, as the behavior of an agent is influenced not only by its own motion but also by interactions with surrounding objects. To address this challenge, Graph Neural Networks (GNNs) have emerged as a promising tool for modeling complex interactions in trajectory prediction (Cao, Li, Ma and Tomizuka, 2021; Mohamed, Qian, Elhoseiny, & Claudel, 2020; Sun, Jiang and Lu, 2020; Zhou et al., 2021). By representing the scene as a graph, where each node corresponds to an agent and the edges capture interactions between them, GNNs can effectively capture spatial dependencies and interactions between agents. This makes GNNs well-suited for predicting trajectories that accurately capture the behavior of agents in complex traffic scenes.
- **Traffic Anomaly Detection.** Anomaly detection is an essential support for ITS. There are lots of traffic anomalies in daily transportation systems, for example, traffic accidents, extreme weather and unexpected situations. Handling these traffic anomalies timely can improve the service quality of public transportation. The main trouble of traffic anomaly detection is the highly twisted spatial–temporal characteristics of traffic data. The criteria and influence of traffic anomaly vary among locations and times. GNNs have been introduced and achieved success in this domain (Chen, Chen, Zhang, Yuan and Cheng, 2021; Deng & Hooi, 2021; Deng, Lian, Huang, & Chen, 2022; Zhang et al., 2022).
- **Others.** Traffic demand prediction targets at estimating the future number of traveling at some location. It is of vital and practical significance in the resource scheduling for ITS. By using GNNs, the spatial dependencies of demands can be revealed (Yang, Heppenstall, Turner, & Comber, 2020; Yao et al., 2018). What is more, urban vehicle emission analysis is also considered in recent work, which is closely related to environmental protection and gains increasing researcher attention (Xu, Kang, Cao, & Li, 2020).

14.2. Traffic graph construction

14.2.1. Traffic graph

The traffic network is represented as a graph $\mathcal{G} = (V, E, A)$, where V is the set of N traffic nodes, E is the set of edges, and $A \in \mathbb{R}^{N \times N}$ is an adjacency matrix representing the connectivity of N nodes. In the traffic domain, V usually represents a set of physical nodes, like traffic stations or traffic sensors. The features of nodes typically depend on the specific task. Take traffic flow forecasting as an example. The features are the traffic flows, i.e., the historical time series of nodes. The traffic flow can be represented as a flow matrix $X \in \mathbb{R}^{N \times T}$, where N is the number of traffic nodes and T is the length of historical series, and X_{nt} denotes the traffic flow of node n at time t. The goal of traffic flow forecasting is to learn a mapping function f to predict the traffic flow during future T' steps given the historical T step information, which can be formulated as follows:

$$\left[X_{:,t-T+1}, X_{:,t-T+2}, \ldots, X_{:,t}; \mathcal{G}\right] \xrightarrow{f} \left[X_{:,t+1}, X_{:,t+2}, \ldots, X_{:,t+T'}\right]. \quad (162)$$

14.2.2. Graph construction

Constructing a graph to describe the interactions among traffic nodes, i.e., the design of the adjacency matrix A, is the key part of traffic analysis. The mainstream designs can be divided into two categories, fixed matrix and dynamic matrix.

Fixed matrix. Lots of works assume that the correlations among traffic nodes are fixed and constant over time, and they design a fixed

Table 13
Summary of graph models for traffic analysis.

Models	Tasks	Adjacency matrices	GNN types	Temporal modules
STGCN (Yu, Yin, & Zhu, 2017)	Traffic flow forecasting	Fixed matrix	GCN	TCN
DCRNN (Li, Yu, Shahabi and Liu, 2017)	Traffic flow forecasting	Fixed matrix	ChebNet	RNN
AGCRN (Bai, Yao, Li, Wang, & Wang, 2020)	Traffic flow forecasting	Dynamic matrix	GCN	GRU
ASTGCN (Guo, Lin, Feng, Song, & Wan, 2019)	Traffic flow forecasting	Fixed matrix	GAT	Attention&TCN
GraphWaveNet (Wu, Pan, Long, Jiang and Zhang, 2019)	Traffic flow forecasting	Dynamic matrix	GCN	Gated-TCN
STSGCN (Song, Lin, Guo and Wan, 2020)	Traffic flow forecasting	Dynamic matrix	GCN	Cropping
LSGCN (Huang, Huang, Liu, Dai, & Kong, 2020)	Traffic flow forecasting	Fixed matrix	GAT	GLU
GAC-Net (Song, Ming, Hu, Niu and Gao, 2020)	Traffic flow forecasting	Fixed matrix	GAT	Gated-TCN
STGODE (Fang, Long, Song, & Xie, 2021)	Traffic flow forecasting	Fixed matrix	Graph ODE	TCN
STG-NCDE (Choi, Choi, Hwang, & Park, 2022)	Traffic flow forecasting	Dynamic matrix	GCN	NCDE
DDGCRN (Weng et al., 2023)	Traffic flow forecasting	Dynamic matrix	GAT	RNN
MS-ASTN (Wang, Miao, Chen and Huang, 2020)	OD flow forecasting	OD matrix	GCN	LSTM
Social-STGCNN (Mohamed et al., 2020)	Trajectory prediction	Fixed matrix	GCN	TXP-CNN
RSBG (Sun, Jiang et al., 2020)	Trajectory prediction	Dynamic matrix	GCN	LSTM
ATG (Yuan, Ding, Wang, Jin, & Li, 2022)	Trajectory prediction	Fixed matrix	GODE	NODE
STGAN (Deng et al., 2022)	Anomaly detection	Fixed matrix	GCN	GRU
DMVST-VGNN (Jin, Xi, Sha, Feng and Huang, 2020)	Traffic demand prediction	Fixed matrix	GAT	GLU
DST-GNN (Huang et al., 2022)	Traffic demand prediction	Dynamic matrix	GCN	Transformer
TC-SGC (Pan, Hou, & Li, 2022)	Traffic speed prediction	Fixed matrix	GCN	GRU

and pre-defined adjacency matrix to capture the spatial correlation. Here we list several common choices of fixed adjacency matrix.

The **connectivity matrix** is the most natural construction way. It relies on the support of road map data. The element of the connectivity matrix is defined as 1 if two nodes are physically connected and 0 otherwise. This binary format is convenient to construct and easy to interpret.

The **distance-based matrix** is also a common choice, which shows the connection between two nodes more precisely. The elements of the matrix are defined as the function of distance between two nodes (driving distance or geographical distance). A typical way is to use the threshold Gaussian function as follows,

$$A_{ij} = \begin{cases} \exp(-\frac{d_{ij}^2}{\sigma^2}), & d_{ij} < \epsilon \\ 0, & d_{ij} > \epsilon \end{cases}, \tag{163}$$

where d_{ij} is the distance between node i and j, and σ and ϵ are two hyperparameters to control the distribution and the sparsity of the matrix.

Another kind of fixed adjacency matrix is the **similarity-based matrix**. In fact, a similarity matrix is not an adjacency matrix to some extent. It is constructed according to the similarity of two nodes, which means the neighbors in the similarity graph may be far away in the real world. There are various similarity metrics. For example, many works measure the similarity of two nodes by their functionality, e.g., the distribution of surrounding points of interest (POIs). The assumption behind this is that nodes that share similar functionality may share similar traffic patterns. We can also define the similarity through the historical flow patterns. To compute the similarity of two-time series, a common practice is to use Dynamic Time Wrapping (DTW) algorithm (Müller, 2007), which is superior to other metrics due to its sensitivity to shape similarity rather than point-wise similarity. Specifically, given two time series $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$, DTW is a dynamic programming algorithm defined as

$$D(i, j) = dist(x_i, y_j) + \min(D(i-1, j), D(i, j-1), D(i-1, j-1)), \tag{164}$$

where $D(i, j)$ represents the shortest distance between subseries $X = (x_1, x_2, \ldots, x_i)$ and $Y = (y_1, y_2, \ldots, y_j)$, and $dist(x_i, y_j)$ is some distance metric like absolute distance. As a result, $DTW(X, Y) = D(n, n)$ is set as the final distance between X and Y, which better reflects the similarity of the two-time series compared to the Euclidean distance.

Dynamic matrix. The pre-defined matrix is sometimes unavailable and cannot reflect complete information of spatial correlations. The dynamic adaptive matrix is proposed to solve the issue. The dynamic matrix is learned from input data automatically. To achieve the best prediction performance, the dynamic matrix will manage to infer the hidden correlations among nodes, more than those physical connections.

A typical practice is learning adjacency matrix from node embeddings (Bai et al., 2020). Let $E_A \in \mathbb{R}^{N \times d}$ be a learnable node embedding dictionary, where each row of E_A represents the embedding of a node, N and d denote the number of nodes and the dimension of embeddings respectively. The graph adjacency matrix is defined as the similarities among node embeddings,

$$D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = softmax\left(ReLU(E_A \cdot E_A^T)\right), \tag{165}$$

where $softmax$ function is to perform row-normalization, and $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the Laplacian matrix.

14.3. Typical GNN frameworks in traffic domain

Spatial Temporal Graph Convolution Network (STGCN) (Yu et al., 2017). STGCN is a pioneering work in the spatial–temporal GNN domain. It utilizes graph convolution to capture spatial features, and deploys a gated causal convolution to extract temporal patterns. Specifically, the graph convolution and temporal convolution are defined as follows,

$$\Theta *_\mathcal{G} x = \theta(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x = \theta(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}})x, \tag{166}$$

$$\Gamma *_\mathcal{T} y = P \odot \sigma(Q), \tag{167}$$

where Θ is the parameter of graph convolution, P and Q are the outputs of a 1-d convolution along the temporal dimension. The sigmoid gate $\sigma(Q)$ controls how the states of P are relevant for discovering hidden temporal patterns. In order to fuse features from both spatial and temporal dimension, the spatial convolution layer and the temporal convolution layer are combined to construct a spatial temporal block to jointly deal with graph-structured time series, and more blocks can be stacked to achieve a more scalable and complex model.

Diffusion Convolutional Recurrent Neural Network (DCRNN) (Li, Yu et al., 2017). DCRNN is a representative solution combining graph convolution networks with recurrent neural networks. It captures spatial dependencies by bidirectional random walks on the graph. The diffusion convolution operation on a graph is defined as:

$$X *_\mathcal{G} f_\theta = \sum_{k=0}^{K} \left(\theta_{k,1}(D_O^{-1} A)^k + \theta_{k,2}(D_I^{-1} A)^k\right) X, \tag{168}$$

where θ are parameters for the convolution filter, and $D_O^{-1} A, D_I^{-1} A$ represent the bidirectional diffusion processes respectively. In term of temporal dependency, DCRNN utilizes Gated Recurrent Units (GRU),

and replace the linear transformation in the GRU with the diffusion convolution as follows,

$$r^{(t)} = \sigma(\Theta_r *_{\mathcal{G}} [X^{(t)}, H^{(t-1)}] + b_r), \tag{169}$$

$$u^{(t)} = \sigma(\Theta_u *_{\mathcal{G}} [X^{(t)}, H^{(t-1)}] + b_u), \tag{170}$$

$$C^{(t)} = \tanh(\Theta_C *_{\mathcal{G}} [X^{(t)}, (r^{(t)} \odot H^{(t-1)}] + b_c)), \tag{171}$$

$$H^{(t)} = u^{(t)} \odot H^{(t-1)} + (1 - u^{(t)}) \odot C^{(t)}, \tag{172}$$

where $X^{(t)}, H^{(t)}$ denote the input and output at time t, $r^{(t)}, u^{(t)}$ are the reset and update gates respectively, and $\Theta_r, \Theta_u, \Theta_C$ are parameters of convolution filters. Moreover, DCRNN employs a sequence-to-sequence architecture to predict future series. Both the encoder and the decoder are constructed with diffusion convolutional recurrent layers. The historical time series are fed into the encoder and the predictions are generated by the decoder. The scheduled sampling technique is utilized to solve the discrepancy problem between training and test distribution.

Adaptive Graph Convolutional Recurrent Network (AGCRN) (Bai et al., 2020). The focuses of AGCRN are two-fold. On the one hand, it argues that the temporal patterns are diversified and thus parameter-sharing for each node is inferior; on the other hand, it proposes that the pre-defined graph may be intuitive and incomplete for the specific prediction task. To mitigate the two issues, it designs a Node Adaptive Parameter Learning (NAPL) module to learn node-specific patterns for each traffic series, and a Data Adaptive Graph Generation (DAGG) module to infer the hidden correlations among nodes from data and to generate the graph during training. Specifically, the NAPL module is defined as follows,

$$Z = (I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X E_{\mathcal{G}} W_{\mathcal{G}} + E_{\mathcal{G}} b_{\mathcal{G}}, \tag{173}$$

where $X \in \mathbb{R}^{N \times C}$ is the input feature, $E_{\mathcal{G}} \in \mathbb{R}^{N \times d}$ is a node embedding dictionary, d is the embedding dimension ($d \ll N$), $W_{\mathcal{G}} \in \mathbb{R}^{d \times C \times F}$ is a weight pool. The original parameter Θ in the graph convolution is replaced by the matrix production of $E_{\mathcal{G}} W_{\mathcal{G}}$, and the same operation is applied for the bias. This can help the model to capture node-specific patterns from a pattern pool according to the node embedding. The DAGG module has been introduced in (165). The whole workflow of AGCRN is formulated as follows,

$$\tilde{A} = softmax(ReLU(EE^T)), \tag{174}$$

$$z^{(t)} = \sigma(\tilde{A}[X^{(t)}, H^{(t-1)}]EW_z + Eb_z), \tag{175}$$

$$r^{(t)} = \sigma(\tilde{A}[X^{(t)}, H^{(t-1)}]EW_r + Eb_r), \tag{176}$$

$$\hat{H}^{(t)} = \tanh(\tilde{A}[X, r^{(t)} \odot H^{(t-1)}]EW_h + Eb_h), \tag{177}$$

$$H^{(t)} = z^{(t)} \odot H^{(t-1)} + (1 - z^{(t)}) \odot \hat{H}^{(t)}. \tag{178}$$

Attention Based Spatial-Temporal Graph Convolutional Networks (AST-GCN) (Guo et al., 2019). ASTGCN introduces two kinds of attention mechanisms into spatial–temporal forecasting, i.e., spatial attention and temporal attention. Spatial attention is defined as the following,

$$S = V_S \sigma \left((XW_1)W_2(W_3 X)^T + b_S \right), \tag{179}$$

$$S'_{i,j} = \frac{\exp(S_{i,j})}{\sum_{j=1}^{N} \exp(S_{i,j})}, \tag{180}$$

where S' is the attention score, and $W_1, W_2, W3$ are learnable parameters. A similar construction is applied for temporal attention. Besides the attention mechanism, ASTGCN also introduces multi-component fusion to enhance the prediction ability. The input of ASTGCN consists of three parts, the recent segments, the daily-periodic segments and the weekly-periodic segment. The three segments are processed by the main model independently and fused with learnable weights at last:

$$Y = W_h \odot Y_h + W_d \odot Y_d + W_w \odot Y_w, \tag{181}$$

where Y_h, Y_d, Y_w denotes the predictions of different segments respectively.

15. Summary

This section introduces graph models for traffic analysis and we provide a summary as follows:

- **Techniques.** Traffic analysis is a classical spatial temporal data mining task, and graph models play a vital role for extracting spatial correlations. Typical procedures include graph construction, spatial dimension operations, temporal dimension operations and information fusion. There are multiple implementations for each procedure, each implementation has its strengths and weaknesses. By combining different implementations, various kinds of traffic analysis models can be created. Choosing the right combination of procedures and implementations is critical for achieving accurate and reliable traffic analysis results.

- **Challenges and Limitations.** Despite the remarkable success of graph representation learning in traffic analysis, there are still several challenges that need to be addressed in current studies. Firstly, external data, such as weather and calendar information, are not well-utilized in current models, despite their close relation to traffic status. The challenge lies in how to effectively fuse heterogeneous data to improve traffic analysis accuracy. Secondly, the interpretability of models has been underexplored, which could hinder their deployment in real-world transportation systems. Interpretable models are crucial for building trust and understanding among stakeholders, and more research is needed to develop models that are both accurate and interpretable. Addressing these challenges will be critical for advancing the state-of-the-art in traffic analysis and ensuring the deployment of effective transportation systems.

- **Future Works.** In the future, we anticipate that more data sources will be available for traffic analysis, enabling a more comprehensive understanding of real-world traffic scenes. From data collection to model design, there is still a lot of work to be done to fully leverage the potential of GNNs in traffic analysis. In addition, we expect to see more creative applications of GNN-based traffic analysis, such as designing traffic light control strategies, which can help to improve the efficiency and safety of transportation systems. To achieve these goals, it is necessary to continue advancing the development of GNN-based models and exploring new ways to fuse diverse data sources. Additionally, there is a need to enhance the interpretability of models and ensure their applicability to real-world transportation systems. We believe that these efforts will contribute to the continued success of traffic analysis and the development of intelligent transportation systems.

16. Discussion

In this section, we systematically compare the strengths and weaknesses of different components within both GNN architectures and learning paradigms. This comprehensive exploration aims to shed light on how these components can be harnessed to better serve the field of deep graph representation learning.

16.1. GNN architectures

Here, we summarize the advantages and disadvantages of different components in GNN architectures in Table 14.

Graph Convolutions

- **Advantages:** (i) *Local Information Aggregation:* Graph convolutions excel at capturing intricate local neighborhood information, making them highly effective for tasks where node representations heavily depend on nearby nodes. (ii) *Parameter Sharing:*

Table 14

Comparison of different components in GNN architectures.

