Deep graph similarity learning: a survey

Guixiang Ma[1] · **Nesreen K. Ahmed[2]** · **Theodore L. Willke[1]** · **Philip S. Yu[3]**

Abstract

In many domains where data are represented as graphs, learning a similarity metric among graphs is considered a key problem, which can further facilitate various learning tasks, such as classification, clustering, and similarity search. Recently, there has been an increasing interest in deep graph similarity learning, where the key idea is to learn a deep learning model that maps input graphs to a target space such that the distance in the target space approximates the structural distance in the input space. Here, we provide a comprehensive review of the existing literature of deep graph similarity learning. We propose a systematic taxonomy for the methods and applications. Finally, we discuss the challenges and future directions for this problem.

Keywords Metric learning · Similarity learning · Graph neural networks · Graph convolutional networks · Higher-order networks · Graph similarity · Structural similarity · Graph matching · Deep graph similarity learning

---

Responsible editor: Hanghang Tong

✉ Guixiang Ma
guixiang.ma@intel.com

Nesreen K. Ahmed
nesreen.k.ahmed@intel.com

Theodore L. Willke
ted.willke@intel.com

Philip S. Yu
psyu@uic.edu

[1] Intel Labs, Intel Corporation, Hillsboro, OR 97124, USA

[2] Intel Labs, Intel Corporation, Santa Clara, CA 95054, USA

[3] Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA

1 Introduction

Learning an adequate similarity measure on a feature space can significantly determine the performance of machine learning methods. Learning such measures automatically from data is the primary aim of similarity learning. Similarity/Metric learning refers to learning a function to measure the distance or similarity between objects, which is a critical step in many machine learning problems, such as classification, clustering, ranking, etc. For example, in k-Nearest Neighbor (kNN) classification (Cover and Hart 1967), a metric is needed for measuring the distance between data points and identifying the nearest neighbors; in many clustering algorithms, similarity measurements between data points are used to determine the clusters. Although there are some general metrics like Euclidean distance that can be used for getting similarity measure between objects represented as vectors, these metrics often fail to capture the specific characteristics of the data being studied, especially for structured data. Therefore, it is essential to find or learn a metric for measuring the similarity of data points involved in the specific task.

Metric learning has been widely studied in many fields on various data types. For instance, in computer vision, metric learning has been explored on images or videos for image classification, object recognition, visual tracking, and other learning tasks (Mensink et al. 2012; Guillaumin et al. 2009; Jiang et al. 2012). In information retrieval, such as in search engines, metric learning has been used to determine the ranking of relevant documents to a given query (Lee et al. 2008; Lim et al. 2013). In this paper, we survey the existing work in similarity learning for graphs, which encode relational structures and are ubiquitous in various domains.

Similarity learning for graphs has been studied for many real applications, such as molecular graph classification in chemoinformatics (Horváth et al. 2004; Fröhlich et al. 2006), protein-protein interaction network analysis for disease prediction (Borgwardt et al. 2007), binary function similarity search in computer security (Li et al. 2019), multi-subject brain network similarity learning for neurological disorder analysis (Ktena et al. 2018), etc. In many of these application scenarios, the number of training samples available is often very limited, making it a difficult problem to directly train a classification or prediction model. With graph similarity learning strategies, these applications benefit from pairwise learning that utilizes every pair of training samples to learn a metric for mapping the input data to the target space, which further facilitates the specific learning task.

In the past few decades, many techniques have emerged for studying the similarity of graphs. Early on, multiple graph similarity metrics were defined, such as the Graph Edit Distance (Bunke and Allermann 1983), Maximum Common Subgraph (Bunke and Shearer 1998; Wallis et al. 2001), and Graph Isomorphism (Dijkman et al. 2009; Berretti et al. 2001), to address the problem of graph similarity search and graph matching. However, the computation of these metrics is an NP-complete problem in general (Zeng et al. 2009). Although some pruning strategies and heuristic methods have been proposed to approximate the values and speed up the computation, it is difficult to analyze the computational complexities of the above heuristic algorithms and the sub-optimal solutions provided by them are also unbounded (Zeng et al. 2009). Therefore, these approaches are feasible only for graphs of relatively small size and

in practical applications where these metrics are of primary interest. Thus it is hard to adapt these methods to new tasks. In addition, for other methods that are relatively more efficient, like the Weisfeiler-Lehman method in Douglas (2011), since it is developed specifically for isomorphism testing without mapping functions, it cannot be applied for general graph similarity learning. More recently, researchers have formulated similarity estimation as a learning problem where the goal is to learn a model that maps a pair of graphs to a similarity score based on the graph representations. For example, graph kernels, such as path-based kernels (Borgwardt and Kriegel 2005) and the subgraph matching kernel (Yan et al. 2005; Yoshida et al. 2019), were proposed for graph similarity learning. Traditional graph embedding techniques, such as geometric embedding, are also leveraged for graph similarity learning (Johansson and Dubhashi 2015).

With the emergence of deep learning techniques, graph neural networks (GNNs) have become a powerful new tool for learning representations on graphs with various structures for various tasks. The main distinction between GNNs and the traditional graph embedding is that GNNs address graph-related tasks in an end-to-end manner, where the representation learning and the target learning task are conducted jointly (Wu et al. 2020), while the