Method	Advantages	Disadvantages
Graph convolutions	Local information aggregation, parameter sharing	Limited global context, sensitivity to graph structure
Graph kernel neural networks	Kernel trick benefits, embedding similarity	Computational complexity, limited scalability
Graph pooling	Hierarchical representation, dimensionality reduction	Information loss, pooling strategy sensitivity
Graph transformer	Attention mechanism, parallelization	Computational intensity, limited interpretability

Shared weights in graph convolutions enable the model to generalize robustly across different regions of the graph, enhancing efficiency.

- **Disadvantages:** (i) *Limited Global Context:* One limitation lies in the potential struggle to capture long-range dependencies in graphs, leading to information loss on more distant nodes. (ii) *Sensitivity to Graph Structure:* The performance of graph convolutions can be influenced by irregularities in graph structures, impacting their adaptability.

Graph Kernel Neural Networks

- **Advantages:** (i) *Kernel Trick Benefits:* Leveraging kernel methods, graph kernel neural networks effectively learn on structured data, providing flexibility in capturing complex relationships. (ii) *Embedding Similarity:* These networks encode similarity measures between nodes, capturing nuanced graph structure information crucial for certain applications.
- **Disadvantages:** (i) *Computational Complexity:* Graph kernel methods may face challenges related to computational expense, particularly when dealing with large graphs. (ii) *Limited Scalability:* Scalability issues arise when extending these methods to graphs of varying sizes and structures.

Graph Pooling

- **Advantages:** (i) *Hierarchical Representation:* Graph pooling contributes to creating hierarchical representations, enabling the model to capture information at multiple levels of granularity, fostering a more comprehensive understanding. (ii) *Dimensionality Reduction:* Effectively reduces computational load by downsampling the graph, making it more computationally tractable.
- **Disadvantages:** (i) *Information Loss:* Aggregating information during pooling may lead to a loss of fine-grained details, potentially impacting the model's performance on certain downstream tasks. (ii) *Pooling Strategy Sensitivity:* The choice of pooling strategy may significantly affect performance, emphasizing the importance of thoughtful strategy selection.

Graph Transformer

- **Advantages:** (i) *Attention Mechanism:* Graph transformers leverage attention mechanisms to capture global dependencies and relationships effectively, enabling the model to consider the entire graph when making predictions. (ii) *Parallelization:* The attention mechanism allows for parallelization of computations, enhancing computational efficiency, particularly on hardware optimized for parallel processing.
- **Disadvantages:** (i) *Computational Intensity:* Transformers can be computationally intensive, especially with large graphs, requiring careful consideration of resource constraints. (ii) *Limited Interpretability:* The attention mechanism, while powerful, may lack interpretability, making it challenging to understand the model's decision-making process.

16.2. Learning paradigms

Here, we summarize the advantages and disadvantages of different components in learning paradigms in Table 15.

Supervised/Semi-Supervised Learning on Graphs

- **Advantages:** (i) *Utilization of Labeled Data:* Supervised learning effectively utilizes labeled data for direct prediction, while semi-supervised learning leverages both labeled and unlabeled data, providing a flexible approach. (ii) *Task-Specific Objectives:* Clear task objectives guide the learning process, allowing for precise model training.
- **Disadvantages:** (i) *Data Dependency:* These paradigms heavily rely on the availability of labeled data, potentially limiting their applicability in scenarios with sparse labels. (ii) *Generalization Challenges:* Generalizing to unseen nodes or graphs can be challenging, particularly in dynamic or evolving graph structures.

Graph Self-Supervised Learning

- **Advantages:** (i) *Data Efficiency:* Utilizes unlabeled data for pre-training, making efficient use of available resources and potentially reducing the need for extensive labeled datasets. (ii) *Transferability:* Pre-trained models can be fine-tuned for downstream tasks, enhancing generalization across various applications.
- **Disadvantages:** (i) *Pretext Task Design:* The effectiveness of self-supervised learning depends on the design of pretext tasks, requiring careful consideration of task relevance. (ii) *Computationally Intensive Pre-training:* The pre-training phase can be computationally intensive, especially when dealing with large and complex graphs.

Graph Structure Learning

- **Advantages:** (i) *Incorporation of Inherent Graph Properties:* Focuses on exploiting the intrinsic structure of graphs, enhancing representation learning by considering the unique properties of graph data. (ii) *Robustness to Label Sparsity:* Less dependent on labeled data, making it suitable for scenarios with limited labels, and potentially more robust in the face of sparse data.
- **Disadvantages:** (i) *Limited Task Specificity:* May not be inherently task-specific and might not outperform task-tailored methods on certain applications that demand specialized representations. (ii) *Sensitivity to Graph Noises:* Performance may degrade in the presence of noisy or irregular graph structures, necessitating preprocessing steps for noise reduction.

17. Future directions

In this section, we outline some prospective future directions of deep graph representation learning based on the above cornerstone, taxonomy, and real-world applications. We also outline a few more directions closer to the theoretical side.

17.1. Application-inspired directions

Since deep graph representation has been widely used these years, many problems have been solved while many others have arisen. While we observe many real-world applications, we conclude plenty of challenging problems that are not yet solved. Here in this subsection, we outline a few.

17.1.1. Fairness in graph representation learning

One common aspect to care about is the fairness concern. Fairness, by definition, refers to the protected features that do not infect the outcome. In general, a data set's fairness refers to that the protected features are not influencing the data distribution. A model's fairness, on the other hand, refers to the concern that the output of our algorithms

Table 15
Comparison of different components in learning paradigms.

Method	Advantages	Disadvantages
Supervised/Semi-supervised learning on graphs	Utilization of labeled data, task-specific objectives	Data dependency, generalization challenges
Graph self-supervised learning	Data efficiency, transferability	Pretext task design, computationally intensive pre-training
Graph structure learning	Incorporation of inherent graph properties, robustness to label sparsity	Limited task specificity, sensitivity to graph noises

should not be affected by certain protected features. The protected features can be race, gender, etc.

Similar to the fairness challenge in many other fields of machine learning (Chouldechova & Roth, 2018; Mehrabi, Morstatter, Saxena, Lerman, & Galstyan, 2021), graph representation learning can easily suffer from bias from the data sets that inherit stereotypes from the real world. As graph representation has become increasingly popular in recent years, researchers are getting fairness into their sights (Dong, Kang, Tong, & Li, 2021; Ma, Deng, & Mei, 2021).

Different from i.i.d. data, graph data contains a lot of relational information that would bring about new challenges to bias detection (Dai & Wang, 2021; Dong, Liu, Jalaian, & Li, 2022). Besides, another challenge is that better algorithms are needed to prevent the model from inheriting the input data's biases (Dong et al., 2022; Ma et al., 2021). Researchers have been working on various fairness notions regarding graph-representation learning. Such as group fairness, individual fairness, and application-specific fairness (Dong, Ma, Wang, Chen, & Li, 2023). Application-specific fairness mostly refers to that in the field of recommender systems (Fu et al., 2020; Li, Chen, Fu, Ge and Zhang, 2021) and knowledge graphs (Fisher, Palfrey, Christodoulopoulos, & Mittal, 2019).

Fairness concern in graph mining is crucial in real-world applications. Researchers have already found significant gender and ethnic group bias in existing recommender systems (Lambrecht & Tucker, 2019; Sweeney, 2013). Not to mention more sensitive domains of applications, such as loan approval and criminal justice (Agarwal, Lakkaraju, & Zitnik, 2021; Sarkar & Alvari, 2020).

17.1.2. Robustness in graph representation learning

Real-world data is always noisy, containing many different kinds of disruptions, and does not end up being a perfectly normal distribution. In the worst case, some noise can potentially prevent a model from learning the correct knowledge. Better robustness refers to the model having a better chance of reaching a relatively good and stable outcome while input is being manipulated. If a model is not robust enough, the performance cannot be relied on. Therefore, robustness is another important yet challenging consideration in deep graph representation learning. It is especially true when given that researchers have already found that existing graph representation learning models are vulnerable to adversarial data samples in general (Bojchevski & Günnemann, 2019; Chen et al., 2020; Wang et al., 2022; Yuan et al., 2023, 2023; Zügner & Günnemann, 2020).

Again, similar to many other machine learning approaches that aim at solving real-world problems (Carlini & Wagner, 2017), improving the robustness of deep graph representation models is a nontrivial direction. Either enhancing the models' robustness or conducting adversarial attacks to challenge the robustness of graph representations, are promising directions to go for Geisler et al. (2021), Günnemann (2022) and Tang et al. (2020).

There have been studies on enhancing graph neural networks' robustness, from different perspectives (Chen, Li et al., 2020; Xu et al., 2021). There they introduced some metrics to measure the robustness of graph models, such as classification margin, and adversarial gap (Xu, Chen et al., 2021).

One of the most common solutions to enhance robustness is to avoid using malicious samples. Starting from the preprocessing steps, some

researchers have already studied how to get rid of the poisoned data. For example, Xu, Wang, Lal, Gunter and Li (2023) worked on sampling sub-graphs from the poisoned training data set and then detected outliers so that they could filter the adversarial edges. In particular, for example, since some attacker models are known as adding high-rank perturbations to data, in order to get rid of those attacks, Entezari, Al-Sayouri, Darvishzadeh, and Papalexakis (2020) "vaccinate" data sets in pre-processing steps, by using low-rank approximation matrices. Some other works have a more general measurement of which edges to eliminate, such as using the Jaccard Similarity score to measure an edge's end nodes and remove the suspicious edges whose score is not high enough (Wu et al., 2019).

Moving one step ahead, it is natural to consider conducting anomaly detection. There has already been a lot of anomaly detection work done on static graphs (Jiang et al., 2019; Wang et al., 2021). These works are of great importance since many traditional anomaly detection methods fail to take link connections into consideration and behave poorly on graph data. Nonetheless, anomaly detection works on dynamic graphs are much fewer (Du, Li, Zheng, & Srikumar, 2017). Robustness on dynamic graphs has been of increasing importance in recent years (Arp et al., 2022) and is also worth being studied.

However, despite the efforts to eliminate malicious samples at an early stage, adversarial attacks can still easily reach the training stage. There we need more careful design of the model to make it robust when facing poisoned data directly on its own. Some researchers are focusing on adversarial training, by either utilizing adversarial mini-max objectives (Jin et al., 2021; Li, Jin, Xu and Tang, 2020), or adding adversarial samples intentionally so as to learn how to deal with them Deng, Dong, and Zhu (2019), Yang, Liang and Zhang (2022) and Zhang, Jia, Wang and Gong (2021). Some other researchers also propose to relieve the threat of poisoned data by using a variance-based attention mechanism, under the assumption that fake links and nodes should have higher uncertainty in prediction results (Zhu, Zhang, Cui, & Zhu, 2019).

17.1.3. Adversarial reprogramming

With the emergence of pre-trained graph neural network models (Hu, Dong, Wang, Chang and Sun, 2020; Hu et al., 2019; Qiu, Chen et al., 2020), introducing adversarial reprogramming (Elsayed, Goodfellow, & Sohl-Dickstein, 2018; Zheng et al., 2021) into deep graph representation learning becomes another possibility as well. The major difference between adversarial reprogramming and adversarial attack (Huang, Papernot, Goodfellow, Duan, & Abbeel, 2017; Thys, Van Ranst, & Goedemé, 2019; Zhao, Yan and Wei, 2020) lies in whether or not there is a particular target after putting some adversarial samples against the model. An adversarial attack requires some small modifications to the input data samples. An adversarial attack is considered successful once the result is influenced. However, under the adversarial reprogramming settings, the task succeeds if and only if the influenced results can be used for another desired task.

This is to say, without changing much on the model's inner structure or fine-tuning its parameters, we might be able to use some pre-trained graph models for some other tasks that were not planned to be solved by these models in the first place. In other deep learning fields, adversarial reprogramming problems are normally solved by having

the input carefully encoded, and output cleverly mapped. On some graph data sets, such as chemical data sets and biology data sets, pre-trained models are already available. Therefore, there is a possibility that adversarial reprogramming could be applied in the future.

17.1.4. Generalizing to out of distribution data

In order to perform better on unobserved data sets, in the ideal case, the representation we learn should better be able to generalize to some out-of-distribution (OOD) data. Being out-of-distribution is not identical to being misclassified. The misclassified samples are coming from the same distribution of the training data but the model fails to classify it correctly, while out-of-distribution refers to the case where the sample comes from a distribution other than the training data (Hendrycks & Gimpel, 2016; Luo, Zhao, Qin, Ju and Zhang, 2023). Being able to generalize to out-of-distribution data will greatly enhance a model's reliability in real life. And studying out-of-distribution generalized graph representation (Li, Wang, Zhang, & Zhu, 2022a) is an opening field (Li, Wang, Zhang, & Zhu, 2022b). This is partly because of, currently, even the problem of detecting out-of-distribution data samples is not fully conquered yet (Hendrycks & Gimpel, 2016).

In order to do something on the out-of-distribution data samples, we need to detect which samples belong to this type first. Detecting OOD samples itself is somewhat similar to novelty detection, or outlier detection problems (Pimentel, Clifton, Clifton, & Tarassenko, 2014). Their major difference is whether or not a well-performed model conducting the original tasks remains part of our goal. Novelty detection cares only about figuring out who are the outliers; OOD detection requires our model to detect the outliers while keeping the performance unharmed at the same time.

17.1.5. Interpretability in graph representation learning

Interpretability concern is another limitation that exists when researchers try to apply deep graph representation learning to some of the emerging application fields. For instance, in the field of computational social science, researchers are urging more efforts in integrating explanation and prediction together (Hofman et al., 2021). So as drug discovery, being able to explain why such a structure is chosen instead of another option, is very important (Jiménez-Luna, Grisoni, & Schneider, 2020). Generally speaking, neural networks are completely in black-box mode to human knowledge without making efforts to make them interpretable and explainable. Although more and more tasks are being handled by deep learning methods in many fields, the tool remains mysterious to most human beings. Even an expert in deep learning cannot easily explain to you how the tasks are performed and what the model has learned from the data. This situation reduces the trustworthiness of the neural network models, prevents a human from learning more from the models' results, and even limits the potential improvements of the models themselves, without sufficient feedback to human beings.

Seeking for better interpretability is not only some personal interests of companies and researchers, in fact, as more and more ethical concerns arose since more and more black-box decisions were made by AI algorithms, interpretability has become a legal requirement (Goodman & Flaxman, 2017).

Various approaches have been applied, serving the goal of better interpretability (Zhang, Tiňo, Leonardis and Tang, 2021). There we find existing works that provide either ad-hoc explanations after the results come out, or those actively change the model structure to provide better explanations; explanations by providing similar examples, highlighting some attributes to the input features, by making sense of some hidden layers and extract semantics from them, or by extracting logical rules; we also see local explanations that explain some particular samples, global explanations that explain the network as a whole, or hybrid. Most of those existing directions make sense in a graph representation learning setting.

Not a consensus has been reached on what are the best methods of making a model interpretable. Researchers are still actively exploring every possibility, and thus there are plenty of challenges and interesting topics in this direction.

17.1.6. Causality in graph representation learning

Most of the existing studies (Wu et al., 2020; Zhou, Cui et al., 2020) on graph representation learning is good at handling predictive (i.e. node classification, graph classification, link prediction) or descriptive (e.g. shortest path, centrality) tasks on static graphs. However, graph data, such as social networks, and citation networks, can change over time. To analyze their dynamic features, there is an emerging need to analyze causality on graphs. That is to say, apply some treatment to the system, and predict the following outcome.

In recent years, there have been increasing research works focusing on combining causality and machine learning models (Hu & Li, 2021; Luo et al., 2023; Madumal, Miller, Sonenberg, & Vetere, 2020; Richens, Lee, & Johri, 2020). It is widely believed that making good use of causality will help models gain higher performances. However, finding the right way to model causality in many real-world scenarios remains challenging. In real-world data sets, there usually exist unobserved confounders (i.e. the variables that affect both treatment assignment and outcome). To mitigate the confounder bias, most of the causality models, such as TARNet (treatment-agnostic representation network) which learns the representation of confounders, and CFR (counterfactual regression) which uses representation balancing techniques to minimize the distribution distance between the confounders' representations of the treatment and controlled groups (Shalit, Johansson, & Sontag, 2017), are built upon strong ignorability assumption, meaning that there is no other confounder except for those who are already included in the observed features (Ma & Li, 2022). On the other hand, models such as CEVAE (causal effect variational autoencoder) did not rely on the strong ignorability assumption. Instead, it assumes that it can infer the hidden confounders from the observed features (Louizos et al., 2017).

In particular, graph data is especially challenging for causal inference, due to the following few reasons (Ma & Li, 2022): (1) multimodality of the data; (2) non-neglectable hidden confounders; (3) complicated forms e.g. dynamic graphs; (4) network inference among individuals that no longer fulfill the assumptions of many traditional causal effect models; (5) graph-structured treatment.

Something to note is that the most common kind of graph that comes along the causal study, called "causal graph", is not necessarily identical to the kind of graphs we are studying in deep graph representation learning. Causal graphs are the kind of graphs whose nodes are factors and links represent causal relations. Up till now, they are among the most reliable tools for causal inference study. Traditionally, causal graphs are defined by human experts. Recent works have shown that neural networks can help with scalable causal graph generation (Xu, Huang and Yoo, 2019). From this perspective, the story can be the other side around: besides using causal relations to enhance graph representation learning, it is also possible to use graph representation learning strategies to help with causal study. For example, a knowledge graph can be used to diagnose the root cause of performance anomaly in cloud applications (Qiu, Du, Yin, Zhang and Qian, 2020). Many researchers have already worked on mining the causality in graph-structured data and found this direction appealing (Guo, Li, & Liu, 2020; Ma, Dong, Huang, Mietchen, & Li, 2022).

17.1.7. Emerging application fields

Besides the above-mentioned directions solving existing challenges in the deep learning world, there are many emerging fields of application that naturally come along with graph-structured data.

For instance, the emerging fields of social network analysis that help with traditional social science or political science studies, and drug discovery that helps with medical science. Due to the nature of

the data, such as the social network interactions (i.e. follow, retweet, reply), and drug molecule structures (i.e. atoms and bounds in between them), can be easily depicted as graph-structured data. Therefore, deep graph representation learning has much to do in these fields (Abbas, 2021; Gaudelet et al., 2021; Yang et al., 2023; Zhu, 2022).

Some basic problems on the social network are easily solved using graph representation learning strategies. Those basic problems include node classification, link prediction, graph classification, and so on. In practice, those problem settings could refer to real-world problems such as ideology prediction, interaction prediction, analyzing a social group, etc. However, social network data typically has many unique features that could potentially stop the general-purposed models from performing well. For instance, social media data can be sparse, incomplete, and can be extremely imbalanced (Zhao, Zhang and Wang, 2021). On the one hand, social media platforms themselves never have a clear and consistent topic. On the other hand, people have clear goals when studying social media data, such as controversy detection (Benslimane, Azé, Bringay, Servajean, & Mollevi, 2022), rumor detection (Hamidian & Diab, 2019; Takahashi & Igata, 2012), misinformation and disinformation detection (Di Domenico, Sit, Ishizaka, & Nunan, 2021), or studying the dynamics of the system (Kipf, Fetaya, Wang, Welling, & Zemel, 2018; Luo et al., 2023, 2023). The real-world social media data is naturally unlabeled, making even data annotation (i.e. labeling) itself a challenging task to deal with. There are still a lot of open quests to be conquered, which deep graph representation learning can help with.

As for drug discovery, researchers have some interest in other perspectives beyond simply proposing a set of potentially functional structures, which is widely seen today. The other perspectives include having more interpretable results from the model's proposals (Jiménez-Luna et al., 2020; Preuer, Klambauer, Rippmann, Hochreiter, & Unterthiner, 2019), and considering synthetic accessibility (Xie et al., 2021). These directions are important, in answer to some doubt on AI from the society (Goodman & Flaxman, 2017), as well as from the tradition of chemistry studies (Schneider et al., 2020). Similar to the challenges we faced when combining social science and neural networks, chemical science would also prefer the black-box AI models to be interpretable instead. Some chemical scientists would also prefer AI tools to provide them with synthetic routes instead of the targeting structure itself. In practice, proposing new molecule structures is usually not the bottleneck, but synthesizing is. There are already some existing works focusing on conquering this problem (Empel & Koenigs, 2019; Ishida, Terayama, Kojima, Takasu, & Okuno, 2022). But so far there is a gap between chemical experiments and AI tools, indicating that there is still plenty of improvement to be made.

Some chemistry researchers also found it useful to have material data better organized, given that the molecule structures are becoming increasingly complex, and a massive amount of research papers are describing the material's features from different aspects (Walsh et al., 2023). This direction might be more closely related to knowledge base or even database systems. But in a way, given that the polymer structure is typically a node-link graph, graph representation learning might be able to help with dealing with such issues.

Besides, we also realize that most of the machine-generated molecule structures will be regarded as useless from a chemical scientist's view. The root cause is that, from a human expert's perspective, some properties (i.e. features) of the molecules are obvious, but to a machine learning model, it is not. A well-trained human expert will realize that some molecule structures are not stable under room temperature and standard atmosphere pressure. But according to the existing benchmarks and the available labeled data sets, a machine learning model could regard that particular structure as the best option.

In a way, we believer that, as more and more wet experiments are involved, researchers will gradually realize the importance of combining expertise knowledge with the graph representation learning

models. It will enhance the interpretability and trustworthiness of many applications.

17.2. Theory-driven directions

Some other future directions dig into the root of graph theory. More specifically, it focuses on some fundamental improvements in neural network structure design, or better ways of expressing the graph representations. These directions require background knowledge of their mathematical backgrounds. All in all, breakthroughs in these directions might not end up with immediate impact, but every study in these directions has the potential to change the entire field, sooner or later.

17.2.1. Mathematical proof of feasibility

It has been a long-lasting problem that most of the existing deep learning approaches lack mathematical proof of their learnability, bound, etc (Bartlett, Foster, & Telgarsky, 2017; Bouzerdoum & Pattison, 1993; Liu et al., 2023). This problem relates to the difficulty of providing theoretical proof on a complicated structure like neural network (Grohs & Voigtlaender, 2021).

Currently, most of the theoretical proof aims at figuring out theoretical bounds (Bartlett, Harvey, Liaw, & Mehrabian, 2019; Harvey, Liaw, & Mehrabian, 2017; Karpinski & Macintyre, 1997). There are multiple types of bounds with different problem settings. Such as: given a known architecture of the model, with input data satisfying particular normal distribution, prove that training will converge, and provide the estimated number of iterations. Most of these architectures being studied are simple, such as those made of multi-layer perceptron (MLP), or simply studying the updates of parameters in a single fully-connected layer.

In the field of deep graph representation learning, neural network architectures are typically much more complex than MLPs. Graph neural networks (GNNs), since the very beginning (Defferrard et al., 2016; Kipf & Welling, 2016a), involve a lot of approximation and simplification of mathematical theorems. Nowadays, most researchers rely heavily on the experimental results. No matter how wild an idea is, as long as it finally works out in an experiment, say, being able to converge and the results are acceptable, the design is acceptable. All these practices make the entire field somewhat experiments-oriented or experience-oriented, while there remains a huge gap between the theoretical proof and the frontier of deep graph representation.

It will be more than beneficial to the whole field if some researchers can push forward these theoretical foundations. However, these problems are incredibly challenging.

17.2.2. Combining spectral graph theory

Down to the theory foundations, the idea of graph neural networks (Defferrard et al., 2016; Kipf & Welling, 2016a; Shuman et al., 2013) initially comes from spectral graph theory (Chung, 1997). In recent years, many researchers have investigated possible improvements in graph representation learning strategies via utilizing spectral graph theory (Chen et al., 2020; He et al., 2022; MansourLakouraj, Gautam, Livani, & Benidris, 2022; Yang, Zhou, & Liu, 2021). For example, graph Laplacian is closely related to many properties, such as the connectivity of a graph. By studying the properties of Laplacian, it is possible to provide proof of graph neural network models' properties and to propose better models with desired advantages, such as robustness (Fu, Zhao, & Bian, 2022; Runwal, Kumar, et al., 2022).

Spectral graph theory provides a lot of useful insights into graph representation learning from a new perspective. There is a lot to be done in this direction.

17.2.3. From graph to manifolds

Many researchers are devoted to the direction of learning graph representation in non-Euclidean spaces (Asif et al., 2021; Saxena, Liu, & King, 2020). That is to say, to embed and compute on some other spaces that are not Euclidean, such as hyperbolic and spherical spaces.

Theoretical reasoning and experimental results have shown certain advantages of working on manifolds instead of standard Euclidean space. It is believed that these advantages are brought by their abilities to capture complex correlations on the surface manifold (Zhou et al., 2022). Besides, researchers have shown that, by combining standard graph representation learning strategies and manifold assumptions, models work better on preserving and acquiring the locality and similarity relationships (Fu & Liu, 2021). Intuitively, sometimes two nodes' embeddings are regarded as way too similar in Euclidean space, but in non-Euclidean space, they are easily distinguishable.

18. Conclusion

In this survey, we present a comprehensive and up-to-date overview of deep graph representation learning. We present a novel taxonomy of existing algorithms categorized into GNN architectures, learning paradigms, and applications. Technically, we first summarize the ways of GNN architectures namely graph convolutions, graph kernel neural networks, graph pooling, and graph transformer. Based on the different training objectives, we present three types of the most recent advanced learning paradigms namely: supervised/semi-supervised learning on graphs, graph self-supervised learning, and graph structure learning. Then, we provide several promising applications to demonstrate the effectiveness of deep graph representation learning. Last but not least, we discuss the future directions in deep graph representation learning that have potential opportunities.

CRediT authorship contribution statement

Wei Ju: Conceptualization, Funding acquisition, Methodology, Writing – original draft. **Zheng Fang:** Methodology, Writing – original draft. **Yiyang Gu:** Methodology, Writing – original draft. **Zequn Liu:** Methodology, Writing – original draft. **Qingqing Long:** Methodology, Writing – original draft. **Ziyue Qiao:** Methodology, Writing – original draft. **Yifang Qin:** Methodology, Writing – original draft. **Jianhao Shen:** Methodology, Writing – original draft. **Fang Sun:** Methodology, Writing – original draft. **Zhiping Xiao:** Methodology, Writing – original draft. **Junwei Yang:** Methodology, Writing – original draft. **Jingyang Yuan:** Methodology, Writing – original draft. **Yusheng Zhao:** Methodology, Writing – original draft. **Yifan Wang:** Methodology, Writing – original draft. **Xiao Luo:** Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Ming Zhang:** Conceptualization, Funding acquisition, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors are grateful to the anonymous reviewers for critically reading the manuscript and for giving important suggestions to improve their paper. All authors contributed equally to this research.

References

Abbas, A. M. (2021). Social network analysis using deep learning: applications and schemes. *Social Network Analysis and Mining, 11*, 1–21.

Agarwal, C., Lakkaraju, H., & Zitnik, M. (2021). Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in artificial intelligence* (pp. 2114–2124). PMLR.

Ahmad, W., Tayara, H., & Chong, K. T. (2023). Attention-based graph neural network for molecular solubility prediction. *ACS Omega, 8*, 3236–3244.

Ahmed, I., Galoppo, T., Hu, X., & Ding, Y. (2021). Graph regularized autoencoder and its application in unsupervised anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*, 4110–4124.

Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., & Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 37–48).

Al-Rfou, R., Perozzi, B., & Zelle, D. (2019). Ddgk: Learning graph representations for deep divergence graph kernels. In *The world wide web conference* (pp. 37–48).

AlBadani, B., Shi, R., Dong, J., Al-Sabri, R., & Moctard, O. B. (2022). Transformer-based graph convolutional network for sentiment analysis. *Applied Sciences, 12*, 1316.

Alon, U., & Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. arXiv preprint arXiv:2006.05205.

Altae-Tran, H., Ramsundar, B., Pappu, A. S., & Pande, V. (2017). Low data drug discovery with one-shot learning. *ACS Central Science, 3*, 283–293.

Anderson, B., Hy, T. S., & Kondor, R. (2019). Cormorant: Covariant molecular neural networks. *Vol. 32*, In *NeurIPS*. URL: https://proceedings.neurips.cc/paper/2019/file/03573b32b2746e6e8ca98b9123f2249b-Paper.pdf.

Anderson, W. N., Jr., & Morley, T. D. (1985). Eigenvalues of the laplacian of a graph. *Linear and Multilinear Algebra, 18*, 141–145.

Aouichaoui, A. R., Fan, F., Mansouri, S. S., Abildskov, J., & Sin, G. (2023). Combining group-contribution concept and graph neural networks toward interpretable molecular property models. *Journal of Chemical Information and Modeling, 63*, 725–744.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning* (pp. 214–223). PMLR.

Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., et al. (2022). Dos and don'ts of machine learning in computer security. In *31st USENIX security symposium* (pp. 3971–3988).

Asif, N. A., Sarker, Y., Chakrabortty, R. K., Ryan, M. J., Ahamed, M. H., Saha, D. K., et al. (2021). Graph neural network: A comprehensive review on non-euclidean space. *IEEE Access, 9*, 60588–60606.

Assouel, R., Ahmed, M., Segler, M. H., Saffari, A., & Bengio, Y. (2018). Defactor: Differentiable edge factorization-based probabilistic graph generation. arXiv preprint arXiv:1811.09766.

Bacciu, D., Errica, F., Micheli, A., & Podda, M. (2020). A gentle introduction to deep learning for graphs. *Neural Networks, 129*, 203–221.

Bagal, V., Aggarwal, R., Vinod, P., & Priyakumar, U. D. (2021). Molgpt: molecular generation using a transformer-decoder model. *Journal of Chemical Information and Modeling, 62*, 2064–2076.

Bai, X., Wang, M., Lee, I., Yang, Z., Kong, X., & Xia, F. (2019). Scientific paper recommendation: A survey. *IEEE Access, 7*, 9324–9339.

Bai, L., Yao, L., Li, C., Wang, X., & Wang, C. (2020). Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems, 33*, 17804–17815.

Balcilar, M., Héroux, P., Gauzere, B., Vasseur, P., Adam, S., & Honeine, P. (2021). Breaking the limits of message passing graph neural networks. In *International conference on machine learning* (pp. 599–608). PMLR.

Bartlett, P. L., Foster, D. J., & Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. *Advances in Neural Information Processing Systems, 30*.

Bartlett, P. L., Harvey, N., Liaw, C., & Mehrabian, A. (2019). Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research, 20*, 2285–2301.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., et al. (2021). E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. arXiv:2101.03164.

Belkin, M., & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the conference on neural information processing systems* (pp. 585–591).

Benslimane, S., Azé, J., Bringay, S., Servajean, M., & Mollevi, C. (2022). A text and gnn based controversy detection method on social media. *World Wide Web*, 1–27.

Berg, R. v. d., Kipf, T. N., & Welling, M. (2017). Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263.

Bian, T., Xiao, X., Xu, T., Zhao, P., Huang, W., Rong, Y., et al. (2020). Rumor detection on social media with bi-directional graph convolutional networks. *Vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 549–556).

Bianchi, F. M., Grattarola, D., & Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning* (pp. 874–883). PMLR.

Bo, D., Wang, X., Shi, C., & Shen, H. (2021). Beyond low-frequency information in graph convolutional networks. *Vol. 35*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3950–3957).

Bojchevski, A., & Günnemann, S. (2019). Adversarial attacks on node embeddings via graph poisoning. In *International conference on machine learning* (pp. 695–704). PMLR.

Borgwardt, K. M., & Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining* (pp. 8–pp). IEEE.

Bouadjenek, M. R., Sanner, S., & Wu, G. (2023). A user-centric analysis of social media for stock market prediction. *ACM Transactions on the Web, 17*, 1–22.

Bouritsas, G., Frasca, F., Zafeiriou, S. P., & Bronstein, M. (2022). Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bouzerdoum, A., & Pattison, T. R. (1993). Neural network for quadratic optimization with bound constraints. *IEEE Transactions on Neural Networks, 4*, 293–304.

Box, G. E., & Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association, 65*, 1509–1526.

Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J., & Welling, M. (2022). Geometric and physical quantities improve e(3) equivariant message passing. In *ICLR*. URL: https://openreview.net/forum?id=_xwr8gOBeV1.

Brody, S., Alon, U., & Yahav, E. (2021). How attentive are graph attention networks? arXiv preprint arXiv:2105.14491.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.

Bui, K.-H. N., Cho, J., & Yi, H. (2021). Spatial–temporal graph neural network for traffic forecasting: An overview and open research issues. *Applied Intelligence*, 1–12.

Cai, D., & Lam, W. (2020). Graph transformer for graph-to-sequence learning. *Vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 7464–7471).

Cai, H., Zheng, V. W., & Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering, 30*, 1616–1637.

Camacho, D., Panizo-LLedot, Á., Bello-Orgaz, G., Gonzalez-Pardo, A., & Cambria, E. (2020). The four dimensions of social network analysis: An overview of research methods, applications, and software tools. *Information Fusion, 63*, 88–120.

Cao, D., Li, J., Ma, H., & Tomizuka, M. (2021). Spectral temporal graph neural network for trajectory prediction. In *2021 IEEE international conference on robotics and automation* (pp. 1839–1845). IEEE.

Cao, J., Li, Q., Tu, W., Gao, Q., Cao, R., & Zhong, C. (2021). Resolving urban mobility networks from individual travel graphs using massive-scale mobile phone tracking data. *Cities, 110*, Article 103077.

Cao, S., Lu, W., & Xu, Q. (2015). Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management* (pp. 891–900).

Cao, S., Lu, W., & Xu, Q. (2016). Deep neural networks for learning graph representations. *Vol. 30*, In *Proceedings of the AAAI conference on artificial intelligence*.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In *Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part I 16* (pp. 213–229). Springer.

Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy* (pp. 39–57). Ieee.

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., & Murphy, K. (2022). Machine learning on graphs: A model and comprehensive taxonomy. *Journal of Machine Learning Research, 23*, 1–64.

Chauhan, J., Nathani, D., & Kaul, M. (2020). Few-shot learning on graphs via super-classes based on graph spectral measures. arXiv preprint arXiv:2002.12815.

Chen, T., Bian, S., & Sun, Y. (2019). Are powerful graph neural nets necessary? a dissection on graph classification. arXiv preprint arXiv:1905.04579.

Chen, Z., Chen, L., Villar, S., & Bruna, J. (2020). Can graph neural networks count substructures? *Advances in Neural Information Processing Systems, 33*, 10383–10395.

Chen, Z., Chen, F., Zhang, L., Ji, T., Fu, K., Zhao, L., et al. (2020). Bridging the gap between spatial and spectral domains: A survey on graph neural networks. arXiv preprint arXiv:2002.11867.

Chen, Z., Chen, D., Zhang, X., Yuan, Z., & Cheng, X. (2021). Learning graph structures with transformer for multivariate time series anomaly detection in iot. *IEEE Internet of Things Journal*.

Chen, S., Guhur, P.-L., Schmid, C., & Laptev, I. (2021). History aware multimodal transformer for vision-and-language navigation. *Advances in Neural Information Processing Systems, 34*, 5834–5847.

Chen, D., Jacob, L., & Mairal, J. (2020). Convolutional kernel networks for graph-structured data. arXiv preprint arXiv:2003.05189.

Chen, L., Li, J., Peng, J., Xie, T., Cao, Z., Xu, K., et al. (2020). A survey of adversarial learning on graphs. arXiv preprint arXiv:2003.05730.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3438–3445).

Chen, D., O'Bray, L., & Borgwardt, K. (2022). Structure-aware transformer for graph representation learning. In *International conference on machine learning* (pp. 3469–3489). PMLR.

Chen, H., Perozzi, B., Hu, Y., & Skiena, S. (2018). Harp: Hierarchical representation learning for networks. *Vol. 32*, In *Proceedings of the AAAI conference on artificial intelligence*.

Chen, G., Tian, Y., & Song, Y. (2020). Joint aspect extraction and sentiment analysis with directional graph convolutional networks. In *Proceedings of the 28th international conference on computational linguistics* (pp. 272–279).

Chen, F., Wang, Y.-C., Wang, B., & Kuo, C.-C. J. (2020). Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing, 9*, Article e15.

Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020). Simple and deep graph convolutional networks. In *International conference on machine learning* (pp. 1725–1735). PMLR.

Chen, C., Wu, Y., Dai, Q., Zhou, H.-Y., Xu, M., Yang, S., et al. (2022). A survey on graph neural networks and graph transformers in computer vision: A task-oriented perspective. arXiv preprint arXiv:2209.13232.

Chen, Y., Wu, L., & Zaki, M. (2020). Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in Neural Information Processing Systems, 33*, 19314–19326.

Chen, X., Yuan, Y., Zeng, G., & Wang, J. (2021). Semi-supervised semantic segmentation with cross pseudo supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2613–2622).

Chen, B., Zhang, J., Tang, J., Cai, L., Wang, Z., Zhao, S., et al. (2020). Conna: Addressing name disambiguation on the fly. *IEEE Transactions on Knowledge and Data Engineering*.

Cho, H., & Yu, Y. (2018). Link prediction for interdisciplinary collaboration via co-authorship network. *Social Network Analysis and Mining, 8*, 1–12.

Choi, J., Choi, H., Hwang, J., & Park, N. (2022). Graph neural controlled differential equations for traffic forecasting. *Vol. 36*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 6367–6374).

Chouldechova, A., & Roth, A. (2018). The frontiers of fairness in machine learning. arXiv preprint arXiv:1810.08810.

Chuan, P. M., Son, L. H., Ali, M., Khang, T. D., Huong, L. T., & Dey, N. (2018). Link prediction in co-authorship networks based on hybrid content similarity metric. *Applied Intelligence, 48*, 2470–2486.

Chung, F. R. (1997). *Spectral graph theory*: *vol. 92*, American Mathematical Soc..

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Coley, C. W., Barzilay, R., Green, W. H., Jaakkola, T. S., & Jensen, K. F. (2017). Convolutional embedding of attributed molecular graphs for physical property prediction. *Journal of Chemical Information and Modeling, 57*, 1757–1772.

Connor, J. T., Martin, R. D., & Atlas, L. E. (1994). Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks, 5*, 240–254.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., & Veličković, P. (2020). Principal neighbour-hood aggregation for graph nets. *Advances in Neural Information Processing Systems, 33*, 13260–13271.

Cremer, J., Medrano Sandonas, L., Tkatchenko, A., Clevert, D.-A., & De Fabritiis, G. (2023). Equivariant graph neural networks for toxicity prediction. *Chemical Research in Toxicology*.

Dahou, A., Mabrouk, A., Ewees, A. A., Gaheen, M. A., & Abd Elaziz, M. (2023). A social media event detection framework based on transformers and swarm optimization for public notification of crises and emergency management. *Technological Forecasting and Social Change, 192*, Article 122546.

Dai, E., & Wang, S. (2021). Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 680–688).

Dai, T., Zhao, J., Li, D., Tian, S., Zhao, X., & Pan, S. (2023). Heterogeneous deep graph convolutional network with citation relational bert for covid-19 inline citation recommendation. *Expert Systems with Applications, 213*, Article 118841.

De Cao, N., & Kipf, T. (2018). Molgan: An implicit generative model for small molecular graphs. arXiv preprint arXiv:1805.11973.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems, 29*.

Delaney, J. S. (2004). Esol: estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences, 44*, 1000–1005.

Deng, Z., Dong, Y., & Zhu, J. (2019). Batch virtual adversarial training for graph convolutional networks. arXiv preprint arXiv:1902.09192.

Deng, A., & Hooi, B. (2021). Graph neural network-based anomaly detection in multivariate time series. *Vol. 35*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4027–4035).

Deng, L., Lian, D., Huang, Z., & Chen, E. (2022). Graph convolutional adversarial networks for spatiotemporal anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems, 33*, 2416–2428.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Di Domenico, G., Sit, J., Ishizaka, A., & Nunan, D. (2021). Fake news, social media and marketing: A systematic review. *Journal of Business Research, 124*, 329–341.

Dimitrakopoulos, G., & Demestichas, P. (2010). Intelligent transportation systems. *IEEE Vehicular Technology Magazine, 5*, 77–84.

Domingos, P., & Richardson, M. (2001). Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 57–66).

Dong, Y., Chawla, N. V., & Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 135–144).

Dong, Y., Kang, J., Tong, H., & Li, J. (2021). Individual fairness for graph neural networks: A ranking based approach. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 300–310).

Dong, Y., Liu, N., Jalaian, B., & Li, J. (2022). Edits: Modeling and mitigating data bias for graph neural networks. In *Proceedings of the ACM web conference 2022* (pp. 1259–1269).

Dong, Y., Ma, J., Wang, S., Chen, C., & Li, J. (2023). Fairness in graph mining: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Du, Y., Fu, T., Sun, J., & Liu, S. (2022). Molgensurvey: A systematic survey in machine learning models for molecule design. arXiv preprint arXiv:2203.14500.

Du, Y., Guo, X., Shehu, A., & Zhao, L. (2022). Interpretable molecular graph generation via monotonic constraints. In *Proceedings of the 2022 SIAM international conference on data mining* (pp. 73–81). SIAM.

Du, S. S., Hou, K., Salakhutdinov, R. R., Poczos, B., Wang, R., & Xu, K. (2019). Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in neural information processing systems* (pp. 5723–5733).

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (pp. 1285–1298).

Du, J., Wang, S., Miao, H., & Zhang, J. (2021). Multi-channel pooling graph neural networks. In Z.-H. Zhou (Ed.), *Proceedings of the thirtieth international joint conference on artificial intelligence* (pp. 1442–1448). International Joint Conferences on Artificial Intelligence Organization, http://dx.doi.org/10.24963/ijcai.2021/199, main Track.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., et al. (2015). Convolutional networks on graphs for learning molecular fingerprints. *Advances in Neural Information Processing Systems, 28*.

Dwivedi, V. P., & Bresson, X. (2020). A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699.

Eberhardt, J., Santos-Martins, D., Tillack, A. F., & Forli, S. (2021). Autodock vina 1.2. 0: New docking methods, expanded force field, and python bindings. *Journal of Chemical Information and Modeling, 61*, 3891–3898.

Elinas, P., Bonilla, E. V., & Tiao, L. (2020). Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. *Advances in Neural Information Processing Systems, 33*, 18648–18660.

Elsayed, G. F., Goodfellow, I., & Sohl-Dickstein, J. (2018). Adversarial reprogramming of neural networks. arXiv preprint arXiv:1806.11146.

Empel, C., & Koenigs, R. M. (2019). Artificial-intelligence-driven organic synthesis—en route towards autonomous synthesis? *Angewandte Chemie International Edition, 58*, 17114–17116.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., & Papalexakis, E. E. (2020). All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th international conference on web search and data mining* (pp. 169–177).

Errica, F., Podda, M., Bacciu, D., & Micheli, A. (2019). A fair comparison of graph neural networks for graph classification. arXiv preprint arXiv:1912.09893.

Fan, Z., Liu, Z., Zhang, J., Xiong, Y., Zheng, L., & Yu, P. S. (2021). Continuous-time sequential recommendation with temporal graph collaborative transformer. In *Proceedings of the 30th ACM international conference on information & knowledge management* (pp. 433–442).

Fang, Z., Long, Q., Song, G., & Xie, K. (2021). Spatial–temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 364–373).

Fang, Z., Xu, L., Song, G., Long, Q., & Zhang, Y. (2022). Polarized graph neural networks. In *Proceedings of the ACM web conference 2022* (pp. 1404–1413).

Fang, Y., Zhang, Q., Yang, H., Zhuang, X., Deng, S., Zhang, W., et al. (2022). Molecular contrastive learning with chemical element knowledge graph. *Vol. 36*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3968–3976).

Fang, Y., Zhang, Q., Zhang, N., Chen, Z., Zhuang, X., Shao, X., et al. (2023). Knowledge graph-enhanced molecular contrastive learning with functional prompt. *Nature Machine Intelligence*, 1–12.

Farhat, M., & Ben-Hamadou, A. (2023). Graph self-supervised learning for endoscopic image matching. arXiv preprint arXiv:2306.11141.

Feinberg, E. N., Sur, D., Wu, Z., Husic, B. E., Mai, H., Li, Y., et al. (2018). Potentialnet for molecular property prediction. *ACS Central Science, 4*, 1520–1530.

Feng, S., Wan, H., Wang, N., & Luo, M. (2021). Botrgcn: Twitter bot detection with relational graph convolutional networks. In *Proceedings of the 2021 IEEE/ACM international conference on advances in social networks analysis and mining* (pp. 236–239).

Feng, A., You, C., Wang, S., & Tassiulas, L. (2022). Kergnns: Interpretable graph neural networks with graph kernels. ArXiv Preprint: https://arxiv.org/abs/2201.00491.

Feng, Y., You, H., Zhang, Z., Ji, R., & Gao, Y. (2019). Hypergraph neural networks. *Vol. 33*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3558–3565).

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., et al. (2020). Graph random neural networks for semi-supervised learning on graphs. (pp. 22092–22103).

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135). PMLR.

Finzi, M., Stanton, S., Izmailov, P., & Wilson, A. G. (2020). Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *ICML*.

Fischer, E. (1894). Einfluss der configuration auf die wirkung der enzyme. *Berichte der Deutschen Chemischen Gesellschaft, 27*, 2985–2993.

Fisher, J., Palfrey, D., Christodoulopoulos, C., & Mittal, A. (2019). Measuring social bias in knowledge graph embeddings. arXiv preprint arXiv:1912.02761.

Franceschi, L., Niepert, M., Pontil, M., & He, X. (2019). Learning discrete structures for graph neural networks. In *International conference on machine learning* (pp. 1972–1982). PMLR.

Francoeur, P. G., Masuda, T., Sunseri, J., Jia, A., Iovanisci, R. B., Snyder, I., et al. (2020). Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of Chemical Information and Modeling, 60*, 4200–4215.

Frisch, M. e., Trucks, G., Schlegel, H., Scuseria, G., Robb, M., Cheeseman, J., et al. (2016). Gaussian 16.

Fu, S., & Liu, W. (2021). Recent advances of manifold-based graph convolutional networks for remote sensing images recognition. In *Generalization with deep learning: for improvement on sensing capability* (pp. 209–232).

Fu, Z., Xian, Y., Gao, R., Zhao, J., Huang, Q., Ge, Y., et al. (2020). Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 69–78).

Fu, G., Zhao, P., & Bian, Y. (2022). *p*-Laplacian based graph neural networks. In *International conference on machine learning* (pp. 6878–6917). PMLR.

Fuchs, F., Worrall, D., Fischer, V., & Welling, M. (2020). Se(3)-transformers: 3d roto-translation equivariant attention networks. *Vol. 33*, In *NeurIPS*. URL: https://proceedings.neurips.cc/paper/2020/file/15231a7ce4ba789d13b722cc5c955834-Paper.pdf.

Ganea, O.-E., Huang, X., Bunne, C., Bian, Y., Barzilay, R., Jaakkola, T., et al. (2021). Independent se (3)-equivariant models for end-to-end rigid protein docking. arXiv preprint arXiv:2111.07786.

Gao, X., Hu, W., & Guo, Z. (2020). Exploring structure-adaptive graph learning for robust semi-supervised classification. In *2020 ieee international conference on multimedia and expo* (pp. 1–6). IEEE.

Gao, H., & Ji, S. (2019a). Graph representation learning via hard and channel-wise attention networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 741–749).

Gao, H., & Ji, S. (2019b). Graph u-nets. In *International conference on machine learning* (pp. 2083–2092). PMLR.

Gao, H., Liu, Y., & Ji, S. (2021). Topology-aware graph pooling networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*, 4512–4518.

Gao, Y., Wang, X., He, X., Feng, H., & Zhang, Y. (2023). Rumor detection with self-supervised learning on texts and social graph. *Frontiers of Computer Science, 17*, Article 174611.

Gärtner, T., Flach, P., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In *Proceedings of Computational Learning theory and kernel machines* (pp. 129–143).

Gasteiger, J., Bojchevski, A., & Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997.

Gasteiger, J., Weißenberger, S., & Günnemann, S. (2019). Diffusion improves graph learning. *Advances in Neural Information Processing Systems, 32*.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., et al. (2021). Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics, 22*, bbab159.

Gayvert, K. M., Madhukar, N. S., & Elemento, O. (2016). A data-driven approach to predicting successes and failures of clinical trials. *Cell Chemical Biology, 23*, 1294–1301.

Gebauer, N. W., Gastegger, M., Hessmann, S. S., Müller, K.-R., & Schütt, K. T. (2022). Inverse design of 3d molecular structures with conditional generative neural networks. *Nature Communications, 13*, 1–11.

Gebauer, N., Gastegger, M., & Schütt, K. (2019). Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. *Advances in Neural Information Processing Systems, 32*.

Geiger, M., & Smidt, T. (2022). e3nn: Euclidean neural networks. arXiv preprint arXiv:2207.09453.

Geisler, S., Li, Y., Mankowitz, D. J., Cemgil, A. T., Günnemann, S., & Paduraru, C. (2023). Transformers meet directed graphs. In *International conference on machine learning* (pp. 11144–11172). PMLR.

Geisler, S., Schmidt, T., Şirin, H., Zügner, D., Bojchevski, A., & Günnemann, S. (2021). Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems, 34*, 7637–7649.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning* (pp. 1263–1272). PMLR.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., et al. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science, 4*, 268–276.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2020). Generative adversarial networks. *Communications of the ACM, 63*, 139–144.

Goodman, B., & Flaxman, S. (2017). European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine, 38*, 50–57.

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems, 151*, 78–94.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems, 33*, 21271–21284.

Grohs, P., & Voigtlaender, F. (2021). Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces. arXiv preprint arXiv:2104.02746.

Groom, C. R., Bruno, I. J., Lightfoot, M. P., & Ward, S. C. (2016). The cambridge structural database. *Acta Crystallographica Section B: Structural Science, Crystal Engineering and Materials, 72*, 171–179.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855–864).

Guan, J., Qian, W. W., Peng, X., Su, Y., Peng, J., & Ma, J. (2023). 3D equivariant diffusion for target-aware molecule generation and affinity prediction. arXiv preprint arXiv:2303.03543.

Günnemann, S. (2022). Graph neural networks: Adversarial robustness. In *Graph neural networks: foundations, frontiers, and applications* (pp. 149–176). Springer.

Guo, T., Guo, K., Liang, Z., Guo, Z., Chawla, N. V., Wiest, O., et al. (2023). What indeed can gpt models do in chemistry? a comprehensive benchmark on eight tasks. arXiv preprint arXiv:2305.18365.

Guo, R., Li, J., & Liu, H. (2020). Learning individual causal effects from networked observational data. In *Proceedings of the 13th international conference on web search and data mining* (pp. 232–240).

Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019). Attention based spatial–temporal graph convolutional networks for traffic flow forecasting. *Vol. 33, In Proceedings of the AAAI conference on artificial intelligence* (pp. 922–929).

Guo, Z., Zhang, C., Yu, W., Herr, J., Wiest, O., Jiang, M., et al. (2021). Few-shot graph learning for molecular property prediction. In *Proceedings of the web conference 2021* (pp. 2559–2567).

Hamidian, S., & Diab, M. T. (2019). Rumor detection and classification for twitter data. arXiv preprint arXiv:1912.08926.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems, 30*.

Hammond, D. K., Vandergheynst, P., & Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis, 30*, 129–150.

Han, J., Rong, Y., Xu, T., & Huang, W. (2022). Geometrically equivariant graph neural networks: A survey. arXiv preprint arXiv:2202.07230.

Hao, Z., Lu, C., Huang, Z., Wang, H., Hu, Z., Liu, Q., et al. (2020). Asgn: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 731–752).

Harvey, N., Liaw, C., & Mehrabian, A. (2017). Nearly-tight vc-dimension bounds for piecewise linear neural networks. In *Conference on learning theory* (pp. 1064–1068). PMLR.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 639–648).

He, Y., Permultter, M., Reinert, G., & Cucuringu, M. (2022). Msgnn: A spectral graph neural network based on a novel magnetic signed laplacian. arXiv preprint arXiv:2209.00546.

He, D., Zhao, Y., Luo, J., Hui, T., Huang, S., Zhang, A., et al. (2021). Transrefer3d: Entity-and-relation aware transformer for fine-grained 3d visual grounding. In *Proceedings of the 29th ACM international conference on multimedia* (pp. 2344–2352).

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their Applications, 13*, 18–28.

Henaff, M., Bruna, J., & LeCun, Y. (2015). Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163.

Hendrycks, D., & Gimpel, K. (2016). A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*, 504–507.

Ho, T. K. K., & Armanfard, N. (2023). Self-supervised learning for anomalous channel detection in eeg graphs: application to seizure analysis. *Vol. 37, In Proceedings of the AAAI conference on artificial intelligence* (pp. 7866–7874).

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems, 33*, 6840–6851.

Hofman, J. M., Watts, D. J., Athey, S., Garip, F., Griffiths, T. L., Kleinberg, J., et al. (2021). Integrating explanation and prediction in computational social science. *Nature, 595*, 181–188.

Hoogeboom, E., Satorras, V. G., Vignac, C., & Welling, M. (2022). Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning* (pp. 8867–8887). PMLR.

Hu, Z., Dong, Y., Wang, K., Chang, K.-W., & Sun, Y. (2020). Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1857–1867).

Hu, Z., Dong, Y., Wang, K., & Sun, Y. (2020). Heterogeneous graph transformer. In *Proceedings of the web conference 2020* (pp. 2704–2710).

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., et al. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems, 33*, 22118–22133.

Hu, Z., & Li, L. E. (2021). A causal lens for controllable text generation. *Advances in Neural Information Processing Systems, 34*, 24941–24955.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., et al. (2019). Strategies for pre-training graph neural networks. arXiv preprint arXiv:1905.12265.

Hu, F., Zhu, Y., Wu, S., Huang, W., Wang, L., & Tan, T. (2020). Graphair: Graph representation learning with neighborhood aggregation and interaction. *Pattern Recognition, 112*, Article 107745.

Huang, S., Elhoseiny, M., Elgammal, A., & Yang, D. (2015). Learning hypergraph-regularized attribute predictors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 409–417).

Huang, W., Han, J., Rong, Y., Xu, T., Sun, F., & Huang, J. (2022). Constrained graph mechanics networks. In *ICLR*. URL: https://openreview.net/forum?id=SHbhHHfePhP.

Huang, R., Huang, C., Liu, Y., Dai, G., & Kong, W. (2020). Lsgcn: Long short-term traffic prediction with graph convolutional networks. *Vol. 7, In IJCAI* (pp. 2355–2361).

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P. (2017). Adversarial attacks on neural network policies. arXiv preprint arXiv:1702.02284.

Huang, J., & Yang, J. (2021). Unignn: a unified framework for graph and hypergraph neural networks. arXiv preprint arXiv:2105.00956.

Huang, J., Yi, P., Wang, J., Li, M., Peng, J., & Xiong, X. (2022). A dynamical spatial–temporal graph neural network for traffic demand prediction. *Information Sciences, 594*, 286–304.

Hussain, M. S., Zaki, M. J., & Subramanian, D. (2021). Edge-augmented graph transformers: Global self-attention is enough for graphs. arXiv preprint arXiv:2108.03348.

Hutchinson, M. J., Le Lan, C., Zaidi, S., Dupont, E., Teh, Y. W., & Kim, H. (2021). Lietransformer: equivariant self-attention for lie groups. In *ICML*.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., & Coleman, R. G. (2012). Zinc: a free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling, 52*, 1757–1768.

Ishida, S., Terayama, K., Kojima, R., Takasu, K., & Okuno, Y. (2022). Ai-driven synthetic route design incorporated with retrosynthesis knowledge. *Journal of Chemical Information and Modeling, 62*, 1357–1367.

Islam, M. A., Mohammad, M. M., Das, S. S. S., & Ali, M. E. (2022). A survey on deep learning based point-of-interest (poi) recommendations. *Neurocomputing, 472*, 306–325.

Jablonka, K. M., Schwaller, P., Ortega-Guerrero, A., & Smit, B. (2023). Leveraging large language models for predictive chemistry.

Ji, S., Feng, Y., Ji, R., Zhao, X., Tang, W., & Gao, Y. (2020). Dual channel hyper-graph collaborative filtering. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2020–2029).

Ji, J., Wang, J., Huang, C., Wu, J., Xu, B., Wu, Z., et al. (2023). Spatio-temporal self-supervised learning for traffic flow prediction. *Vol. 37, In Proceedings of the AAAI conference on artificial intelligence* (pp. 4356–4364).

Jiang, P. (2023). Gode–integrating biochemical knowledge graph into pre-training molecule graph neural network. arXiv preprint arXiv:2306.01631.

Jiang, J., Chen, J., Gu, T., Choo, K.-K. R., Liu, C., Yu, M., et al. (2019). Anomaly detection with graph convolutional networks for insider threat and fraud detection. In *MILCOM 2019-2019 IEEE military communications conference* (pp. 109–114). IEEE.

Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, Article 117921.

Jiang, Z., Yin, Y., Gao, L., Lu, Y., & Liu, X. (2018). Cross-language citation recommendation via hierarchical representation learning on heterogeneous graph. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 635–644).

Jiang, B., Zhang, Z., Lin, D., Tang, J., & Luo, B. (2019). Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11313–11320).

Jiao, Y., Xiong, Y., Zhang, J., Zhang, Y., Zhang, T., & Zhu, Y. (2020). Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE international conference on data mining* (pp. 222–231). IEEE.

Jiménez-Luna, J., Grisoni, F., & Schneider, G. (2020). Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence, 2*, 573–584.

Jin, W., Barzilay, R., & Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning* (pp. 2323–2332). PMLR.

Jin, W., Derr, T., Liu, H., Wang, Y., Wang, S., Liu, Z., et al. (2020). Self-supervised learning on graphs: Deep insights and new direction. arXiv preprint arXiv:2006.10141.

Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., et al. (2021). Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, *22*, 19–34.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., & Tang, J. (2020). Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 66–74).

Jin, G., Xi, Z., Sha, H., Feng, Y., & Huang, J. (2020). Deep multi-view spatiotemporal virtual graph neural network for significant citywide ride-hailing demand prediction. arXiv preprint arXiv:2007.15189.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J., & Dror, R. (2020). Learning from protein structure with geometric vector perceptrons. arXiv preprint arXiv:2009.01411.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J. L., & Dror, R. (2021). Learning from protein structure with geometric vector perceptrons. In *ICLR*. URL: https://openreview.net/forum?id=1YLJDvSx6J4.

Ju, W., Gu, Y., Chen, B., Sun, G., Qin, Y., Liu, X., et al. (2023). Glcc: A general framework for graph-level clustering. *Vol. 37*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4391–4399).

Ju, W., Gu, Y., Luo, X., Wang, Y., Yuan, H., Zhong, H., et al. (2023). Unsupervised graph-level representation learning with hierarchical contrasts. *Neural Networks*, *158*, 359–368.

Ju, W., Liu, Z., Qin, Y., Feng, B., Wang, C., Guo, Z., et al. (2023). Few-shot molecular property prediction via hierarchically structured learning on relation graphs. *Neural Networks*.

Ju, W., Luo, X., Ma, Z., Yang, J., Deng, M., & Zhang, M. (2022). Ghnn: Graph harmonic neural networks for semi-supervised graph-level classification. *Neural Networks*, *151*, 70–79.

Ju, W., Luo, X., Qin, Y., Qiao, Z., Yi, S., Liu, L., et al. (2024). Focus on informative graphs! semi-supervised active learning for graph-level classification.

Ju, W., Luo, X., Qu, M., Wang, Y., Chen, C., Deng, M., et al. (2022). Tgnn: A joint semi-supervised framework for graph-level classification. In *Proceedings of the international joint conference on artificial intelligence* (pp. 2122–2128).

Ju, W., Qin, Y., Qiao, Z., Luo, X., Wang, Y., Fu, Y., et al. (2022). Kernel-based substructure exploration for next poi recommendation. arXiv preprint arXiv:2210.03969.

Ju, W., Qin, Y., Yi, S., Mao, Z., Zheng, K., Liu, L., et al. (2023). Zero-shot node classification with graph contrastive embedding network. *Transactions on Machine Learning Research*.

Ju, W., Yang, J., Qu, M., Song, W., Shen, J., & Zhang, M. (2022). Kgnn: Harnessing kernel-based networks for semi-supervised graph classification. In *Proceedings of the fifteenth ACM international conference on web search and data mining* (pp. 421–429).

Kang, U., Tong, H., & Sun, J. (2012). Fast random walk graph kernel. In *Proceedings of the SIAM international conference on data mining* (pp. 828–838). SIAM.

Karpinski, M., & Macintyre, A. (1997). Polynomial bounds for vc dimension of sigmoidal and general pfaffian neural networks. *Journal of Computer and System Sciences*, *54*, 169–176.

Kashima, H., Tsuda, K., & Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In *Proceedings of international conference on machine learning* (pp. 321–328).

Keikha, M. M., Rahgozar, M., Asadpour, M., & Abdollahi, M. F. (2020). Influence maximization across heterogeneous interconnected networks based on deep learning. *Expert Systems with Applications*, *140*, Article 112905.

Khoshraftar, S., & An, A. (2022). A survey on graph representation learning methods. arXiv preprint arXiv:2204.01855.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., & Zemel, R. (2018). Neural relational inference for interacting systems. In *International conference on machine learning* (pp. 2688–2697). PMLR.

Kipf, T. N., & Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Kipf, T. N., & Welling, M. (2016b). Variational graph auto-encoders. arXiv preprint arXiv:1611.07308.

Klicpera, J., Becker, F., & Günnemann, S. (2021). Gemnet: Universal directional graph neural networks for molecules. In *NeurIPS*. URL: https://openreview.net/forum?id=HS_sOaxS9K-.

Klicpera, J., Groß, J., & Günnemann, S. (2020). Directional message passing for molecular graphs. In *ICLR*. URL: https://openreview.net/forum?id=B1eWbxStPH.

Köhler, J., Klein, L., & Noe, F. (2020). Equivariant flows: Exact likelihood generative learning for symmetric densities. In *ICML*. URL: https://proceedings.mlr.press/v119/kohler20a.html.

Kong, X., Jiang, H., Wang, W., Bekele, T. M., Xu, Z., & Wang, M. (2017). Exploring dynamic research interest and academic influence for scientific collaborator recommendation. *Scientometrics*, *113*, 369–385.

Kong, X., Jiang, H., Yang, Z., Xu, Z., Xia, F., & Tolba, A. (2016). Exploiting publication contents and collaboration networks for collaborator recommendation. *PLoS One*, *11*, Article e0148492.

Kong, X., Shi, Y., Yu, S., Liu, J., & Xia, F. (2019). Academic social networks: Modeling, analysis, mining and applications. *Journal of Network and Computer Applications*, *132*, 86–103.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 426–434).

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*, 30–37.

Krenn, M., Häse, F., Nigam, A., Friederich, P., & Aspuru-Guzik, A. (2020). Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, *1*, Article 045024.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., & Tossou, P. (2021). Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, *34*, 21618–21629.

Kriege, N. M., Johansson, F. D., & Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, *5*, 1–42.

Krishnagopal, S., & Ruiz, L. (2023). Graph neural tangent kernel: Convergence on large graphs. arXiv preprint arXiv:2301.10808.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*, 84–90.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, *29*, 1–27.

Kumar, S., Mallik, A., Khetarpal, A., & Panda, B. (2022). Influence maximization in social networks using graph embedding and graph neural network. *Information Sciences*, *607*, 1617–1636.

Laine, S., & Aila, T. (2016). Temporal ensembling for semi-supervised learning. arXiv preprint arXiv:1610.02242.

Lambrecht, A., & Tucker, C. (2019). Algorithmic bias? an empirical study of apparent gender-based discrimination in the display of stem career ads. *Management Science*, *65*, 2966–2981.

Landrum, G., et al. (2013). Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*.

Lee, D., Kim, S., Lee, S., Park, C., & Yu, H. (2021). Learnable structural semantic readout for graph classification. In *2021 IEEE international conference on data mining* (pp. 1180–1185). IEEE.

Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. In *International conference on machine learning* (pp. 3734–3743). PMLR.

Lee, D.-H., et al. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *Vol. 3*, In *Workshop on challenges in representation learning, ICML* (p. 896).

Lei, T., Jin, W., Barzilay, R., & Jaakkola, T. (2017). Deriving neural architectures from sequence and graph kernels. In *International conference on machine learning* (pp. 2024–2033).

Levie, R., Monti, F., Bresson, X., & Bronstein, M. M. (2018). Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, *67*, 97–109.

Li, J., Cai, D., & He, X. (2017). Learning graph-level representation for drug discovery. arXiv preprint arXiv:1709.03741.

Li, Y., Chen, H., Fu, Z., Ge, Y., & Zhang, Y. (2021). User-oriented fairness in recommendation. In *Proceedings of the web conference 2021* (pp. 624–632).

Li, Q., Han, Z., & Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. *Vol. 32*, In *Proceedings of the AAAI conference on artificial intelligence*.

Li, Y., Hu, J., Wang, Y., Zhou, J., Zhang, L., & Liu, Z. (2019). Deepscaffold: a comprehensive tool for scaffold-based de novo drug discovery using deep learning. *Journal of Chemical Information and Modeling*, *60*, 77–91.

Li, J., Huang, Y., Chang, H., & Rong, Y. (2022). Semi-supervised hierarchical graph classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Li, Y., Jin, W., Xu, H., & Tang, J. (2020). Deeprobust: A pytorch library for adversarial attacks and defenses. arXiv preprint arXiv:2005.06149.

Li, H., Li, X., Zhang, X., Zhao, C., & Wang, Z. (2023). Detecting early-warning signals for social emergencies by temporal network sociomarkers. *Information Sciences*, *627*, 189–204.

Li, A., & Pan, Y. (2016). Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, *62*, 3290–3339.

Li, Y., Pei, J., & Lai, L. (2021a). Learning to design drug-like molecules in three-dimensional space using deep generative models. arXiv preprint arXiv:2104.08474.

Li, Y., Pei, J., & Lai, L. (2021b). Structure-based de novo drug design using 3d deep generative models. *Chemical Science*, *12*, 13664–13675.

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., & Ma, J. (2017). Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 1419–1428).

Li, J., Rong, Y., Cheng, H., Meng, H., Huang, W., & Huang, J. (2019). Semi-supervised graph classification: A hierarchical graph perspective. In *Proceedings of the web conference* (pp. 972–982).

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2015). Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493.

Li, H., Wang, X., Zhang, Z., & Zhu, W. (2022a). Ood-gnn: Out-of-distribution generalized graph neural network. *IEEE Transactions on Knowledge and Data Engineering*.

Li, H., Wang, X., Zhang, Z., & Zhu, W. (2022b). Out-of-distribution generalization on graphs: A survey. arXiv preprint arXiv:2202.07987.

Li, R., Wang, S., Zhu, F., & Huang, J. (2018). Adaptive graph convolutional neural networks. *Vol. 32*, In *Proceedings of the AAAI conference on artificial intelligence*.

Li, G., Xiong, C., Thabet, A., & Ghanem, B. (2020). Deepergcn: All you need to train deeper gcns. arXiv preprint arXiv:2006.07739.

Li, P., Yang, Y., Pagnucco, M., & Song, Y. (2022). Cognet: Cooperative graph neural networks. In *Proceedings of the international joint conference on neural networks*.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926.

Li, Z., Yu, J., Zhang, G., & Xu, L. (2023). Dynamic spatio-temporal graph network with adaptive propagation mechanism for multivariate time series forecasting. *Expert Systems with Applications, 216*, Article 119374.

Li, Y., Zemel, R., Brockschmidt, M., & Tarlow, D. (2016). Gated graph sequence neural networks. In *Proceedings of ICLR'16*.

Li, Z., Zhang, L., & Song, G. (2019). Gcn-lase: Towards adequately incorporating link attributes in graph convolutional networks. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence* (pp. 2959–2965).

Li, H., Zhao, Y., Mao, Z., Qin, Y., Xiao, Z., Feng, J., et al. (2024). A survey on graph neural networks in intelligent transportation systems. arXiv preprint arXiv:2401.00713.

Li, S., Zhou, J., Xu, T., Dou, D., & Xiong, H. (2022). Geomgcl: geometric graph contrastive learning for molecular property prediction. *Vol. 36, In Proceedings of the AAAI conference on artificial intelligence* (pp. 4541–4549).

Liang, Y., Zhang, Y., Gao, D., & Xu, Q. (2020). Mxpool: Multiplex pooling for hierarchical graph representation learning. arXiv preprint arXiv:2004.06846.

Liao, R., Zhao, Z., Urtasun, R., & Zemel, R. S. (2019). Lanczosnet: Multi-scale deep graph convolutional networks. arXiv preprint arXiv:1901.01484.

Lim, J., Um, D., Chang, H. J., Jo, D. U., & Choi, J. Y. (2021). Class-attentive diffusion network for semi-supervised classification. *Vol. 35, In Proceedings of the AAAI conference on artificial intelligence* (pp. 8601–8609).

Lin, H., Huang, Y., Liu, M., Li, X., Ji, S., & Li, S. Z. (2022). Diffbp: Generative diffusion of 3d molecules for target protein binding. arXiv preprint arXiv:2211.11214.

Lin, J., Xu, H., Woicik, A., Ma, J., & Wang, S. (2022). Pisces: A cross-modal contrastive learning approach to synergistic drug combination prediction. *bioRxiv*.

Liu, Q., Allamanis, M., Brockschmidt, M., & Gaunt, A. (2018). Constrained graph variational autoencoders for molecule design. *Advances in Neural Information Processing Systems, 31*.

Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., & Song, L. (2018). Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 2077–2085).

Liu, L., Cheung, W. K., Li, X., & Liao, L. (2016). Aligning users across social networks using network embedding. *Vol. 16, In Ijcai* (pp. 1774–1780).

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., et al. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10012–10022).

Liu, M., Luo, Y., Uchino, K., Maruhashi, K., & Ji, S. (2022). Generating 3d molecules for target protein binding. arXiv preprint arXiv:2204.09410.

Liu, P., Ren, Y., & Ren, Z. (2023). Git-mol: A multi-modal large language model for molecular science with graph, image, and text. arXiv preprint arXiv:2308.06911.

Liu, C., Shen, J., Xin, H., Liu, Z., Yuan, Y., Wang, H., et al. (2023). Fimo: A challenge formal dataset for automated theorem proving. arXiv preprint arXiv:2309.04295.

Liu, Y., Wang, L., Liu, M., Lin, Y., Zhang, X., Oztekin, B., et al. (2022). Spherical message passing for 3d molecular graphs. In *ICLR*. URL: https://openreview.net/forum?id=givsRXsOt9r.

Liu, Y. L., Wang, Y., Vu, O., Moretti, R., Bodenheimer, B., Meiler, J., et al. (2023). Interpretable chirality-aware graph neural network for quantitative structure activity relationship modeling in drug discovery. *Vol. 37, In Proceedings of the AAAI conference on artificial intelligence* (pp. 14356–14364).

Liu, J., Xia, F., Wang, L., Xu, B., Kong, X., Tong, H., et al. (2019). Shifu2: A network representation learning based model for advisor-advisee relationship mining. *IEEE Transactions on Knowledge and Data Engineering, 33*, 1763–1777.

Liu, Z., Xie, X., & Chen, L. (2018). Context-aware academic collaborator recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1870–1879).

Liu, Q., Zeng, Y., Mokhosi, R., & Zhang, H. (2018). Stamp: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1831–1839).

Liu, G., Zhao, T., Inae, E., Luo, T., & Jiang, M. (2023). Semi-supervised graph imbalanced regression. In *Proceedings of the international ACM SIGKDD conference on knowledge discovery & data mining*.

Long, J., Chen, T., Nguyen, Q. V. H., & Yin, H. (2023). Decentralized collaborative learning framework for next poi recommendation. *ACM Transactions on Information Systems, 41*, 1–25.

Long, Q., Jin, Y., Wu, Y., & Song, G. (2021). Theoretically improving graph neural networks via anonymous walk graph kernels. In *Proceedings of the web conference 2021* (pp. 1204–1214).

Long, Q., Wang, Y., Du, L., Song, G., Jin, Y., & Lin, W. (2019). Hierarchical community structure preserving network embedding: A subspace approach. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 409–418).

Long, Q., Xu, L., Fang, Z., & Song, G. (2021). Hgk-gnn: Heterogeneous graph kernel based graph neural networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 1129–1138).

Long, S., Zhou, Y., Dai, X., & Zhou, H. (2022). Zero-shot 3d drug design by sketching and generating. arXiv preprint arXiv:2209.13865.

Louizos, C., Shalit, U., Mooij, J. M., Sontag, D., Zemel, R., & Welling, M. (2017). Causal effect inference with deep latent-variable models. *Advances in Neural Information Processing Systems, 30*.

Lu, H., Gong, D., Li, Z., Liu, F., & Liu, F. (2023). Sybilhp: Sybil detection in directed social networks with adaptive homophily prediction. *Applied Sciences, 13*, 5341.

Luo, D., Cheng, W., Yu, W., Zong, B., Ni, J., Chen, H., et al. (2021). Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 779–787).

Luo, X., Gu, Y., Jiang, H., Huang, J., Ju, W., Zhang, M., et al. (2023). Graph ode with factorized prototypes for modeling complicated interacting dynamics. arXiv preprint arXiv:2311.06554.

Luo, S., Guan, J., Ma, J., & Peng, J. (2021). A 3d generative model for structure-based drug design. *Advances in Neural Information Processing Systems, 34*, 6229–6239.

Luo, Y., & Ji, S. (2022). An autoregressive flow model for 3d molecular geometry generation from scratch. In *International conference on learning representations*.

Luo, X., Ju, W., Gu, Y., Mao, Z., Liu, L., Yuan, Y., et al. (2023). Self-supervised graph-level representation learning with adversarial contrastive learning. *ACM Transactions on Knowledge Discovery from Data*.

Luo, X., Ju, W., Gu, Y., Qin, Y., Yi, S., Wu, D., et al. (2023). Towards effective semi-supervised node classification with hybrid curriculum pseudo-labeling. *ACM Transactions on Multimedia Computing, Communications and Applications*.

Luo, X., Ju, W., Qu, M., Chen, C., Deng, M., Hua, X.-S., et al. (2022). Dualgraph: Improving semi-supervised graph classification via dual contrastive learning. In *Proceedings of the IEEE international conference on data engineering* (pp. 699–712).

Luo, X., Ju, W., Qu, M., Gu, Y., Chen, C., Deng, M., et al. (2022). Clear: Cluster-enhanced contrast for self-supervised graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Luo, Y., Shi, L., Xu, M., Ji, Y., Xiao, F., Hu, C., et al. (2023). Impact-oriented contextual scholar profiling using self-citation graphs. arXiv preprint arXiv:2304.12217.

Luo, X., Wu, D., Ma, Z., Chen, C., Deng, M., Ma, J., et al. (2021). Cimon: Towards high-quality hash codes. In *Proceedings of the international joint conference on artificial intelligence*.

Luo, Y., Yan, K., & Ji, S. (2021). Graphdf: A discrete flow model for molecular graph generation. In *International conference on machine learning* (pp. 7192–7203). PMLR.

Luo, X., Yuan, J., Huang, Z., Jiang, H., Qin, Y., Ju, W., et al. (2023). Hope: High-order graph ode for modeling interacting dynamics. In *International conference on machine learning* (pp. 23124–23139). PMLR.

Luo, X., Zhao, Y., Mao, Z., Qin, Y., Ju, W., Zhang, M., et al. (2023). Rignn: A rationale perspective for semi-supervised open-world graph classification. *Transactions on Machine Learning Research*.

Luo, X., Zhao, Y., Qin, Y., Ju, W., & Zhang, M. (2023). Towards semi-supervised universal graph classification. *IEEE Transactions on Knowledge and Data Engineering*.

Ma, H., Bian, Y., Rong, Y., Huang, W., Xu, T., Xie, W., et al. (2020). Multi-view graph neural networks for molecular property prediction. arXiv preprint arXiv:2005.13607.

Ma, N., Bu, J., Yang, J., Zhang, Z., Yao, C., Yu, Z., et al. (2020). Adaptive-step graph meta-learner for few-shot graph classification. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 1055–1064).

Ma, X., Chen, Q., Wu, Y., Song, G., Wang, L., & Zheng, B. (2023). Rethinking structural encodings: Adaptive graph transformer for node classification task. In *Proceedings of the ACM web conference 2023* (pp. 533–544).

Ma, J., Deng, J., & Mei, Q. (2021). Subgroup generalization and fairness of graph neural networks. *Advances in Neural Information Processing Systems, 34*, 1048–1061.

Ma, J., Dong, Y., Huang, Z., Mietchen, D., & Li, J. (2022). Assessing the causal impact of covid-19 related policies on outbreak dynamics: A case study in the us. In *Proceedings of the ACM web conference 2022* (pp. 2678–2686).

Ma, M., & Lei, X. (2023). A dual graph neural network for drug–drug interactions prediction based on molecular structure and interactions. *PLoS Computational Biology, 19*, Article e1010812.

Ma, J., & Li, J. (2022). Learning causality with graphs. *AI Magazine, 43*, 365–375.

Ma, L., Lin, C., Lim, D., Romero-Soriano, A., Dokania, P. K., Coates, M., et al. (2023). Graph inductive biases in transformers without message passing. arXiv preprint arXiv:2305.17589.

Ma, Y., Wang, S., Aggarwal, C. C., & Tang, J. (2019). Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 723–731).

Ma, C., & Xia, H. (2023). Author name disambiguation based on heterogeneous graph. *Journal of Computers, 34*, 41–52.

Madhawa, K., Ishiguro, K., Nakago, K., & Abe, M. (2019). Graphnvp: An invertible flow model for molecular graph generation. arXiv preprint arXiv:1905.11600.

Madumal, P., Miller, T., Sonenberg, L., & Vetere, F. (2020). Explainable reinforcement learning through a causal lens. *Vol. 34, In Proceedings of the AAAI conference on artificial intelligence* (pp. 2493–2500).

Man, T., Shen, H., Liu, S., Jin, X., & Cheng, X. (2016). Predict anchor links across social networks via an embedding approach. *Vol. 16, In Ijcai* (pp. 1823–1829).

Manguri, K. H., Ramadhan, R. N., & Amin, P. R. M. (2020). Twitter sentiment analysis on worldwide covid-19 outbreaks. *Kurdistan Journal of Applied Research*, 54–65.

Mansimov, E., Mahmood, O., Kang, S., & Cho, K. (2019). Molecular geometry prediction using a deep generative graph neural network. *Scientific Reports, 9*, 1–13.

MansourLakouraj, M., Gautam, M., Livani, H., & Benidris, M. (2022). A multi-rate sampling pmu-based event classification in active distribution grids with spectral graph neural network. *Electric Power Systems Research, 211*, Article 108145.

Mao, Z., Ju, W., Qin, Y., Luo, X., & Zhang, M. (2023). Rahnet: Retrieval augmented hybrid network for long-tailed graph classification. In *Proceedings of the 31st ACM international conference on multimedia* (pp. 3817–3826).

Mao, Q., Liu, Z., Liu, C., & Sun, J. (2023). Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM web conference 2023* (pp. 599–610).

Margaris, D., Vassilakis, C., & Spiliotopoulos, D. (2019). Handling uncertainty in social media textual information for improving venue recommendation formulation quality in social networks. *Social Network Analysis and Mining, 9*, 1–19.

Maziarka, Łukasz, Danel, T., Mucha, S., Rataj, K., Tabor, J., & Jastrzębski, S. (2020). Molecule attention transformer.

Mazuz, E., Shtar, G., Shapira, B., & Rokach, L. (2023). Molecule generation using transformers and policy gradient reinforcement learning. *Scientific Reports, 13*, 8799.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys, 54*, 1–35.

Meng, X., Wang, S., Liang, Z., Yao, D., Zhou, J., & Zhang, Y. (2021). Semi-supervised anomaly detection in dynamic communication networks. *Information Sciences, 571*, 527–542.

Meng, X.-Y., Zhang, H.-X., Mezei, M., & Cui, M. (2011). Molecular docking: a powerful approach for structure-based drug discovery. *Current Computer-Aided Drug Design, 7*, 146–157.

Meyers, J., Fabian, B., & Brown, N. (2021). De novo molecular design and generative models. *Drug Discovery Today, 26*, 2707–2715.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems, 26*.

Mohamed, A., Qian, K., Elhoseiny, M., & Claudel, C. (2020). Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14424–14432).

Morehead, A., & Cheng, J. (2022). Geometry-complete perceptron networks for 3d molecular graphs. arXiv preprint arXiv:2211.02504.

Morehead, A., & Cheng, J. (2023). Geometry-complete diffusion for 3d molecule generation. arXiv preprint arXiv:2302.04313.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., et al. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. *Vol. 33, In Proceedings of the AAAI conference on artificial intelligence* (pp. 4602–4609).

Moscato, V., & Sperlì, G. (2021). A survey about community detection over online social and heterogeneous information networks. *Knowledge-Based Systems, 224*, Article 107112.

Müller, M. (2007). Dynamic time warping. In *Information retrieval for music and motion* (pp. 69–84).

Nesterov, V., Wieser, M., & Roth, V. (2020). 3Dmolnet: a generative network for molecular structures. arXiv preprint arXiv:2010.06477.

OpenAI (2023). Gpt-4 technical report. arXiv:2303.08774.

Oskarsson, J., Sidén, P., & Lindsten, F. (2023). Temporal graph neural networks for irregular data. In F. Ruiz, J. Dy, & J.-W. van de Meent (Eds.), *Proceedings of machine learning research*: *vol. 206, Proceedings of the 26th international conference on artificial intelligence and statistics* (pp. 4515–4531). PMLR.

Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1105–1114).

Pan, X., Hou, F., & Li, S. (2022). Traffic speed prediction based on time classification in combination with spatial graph convolutional network. *IEEE Transactions on Intelligent Transportation Systems*.

Panagopoulos, G., Malliaros, F., & Vazirgiannis, M. (2020). Multi-task learning for influence estimation and maximization. *IEEE Transactions on Knowledge and Data Engineering*.

Park, H., Lee, S., Hwang, D., Jeong, J., Kim, K.-M., Ha, J.-W., et al. (2021). Learning augmentation for gnns with consistency regularization. *IEEE Access, 9*, 127961–127972.

Paulos, E., Anderson, K., & Townsend, A. (2004). Ubicomp in the urban frontier.

Paulos, E., & Goodman, E. (2004). The familiar stranger: anxiety, comfort, and play in public places. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 223–230).

Peng, X., Guan, J., Peng, J., & Ma, J. (2023). Pocket-specific 3d molecule generation by fragment-based autoregressive diffusion models. URL: https://openreview.net/forum?id=HGsoe1wmRW5.

Peng, X., Luo, S., Guan, J., Xie, Q., Peng, J., & Ma, J. (2022). Pocket2mol: Efficient molecular sampling based on 3d protein pockets. arXiv preprint arXiv:2205.07249.

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

Perozzi, B., Kulkarni, V., Chen, H., & Skiena, S. (2017). Don't walk, skip! online learning of multi-scale network embeddings. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017* (pp. 258–265).

Pilco, D. S., & Rivera, A. R. (2019). Graph learning network: A structure learning algorithm. arXiv preprint arXiv:1905.12665.

Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. *Signal Processing, 99*, 215–249.

Pinheiro, G. A., Da Silva, J. L., & Quiles, M. G. (2022). Smiclr: Contrastive learning on multiple molecular representations for semisupervised and unsupervised representation learning. *Journal of Chemical Information and Modeling, 62*, 3948–3960.

Preuer, K., Klambauer, G., Rippmann, F., Hochreiter, S., & Unterthiner, T. (2019). Interpretable deep learning in drug discovery. In *Explainable AI: interpreting, explaining and visualizing deep learning* (pp. 331–345). Springer.

Qiao, Z., Du, Y., Fu, Y., Wang, P., & Zhou, Y. (2019). Unsupervised author disambiguation using heterogeneous graph convolutional network embedding. In *2019 IEEE international conference on big data* (pp. 910–919). IEEE.

Qiao, Z., Fu, Y., Wang, P., Xiao, M., Ning, Z., Du, Y., et al. (2022). Rpt: Toward transferable model on heterogeneous researcher data via pre-training. *IEEE Transactions on Big Data*.

Qiao, Z., Wang, P., Fu, Y., Du, Y., Wang, P., & Zhou, Y. (2020). Tree structure-aware graph representation learning via integrated hierarchical aggregation and relational metric learning. In *2020 IEEE international conference on data mining* (pp. 432–441). IEEE.

Qiao, Z., Welborn, M., Anandkumar, A., Manby, F. R., & Miller III, T. F. (2020). Orbnet: Deep learning for quantum chemistry using symmetry-adapted atomic-orbital features. *The Journal of Chemical Physics, 153*, Article 124111.

Qin, Y., Ju, W., Wu, H., Luo, X., & Zhang, M. (2024). Learning graph ode for continuous-time sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Qin, Y., Wang, Y., Sun, F., Ju, W., Hou, X., Wang, Z., et al. (2022). Disenpoi: Disentangling sequential and geographical influence for point-of-interest recommendation. arXiv preprint arXiv:2210.16591.

Qin, Y., Wu, H., Ju, W., Luo, X., & Zhang, M. (2023). A diffusion model for poi recommendation. arXiv preprint arXiv:2304.07041.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., et al. (2020). Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1150–1160).

Qiu, J., Du, Q., Yin, K., Zhang, S.-L., & Qian, C. (2020). A causality mining and knowledge graph based method of root cause diagnosis for performance anomaly in cloud applications. *Applied Sciences, 10*, 2166.

Qiu, R., Li, J., Huang, Z., & Yin, H. (2019). Rethinking the item order in session-based recommendation with graph neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 579–588).

Qiu, R., Yin, H., Huang, Z., & Chen, T. (2020). Gag: Global attributed graph neural network for streaming session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 669–678).

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning* (pp. 8748–8763). PMLR.

Ragoza, M., Masuda, T., & Koes, D. R. (2022). Generating 3d molecules conditional on receptor binding sites with deep generative models. *Chemical Science, 13*, 2701–2713.

Ran, B., & Boyce, D. (2012). *Modeling dynamic transportation networks: an intelligent transportation system oriented approach*. Springer Science & Business Media.

Ranjan, E., Sanyal, S., & Talukdar, P. (2020). Asap: Adaptive structure aware pooling for learning hierarchical graph representations. *Vol. 34, In Proceedings of the AAAI conference on artificial intelligence* (pp. 5470–5477).

Rao, X., Wang, H., Zhang, L., Li, J., Shang, S., & Han, P. (2022). Fogs: First-order gradient supervision with learning-based graph for traffic flow forecasting. In *Proceedings of international joint conference on artificial intelligence*. ijcai. org.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). Bpr: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618.

Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning* (pp. 1530–1538). PMLR.

Richardson, M., & Domingos, P. (2002). Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 61–70).

Richens, J. G., Lee, C. M., & Johri, S. (2020). Improving the accuracy of medical diagnosis with causal machine learning. *Nature Communications, 11*, 1–9.

Roney, J. P., Maragakis, P., Skopp, P., & Shaw, D. E. (2021). Generating realistic 3d molecules with an equivariant conditional likelihood model.

Rosa, R. L., Schwartz, G. M., Ruggiero, W. V., & Rodríguez, D. Z. (2018). A knowledge-based recommendation system that includes sentiment analysis and deep learning. *IEEE Transactions on Industrial Informatics, 15*, 2124–2135.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*, 2323–2326.

Ruddigkeit, L., Van Deursen, R., Blum, L. C., & Reymond, J.-L. (2012). Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling, 52*, 2864–2875.

Runwal, B., Kumar, S., et al. (2022). Robustifying gnn via weighted laplacian. In *2022 IEEE international conference on signal processing and communications* (pp. 1–5). IEEE.

Ryu, S., Lim, J., Hong, S. H., & Kim, W. Y. (2018). Deeply learning molecular structure–property relationships using attention-and gate-augmented graph convolutional network. arXiv preprint arXiv:1805.10988.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. *Advances in Neural Information Processing Systems, 30*.

Sadhasivam, S., Valarmathie, P., & Dinakaran, K. (2023). Malicious activities prediction over online social networking using ensemble model. *Intelligent Automation & Soft Computing, 36*, 461–479.

Sandryhaila, A., & Moura, J. M. (2013). Discrete signal processing on graphs. *IEEE Transactions on Signal Processing, 61*, 1644–1656.

Sarkar, S., & Alvari, H. (2020). Mitigating bias in online microfinance platforms: A case study on kiva. org. In *ECML PKDD 2020 workshops: workshops of the European conference on machine learning and knowledge discovery in databases (ECML PKDD 2020): SoGood 2020, PDFL 2020, MLCS 2020, NFMCP 2020, DINA 2020, EDML 2020, XKDD 2020 and INRA 2020, Ghent, Belgium, September 14–18, 2020, proceedings* (pp. 75–91). Springer.

Satorras, V. G., Hoogeboom, E., & Welling, M. (2021). E(n) equivariant graph neural networks. arXiv preprint arXiv:2102.09844.

Saxena, C., Liu, T., & King, I. (2020). A survey of graph curvature and embedding in non-euclidean spaces. In *International conference on neural information processing* (pp. 127–139). Springer.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks, 20*, 61–80.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European semantic web conference* (pp. 593–607). Springer.

Schneider, P., Walters, W. P., Plowright, A. T., Sieroka, N., Listgarten, J., Goodnow, R. A., et al. (2020). Rethinking drug design in the artificial intelligence era. *Nature Reviews Drug Discovery, 19*, 353–364.

Schneuing, A., Du, Y., Harris, C., Jamasb, A., Igashov, I., Du, W., et al. (2022). Structure-based drug design with equivariant diffusion models. arXiv preprint arXiv:2210.13695.

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., & Müller, K.-R. (2018). Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics, 148*, Article 241722.

Schütt, K., Unke, O., & Gastegger, M. (2021). Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *ICML*. URL: https://proceedings.mlr.press/v139/schutt21a.html.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine, 29*, 93.

Shalit, U., Johansson, F. D., & Sontag, D. (2017). Estimating individual treatment effect: generalization bounds and algorithms. In *International conference on machine learning* (pp. 3076–3085). PMLR.

Shawe-Taylor, J., Cristianini, N., et al. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011a). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research, 12*, 2539–2561.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011b). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research, 12*.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., & Borgwardt, K. (2009). Efficient graphlet kernels for large graph comparison. In *Proceedings of international conference on artificial intelligence and statistics* (pp. 488–495).

Shi, C., Li, Y., Zhang, J., Sun, Y., & Philip, S. Y. (2016). A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering, 29*, 17–37.

Shi, C., Luo, S., Xu, M., & Tang, J. (2021). Learning gradient fields for molecular conformation generation. *Vol. 139, In Proceedings of the 38th international conference on machine learning* (pp. 9558–9568).

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., & Tang, J. (2020). Graphaf: a flow-based autoregressive model for molecular graph generation. arXiv preprint arXiv:2001.09382.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine, 30*, 83–98.

Si, Y., Zhang, F., & Liu, W. (2019). An adaptive point-of-interest recommendation method for location-based social networks based on user activity and spatial features. *Knowledge-Based Systems, 163*, 267–282.

Silva, T. H., Viana, A. C., Benevenuto, F., Villas, L., Salles, J., Loureiro, A., et al. (2019). Urban computing leveraging location-based social network data: a survey. *ACM Computing Surveys, 52*, 1–39.

Simonovsky, M., & Komodakis, N. (2017). Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3693–3702).

Simonovsky, M., & Komodakis, N. (2018). Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks* (pp. 412–422). Springer.

Sofuoglu, S. E., & Aviyente, S. (2022). Gloss: Tensor-based anomaly detection in spatiotemporal urban traffic data. *Signal Processing, 192*, Article 108370.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning* (pp. 2256–2265). PMLR.

Song, Y., & Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems, 32*.

Song, C., Lin, Y., Guo, S., & Wan, H. (2020). Spatial–temporal synchronous graph convolutional networks: A new framework for spatial–temporal network data forecasting. *Vol. 34, In Proceedings of the AAAI conference on artificial intelligence* (pp. 914–921).

Song, Q., Ming, R., Hu, J., Niu, H., & Gao, M. (2020). Graph attention convolutional network: Spatiotemporal modeling for urban traffic prediction. In *2020 IEEE 23rd international conference on intelligent transportation systems* (pp. 1–6). IEEE.

Spackman, P. R., Jayatilaka, D., & Karton, A. (2016). Basis set convergence of ccsd(t) equilibrium geometries using a large and diverse set of molecular structures. *The Journal of Chemical Physics, 145*, Article 104101.

Stärk, H., Ganea, O., Pattanaik, L., Barzilay, R., & Jaakkola, T. (2022). Equibind: Geometric deep learning for drug binding structure prediction. In *International conference on machine learning* (pp. 20503–20521). PMLR.

Su, B., Du, D., Yang, Z., Zhou, Y., Li, J., Rao, A., et al. (2022). A molecular multimodal foundation model associating molecule graphs with natural language. arXiv preprint arXiv:2209.05481.

Sugiyama, K., & Kan, M.-Y. (2010). Scholarly paper recommendation via user's recent research interests. In *Proceedings of the 10th annual joint conference on Digital libraries* (pp. 29–38).

Sun, F.-Y., Hoffmann, J., Verma, V., & Tang, J. (2020). Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Proceedings of the international conference on learning representations*.

Sun, J., Jiang, Q., & Lu, C. (2020). Recursive social behavior graph for trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 660–669).

Sun, Q., Li, J., Peng, H., Wu, J., Fu, X., Ji, C., et al. (2022). Graph structure learning with variational information bottleneck. *Vol. 36, In Proceedings of the AAAI conference on artificial intelligence* (pp. 4165–4174).

Sun, X., Wang, Z., Yang, J., & Liu, X. (2020). Deepdom: Malicious domain detection with scalable and heterogeneous graph convolutional networks. *Computers & Security, 99*, Article 102057.

Sun, F., Zhan, Z., Guo, H., Zhang, M., & Tang, J. (2023). Graphvf: Controllable protein-specific 3d molecule generation with variational flow. arXiv preprint arXiv:2304.12825.

Sweeney, L. (2013). Discrimination in online ad delivery. *Communications of the ACM, 56*, 44–54.

Tabassum, S., Pereira, F. S., Fernandes, S., & Gama, J. (2018). Social network analysis: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8*, Article e1256.

Tailor, S. A., Opolka, F., Lio, P., & Lane, N. D. (2022). Do we need anisotropic graph neural networks? In *International conference on learning representations*.

Takahashi, T., & Igata, N. (2012). Rumor detection on twitter. In *The 6th international conference on soft computing and intelligent systems, and the 13th international symposium on advanced intelligence systems* (pp. 452–457). IEEE.

Tan, C., Gao, Z., & Li, S. Z. (2023). Target-aware molecular graph generation. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 410–427). Springer.

Tan, Q., Liu, N., Huang, X., Choi, S.-H., Li, L., Chen, R., et al. (2023). S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking. In *Proceedings of the sixteenth ACM international conference on web search and data mining* (pp. 787–795).

Tang, H., Ji, D., Li, C., & Zhou, Q. (2020). Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 6578–6588).

Tang, X., Li, Y., Sun, Y., Yao, H., Mitra, P., & Wang, S. (2020). Transferring robustness for graph neural network against poisoning attacks. In *Proceedings of the 13th international conference on web search and data mining* (pp. 600–608).

Tang, J., Qu, M., & Mei, Q. (2015). Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1165–1174).

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).

Thölke, P., & Fabritiis, G. D. (2022). Equivariant transformers for neural network based molecular potentials. In *ICLR*. URL: https://openreview.net/forum?id=zNHzqZ9wrRB.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., et al. (2018). Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. arXiv preprint arXiv:1802.08219.

Thys, S., Van Ranst, W., & Goedemé, T. (2019). Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*.

Toivonen, H., Srinivasan, A., King, R. D., Kramer, S., & Helma, C. (2003). Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics*, *19*, 1183–1193.

Uma Maheswari, S., & Dhenakaran, S. (2023). Opinion mining on integrated social networks and e-commerce blog. *IETE Journal of Research*, *69*, 2080–2088.

Unankard, S., Li, X., Sharaf, M., Zhong, J., & Li, X. (2014). Predicting elections from social networks based on sub-event detection and sentiment analysis. In *International conference on web information systems engineering* (pp. 1–16). Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Verma, J., Khedkar, V. M., & Coutinho, E. C. (2010). 3D-qsar in drug design-a review. *Current Topics in Medicinal Chemistry*, *10*, 95–115.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., & Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, *29*.

Walsh, D. J., Zou, W., Schneider, L., Mello, R., Deagen, M. E., Mysona, J., et al. (2023). Community resource for innovation in polymer technology (cript): A scalable polymer material data structure.

Walters, W. P., Stahl, M. T., & Murcko, M. A. (1998). Virtual screening—an overview. *Drug Discovery Today*, *3*, 160–178.

Wan, G., & Kokel, H. (2021). Graph sparsification via meta-learning. In *DLG@ AAAI*.

Wang, Y., Abuduweili, A., Yao, Q., & Dou, D. (2021). Property-aware relation networks for few-shot molecular property prediction. *Advances in Neural Information Processing Systems*, *34*, 17441–17454.

Wang, X., Bo, D., Shi, C., Fan, S., Ye, Y., & Philip, S. Y. (2022). A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data*.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1225–1234).

Wang, J., Ding, K., Hong, L., Liu, H., & Caverlee, J. (2020). Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 1101–1110).

Wang, R., Fang, X., Lu, Y., Yang, C.-Y., & Wang, S. (2005). The pdbbind database: methodologies and updates. *Journal of Medicinal Chemistry*, *48*, 4111–4119.

Wang, X., Fukumoto, F., Li, J., Yu, D., & Sun, X. (2023). Statrl: Spatial–temporal and text representation learning for poi recommendation. *Applied Intelligence*, *53*, 8286–8301.

Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S. (2019). Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval* (pp. 165–174).

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., et al. (2019). Heterogeneous graph attention network. In *The world wide web conference* (pp. 2022–2032).

Wang, X., Jin, B., Du, Y., Cui, P., Tan, Y., & Yang, Y. (2021). One-class graph neural networks for anomaly detection in attributed networks. *Neural Computing and Applications*, *33*, 12073–12085.

Wang, X., Jin, H., Zhang, A., He, X., Xu, T., & Chua, T.-S. (2020). Disentangled graph collaborative filtering. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 1001–1010).

Wang, K., Lin, Y.-A., Weissmann, B., Savva, M., Chang, A. X., & Ritchie, D. (2019). Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics*, *38*, 1–15.

Wang, S., Miao, H., Chen, H., & Huang, Z. (2020). Multi-task adversarial spatial–temporal networks for crowd flow prediction. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 1555–1564).

Wang, Z., Nie, W., Qiao, Z., Xiao, C., Baraniuk, R., & Anandkumar, A. (2022). Retrieval-based controllable molecule generation. arXiv preprint arXiv:2208.11126.

Wang, C., Pan, S., Long, G., Zhu, X., & Jiang, J. (2017). Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 889–898).

Wang, Y., Qin, Y., Sun, F., Zhang, B., Hou, X., Hu, K., et al. (2022). Disenctr: Dynamic graph-based disentangled representation for click-through rate prediction. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 2314–2318).

Wang, Y., Song, Y., Li, S., Cheng, C., Ju, W., Zhang, M., et al. (2022). Disencite: Graph-based disentangled representation learning for context-specific citation generation. *Vol. 36*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 11449–11458).

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *Acm Transactions on Graphics (tog)*, *38*, 1–12.

Wang, Y., Wang, J., Cao, Z., & Barati Farimani, A. (2022). Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, *4*, 279–287.

Wang, Z., Wei, W., Cong, G., Li, X.-L., Mao, X.-L., & Qiu, M. (2020). Global context enhanced graph neural networks for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 169–178).

Wang, G., Ying, R., Huang, J., & Leskovec, J. (2020). Multi-hop attention graph neural network. arXiv preprint arXiv:2009.14332.

Wang, Z., Zeng, J., Wen, J., Gao, M., & Zhou, W. (2023). Point-of-interest recommendation using deep semantic model. *Expert Systems with Applications*, Article 120727.

Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., & Guo, M. (2019). Multi-task feature learning for knowledge graph enhanced recommendation. In *The world wide web conference* (pp. 2000–2010).

Wang, Y., Zhao, Y., Shah, N., & Derr, T. (2022). Imbalanced graph classification via graph-of-graph neural networks. In *Proceedings of the 31st ACM international conference on information & knowledge management* (pp. 2067–2076).

Wang, Y., Zhao, Y., Zhang, Y., & Derr, T. (2023). Collaboration-aware graph convolutional network for recommender systems. In *Proceedings of the ACM web conference 2023* (pp. 91–101).

Wang, J., Zhu, L., Dai, T., & Wang, Y. (2020). Deep memory network with bi-lstm for personalized context-aware citation recommendation. *Neurocomputing*, *410*, 103–113.

Wang, Z., Zhu, Y., Zhang, Q., Liu, H., Wang, C., & Liu, T. (2022). Graph-enhanced spatial–temporal network for next poi recommendation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *16*, 1–21.

Wei, Y., Wang, X., Nie, L., He, X., Hong, R., & Chua, T.-S. (2019). Mmgcn: Multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM international conference on multimedia* (pp. 1437–1445).

Weisfeiler, B., & Leman, A. (1968). The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, *2*, 12–16.

Wen, G., Cao, P., Liu, L., Yang, J., Zhang, X., Wang, F., et al. (2023). Graph self-supervised learning with application to brain networks analysis. *IEEE Journal of Biomedical and Health Informatics*.

Weng, W., Fan, J., Wu, H., Hu, Y., Tian, H., Zhu, F., et al. (2023). A decomposition dynamic graph convolutional recurrent network for traffic forecasting. *Pattern Recognition*, *142*, Article 109670.

Werneck, H., Silva, N., Viana, M. C., Mourão, F., Pereira, A. C., & Rocha, L. (2020). A survey on point-of-interest recommendation in location-based social networks. In *Proceedings of the Brazilian symposium on multimedia and the web* (pp. 185–192).

Wieder, O., Kohlbacher, S., Kuenemann, M., Garon, A., Ducrot, P., Seidel, T., et al. (2020). A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, *37*, 1–12.

Williams, C. K., & Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems* (pp. 682–688).

Withnall, M., Lindelöf, E., Engkvist, O., & Chen, H. (2020). Building attention and edge message passing neural networks for bioactivity and physical–chemical property prediction. *Journal of Cheminformatics*, *12*, 1–18.

Wu, J., Chen, X., Xu, K., & Li, S. (2022). Structural entropy guided graph hierarchical pooling. In *International conference on machine learning* (pp. 24017–24030). PMLR.

Wu, Z., Jain, P., Wright, M., Mirhoseini, A., Gonzalez, J. E., & Stoica, I. (2021). Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, *34*, 13266–13279.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*, 4–24.

Wu, Z., Pan, S., Long, G., Jiang, J., & Zhang, C. (2019). Graph wavenet for deep spatial–temporal graph modeling. In *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 1907–1913).

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., et al. (2018). Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, *9*, 513–530.

Wu, W., Shi, X., He, L., & Jin, H. (2023). Turbomgnn: Improving concurrent gnn training tasks on gpu with fine-grained kernel fusion. *IEEE Transactions on Parallel and Distributed Systems*.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning* (pp. 6861–6871). PMLR.

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., & Tan, T. (2019). Session-based recommendation with graph neural networks. *Vol. 33*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 346–353).

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., & Zhu, L. (2019). Adversarial examples on graph data: Deep insights into attack and defense. arXiv preprint arXiv:1903.01610.

Xia, L., Huang, C., Huang, C., Lin, K., Yu, T., & Kao, B. (2023). Automated self-supervised learning for recommendation. In *Proceedings of the ACM web conference 2023* (pp. 992–1002).

Xia, L., Huang, C., Xu, Y., Zhao, J., Yin, D., & Huang, J. (2022). Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 70–79).

Xia, F., Sun, K., Yu, S., Aziz, A., Wan, L., Pan, S., et al. (2021). Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, *2*, 109–127.

Xia, J., Wu, L., Chen, J., Hu, B., & Li, S. Z. (2022). Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM web conference 2022* (pp. 1070–1079).

Xie, P., Li, T., Liu, J., Du, S., Yang, X., & Zhang, J. (2020). Urban flow prediction from spatiotemporal data using machine learning: A survey. *Information Fusion*, *59*, 1–12.

Xie, Y., Liang, Y., Gong, M., Qin, A., Ong, Y.-S., & He, T. (2022). Semisupervised graph neural networks for graph classification. *IEEE Transactions on Cybernetics*.

Xie, Y., Lv, S., Qian, Y., Wen, C., & Liang, J. (2022). Active and semi-supervised graph neural networks for graph classification. *IEEE Transactions on Big Data*.

Xie, Y., Shi, C., Zhou, H., Yang, Y., Zhang, W., Yu, Y., et al. (2021). Mars: Markov molecular sampling for multi-objective drug discovery. arXiv preprint arXiv:2103. 10432.

Xie, Y., Xu, Z., Zhang, J., Wang, Z., & Ji, S. (2022). Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xiong, H., Vahedian, A., Zhou, X., Li, Y., & Luo, J. (2018). Predicting traffic congestion propagation patterns: A propagation graph approach. In *Proceedings of the 11th ACM SIGSPAtIAL international workshop on computational transportation science* (pp. 60–69).

Xu, J., Chen, J., You, S., Xiao, Z., Yang, Y., & Lu, J. (2021). Robustness of deep learning models on graphs: A survey. *AI Open*, *2*, 69–78.

Xu, C., Ding, A. S., & Zhao, K. (2021). A novel poi recommendation method based on trust relationship and spatial–temporal factors. *Electronic Commerce Research and Applications*, *48*, Article 101060.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? In *International conference on learning representations*.

Xu, C., Huang, H., & Yoo, S. (2019). Scalable causal graph learning through a deep neural network. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 1853–1862).

Xu, Z., Kang, Y., Cao, Y., & Li, Z. (2020). Spatiotemporal graph convolution multifusion network for urban vehicle emission prediction. *IEEE Transactions on Neural Networks and Learning Systems*, *32*, 3342–3354.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning* (pp. 5453–5462). PMLR.

Xu, C., Liu, D., & Mei, X. (2021). Exploring an efficient poi recommendation model based on user characteristics and spatial–temporal factors. *Mathematics*, *9*, 2673.

Xu, B., Shen, H., Cao, Q., Qiu, Y., & Cheng, X. (2019). Graph wavelet neural network. In *International conference on learning representations*.

Xu, X., Wang, H., Lal, A., Gunter, C. A., & Li, B. (2023). Edog: Adversarial edge detection for graph neural networks. In *2023 IEEE conference on secure and trustworthy machine learning* (pp. 291–305). IEEE.

Xu, Z., Wang, S., Zhu, F., & Huang, J. (2017). Seq2seq fingerprint: An unsupervised deep molecular embedding for drug discovery. In *Proceedings of the 8th ACM international conference on bioinformatics, computational biology, and health informatics* (pp. 285–294).

Xu, Z., Wei, P., Liu, S., Zhang, W., Wang, L., & Zheng, B. (2023). Correlative preference transfer with hierarchical hypergraph network for multi-domain recommendation. In *Proceedings of the ACM web conference 2023* (pp. 983–991).

Xu, C., Zhao, P., Liu, Y., Sheng, V. S., Xu, J., Zhuang, F., et al. (2019). Graph contextualized self-attention network for session-based recommendation. *Vol. 19*, In *IJCAI* (pp. 3940–3946).

Yan, X., Song, T., Jiao, Y., He, J., Wang, J., Li, R., et al. (2023). Spatio-temporal hypergraph learning for next poi recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval* (pp. 403–412).

Yang, H., Chen, H., Pan, S., Li, L., Yu, P. S., & Xu, G. (2022). Dual space graph contrastive learning. In *Proceedings of the web conference* (pp. 1238–1247).

Yang, Y., Heppenstall, A., Turner, A., & Comber, A. (2020). Using graph structural information about flows to enhance short-term demand prediction in bike-sharing systems. *Computers, Environment and Urban Systems*, *83*, Article 101521.

Yang, Y., Huang, C., Xia, L., & Huang, C. (2023). Knowledge graph self-supervised rationalization for recommendation. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 3046–3056).

Yang, Y., Huang, C., Xia, L., Liang, Y., Yu, Y., & Li, C. (2022). Multi-behavior hypergraph-enhanced transformer for sequential recommendation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 2263–2274).

Yang, Y., Liang, Y., & Zhang, M. (2022). Pa-gnn: Parameter-adaptive graph neural networks.

Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., et al. (2019). Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, *59*, 3370–3388.

Yang, J., Xu, H., Mirzoyan, S., Chen, T., Liu, Z., Ju, W., et al. (2023). Poisoning scientific knowledge using large language models. *bioRxiv*, 2023-11.

Yang, Z., Zhang, G., Wu, J., Yang, J., Sheng, Q. Z., Xue, S., et al. (2023). A comprehensive survey of graph-level learning. arXiv preprint arXiv:2301.05860.

Yang, C., Zhou, K., & Liu, J. (2021). Supergraph: Spatial–temporal graph-based feature extraction for rotating machinery diagnosis. *IEEE Transactions on Industrial Electronics*, *69*, 4167–4176.

Yao, S., Wang, T., & Wan, X. (2020). Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7145–7154).

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., et al. (2018). Deep multi-view spatial–temporal network for taxi demand prediction. *Vol. 32*, In *Proceedings of the AAAI conference on artificial intelligence*.

Yi, S., Ju, W., Qin, Y., Luo, X., Liu, L., Zhou, Y., et al. (2023). Redundancy-free self-supervised relational learning for graph clustering. *IEEE Transactions on Neural Networks and Learning Systems*.

Yi, S.-Y., Mao, Z., Ju, W., Zhou, Y.-D., Liu, L., Luo, X., et al. (2023). Towards long-tailed recognition for graph classification via collaborative experts. *IEEE Transactions on Big Data*.

Yildirimoglu, M., & Kim, J. (2018). Identification of communities in urban mobility networks using multi-layer graphs of network traffic. *Transportation Research Part C (Emerging Technologies)*, *89*, 254–267.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., & Leskovec, J. (2019). Gnnexplainer: Generating explanations for graph neural networks. In *Proceedings of the conference on neural information processing systems*.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., et al. (2021). Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, *34*, 28877–28888.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems*, *31*.

You, Y., Chen, T., Wang, Z., & Shen, Y. (2020). When does self-supervision help graph convolutional networks? In *International conference on machine learning* (pp. 10871–10880). PMLR.

You, J., Liu, B., Ying, Z., Pande, V., & Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation. *Advances in Neural Information Processing Systems*, *31*.

Yu, L., Du, B., Hu, X., Sun, L., Han, L., & Lv, W. (2021). Deep spatio-temporal graph convolutional network for traffic accident prediction. *Neurocomputing*, *423*, 135–147.

Yu, Z., & Gao, H. (2022). Molecular representation learning via heterogeneous motif graph neural networks. In *International conference on machine learning* (pp. 25581–25594). PMLR.

Yu, X., Gu, Q., Zhou, M., & Han, J. (2012). Citation prediction in heterogeneous bibliographic networks. In *Proceedings of the 2012 SIAM international conference on data mining* (pp. 1119–1130). SIAM.

Yu, S., Liu, J., Yang, Z., Chen, Z., Jiang, H., Tolba, A., et al. (2018). Pave: Personalized academic venue recommendation exploiting co-publication networks. *Journal of Network and Computer Applications*, *104*, 38–47.

Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., & Nguyen, Q. V. H. (2022). Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval* (pp. 1294–1303).

Yu, B., Yin, H., & Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875.

Yu, Z., Yu, J., Cui, Y., Tao, D., & Tian, Q. (2019). Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6281–6290).

Yu, D., Zhang, R., Jiang, Z., Wu, Y., & Yang, Y. (2021). Graph-revised convolutional network. In *Machine learning and knowledge discovery in databases: European conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, proceedings, part III* (pp. 378–393). Springer.

Yuan, Y., Ding, J., Wang, H., Jin, D., & Li, Y. (2022). Activity trajectory generation via modeling spatiotemporal dynamics. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 4752–4762). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3534678.3542671.

Yuan, J., Luo, X., Qin, Y., Mao, Z., Ju, W., & Zhang, M. (2023). Alex: Towards effective graph transfer learning with noisy labels. In *Proceedings of the 31st ACM international conference on multimedia* (pp. 3647–3656).

Yuan, J., Luo, X., Qin, Y., Zhao, Y., Ju, W., & Zhang, M. (2023). Learning on graphs under label noise. In *ICASSP 2023-2023 IEEE international conference on acoustics, speech and signal processing* (pp. 1–5). IEEE.

Yue, H., Zhang, C., Zhang, C., & Liu, H. (2022). Label-invariant augmentation for semi-supervised graph classification. *Advances in Neural Information Processing Systems*, *35*, 29350–29361.

Zang, C., & Wang, F. (2020). Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 617–626).

Zang, X., Zhao, X., & Tang, B. (2023). Hierarchical molecular graph self-supervised learning for property prediction. *Communications Chemistry*, *6*, 34.

Zeng, D., Liu, W., Chen, W., Zhou, L., Zhang, M., & Qu, H. (2023). Substructure aware graph neural networks. *Vol. 37*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 11129–11137).

Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., et al. (2019). Hierarchical graph pooling with structure learning. arXiv preprint arXiv:1911.05954.

Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018). An end-to-end deep learning architecture for graph classification. *Vol. 32, In Proceedings of the AAAI conference on artificial intelligence*.

Zhang, Z., Cui, P., & Zhu, W. (2020). Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Zhang, Z., Jia, J., Wang, B., & Gong, N. Z. (2021). Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM symposium on access control models and technologies* (pp. 15–26).

Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (pp. 83–92).

Zhang, M., & Li, P. (2021). Nested graph neural networks. *Advances in Neural Information Processing Systems*, *34*, 15734–15747.

Zhang, C., Li, Q., & Song, D. (2019). Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing* (pp. 4568–4578).

Zhang, C., Li, W., Wei, D., Liu, Y., & Li, Z. (2022). Network dynamic gcn influence maximization algorithm with leader fake labeling mechanism. *IEEE Transactions on Computational Social Systems*.

Zhang, Z., Liu, Q., Lee, C.-K., Hsieh, C.-Y., & Chen, E. (2023). An equivariant generative framework for molecular graph-structure co-design. *Chemical Science*, *14*, 8380–8392.

Zhang, Z., Liu, Q., Wang, H., Lu, C., & Lee, C.-K. (2021). Motif-based graph self-supervised learning for molecular property prediction. *Advances in Neural Information Processing Systems*, *34*, 15870–15882.

Zhang, B., Luo, S., Wang, L., & He, D. (2023). Rethinking the expressive power of gnns via graph biconnectivity. arXiv preprint arXiv:2301.09505.

Zhang, Z., Min, Y., Zheng, S., & Liu, Q. (2022). Molecule generation for target protein binding with structural motifs. In *The eleventh international conference on learning representations*.

Zhang, Y., Pal, S., Coates, M., & Ustebay, D. (2019). Bayesian graph convolutional neural networks for semi-supervised classification. *Vol. 33, In Proceedings of the AAAI conference on artificial intelligence* (pp. 5829–5836).

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., & Yeung, D.-Y. (2018). Gaan: Gated attention networks for learning on large and spatiotemporal graphs. arXiv preprint arXiv:1803.07294.

Zhang, C., Song, D., Huang, C., Swami, A., & Chawla, N. V. (2019). Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 793–803).

Zhang, Y., Tiňo, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Zhang, Z., Wang, M., Xiang, Y., Huang, Y., & Nehorai, A. (2018). Retgk: Graph kernels based on return probabilities of random walks. In *Advances in neural information processing systems* (pp. 3964–3974).

Zhang, X., Wang, S., Zhu, F., Xu, Z., Wang, Y., & Huang, J. (2018). Seq3seq fingerprint: towards end-to-end semi-supervised deep drug discovery. In *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics* (pp. 404–413).

Zhang, F., Yuan, N. J., Lian, D., Xie, X., & Ma, W.-Y. (2016). Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 353–362).

Zhang, J., Zhang, H., Xia, C., & Sun, L. (2020). Graph-bert: Only attention is needed for learning graph representations. arXiv preprint arXiv:2001.05140.

Zhang, Y., Zhang, F., Yao, P., & Tang, J. (2018). Name disambiguation in aminer: Clustering, maintenance, and human in the loop. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1002–1011).

Zhang, H., Zhao, S., Liu, R., Wang, W., Hong, Y., & Hu, R. (2022). Automatic traffic anomaly detection on the road network with spatial–temporal graph neural network representation learning. *Wireless Communications and Mobile Computing*, *2022*.

Zhang, X., & Zitnik, M. (2020). Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems*, *33*, 9263–9275.

Zhao, Y., Chen, J., Gao, C., Wang, W., Yang, L., Ren, H., et al. (2022). Target-driven structured transformer planner for vision-language navigation. In *Proceedings of the 30th ACM international conference on multimedia* (pp. 4194–4203).

Zhao, J., Dong, Y., Ding, M., Kharlamov, E., & Tang, J. (2021). Adaptive diffusion in graph neural networks. *Advances in Neural Information Processing Systems*, *34*, 23321–23333.

Zhao, H., Liu, S., Ma, C., Xu, H., Fu, J., Deng, Z.-H., et al. (2023). Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning. *bioRxiv*, 2023-05.

Zhao, Z., Liu, W., Qian, Y., Nie, L., Yin, Y., & Zhang, Y. (2018). Identifying advisor-advisee relationships from co-author networks via a novel deep model. *Information Sciences*, *466*, 258–269.

Zhao, Y., Luo, X., Ju, W., Chen, C., Hua, X.-S., & Zhang, M. (2023). *Dynamic hypergraph structure learning for traffic flow forecasting*. ICDE.

Zhao, P., Luo, A., Liu, Y., Zhuang, F., Xu, J., Li, Z., et al. (2020). Where to go next: A spatio-temporal gated network for next poi recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Zhao, H., Ma, S., Zhang, D., Deng, Z.-H., & Wei, F. (2023). Are more layers beneficial to graph transformers? arXiv preprint arXiv:2303.00579.

Zhao, L., Sawlani, S., Srinivasan, A., & Akoglu, L. (2022). Graph anomaly detection with unsupervised gnns. arXiv preprint arXiv:2210.09535.

Zhao, J., Wang, X., Shi, C., Hu, B., Song, G., & Ye, Y. (2021). Heterogeneous graph structure learning for graph neural networks. *Vol. 35, In Proceedings of the AAAI conference on artificial intelligence* (pp. 4697–4705).

Zhao, J., Wen, Q., Ju, M., Zhang, C., & Ye, Y. (2023). Self-supervised graph structure refinement for graph neural networks. In *Proceedings of the sixteenth ACM international conference on web search and data mining* (pp. 159–167).

Zhao, Y., Yan, H., & Wei, X. (2020). Object hider: Adversarial patch attack against object detectors. arXiv preprint arXiv:2010.14974.

Zhao, T., Zhang, X., & Wang, S. (2021). Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 833–841).

Zheng, Y., Feng, X., Xia, Z., Jiang, X., Demontis, A., Pintor, M., et al. (2021). Why adversarial reprogramming works, when it fails, and how to tell the difference. arXiv preprint arXiv:2108.11673.

Zheng, L., Li, Z., Li, J., Li, Z., & Gao, J. (2019). Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn. In *IJCAI* (pp. 4419–4425).

Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., et al. (2020). Robust graph representation learning via neural sparsification. In *International conference on machine learning* (pp. 11458–11468). PMLR.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., et al. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81.

Zhou, D., Huang, J., & Schölkopf, B. (2006). Learning with hypergraphs: Clustering, classification, and embedding. *Advances in Neural Information Processing Systems*, *19*.

Zhou, F., Li, R., Gao, Q., Trajcevski, G., Zhang, K., & Zhong, T. (2022). Dynamic manifold learning for land deformation forecasting.

Zhou, H., Ren, D., Xia, H., Fan, M., Yang, X., & Huang, H. (2021). Ast-gnn: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction. *Neurocomputing*, *445*, 298–308.

Zhou, Y., Wang, W., Qiao, Z., Xiao, M., & Du, Y. (2020). A survey on the construction methods and applications of sci-tech big data knowledge graph. *Scientia Sinica Informationis*, *50*, 957.

Zhou, Z., Zhang, S., & Huang, Z. (2019). Dynamic self-training framework for graph convolutional networks.

Zhou, Y., Zheng, H., Huang, X., Hao, S., Li, D., & Zhao, J. (2022). Graph neural networks: Taxonomy, advances, and trends. *ACM Transactions on Intelligent Systems and Technology*, *13*, 1–54.

Zhu, Z. (2022). A survey of gnn in bioinformation data.

Zhu, H., & Koniusz, P. (2021). Simple spectral graph convolution. In *International conference on learning representations*.

Zhu, Y., Quan, L., Chen, P.-Y., Kim, M. C., & Che, C. (2023). Predicting coauthorship using bibliographic network embedding. *Journal of the Association for Information Science and Technology*, *74*, 388–401.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159.

Zhu, J., Xia, Y., Liu, C., Wu, L., Xie, S., Wang, T., et al. (2022). Direct molecular conformation generation. arXiv preprint arXiv:2202.01356.

Zhu, J., Xia, Y., Wu, L., Xie, S., Qin, T., Zhou, W., et al. (2022). Unified 2d and 3d pre-training of molecular representations. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 2626–2636).

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., & Wang, L. (2020). Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., & Wang, L. (2021). Graph contrastive learning with adaptive augmentation. In *Proceedings of the web conference 2021* (pp. 2069–2080).

Zhu, Y., Xu, Y., Yu, F., Wu, S., & Wang, L. (2020). Cagnn: Cluster-aware graph neural networks for unsupervised graph representation learning. arXiv preprint arXiv:2009.01674.

Zhu, D., Zhang, Z., Cui, P., & Zhu, W. (2019). Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1399–1407).

Zügner, D., & Günnemann, S. (2020). Certifiable robustness of graph convolutional networks under structure perturbations. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1656–1665).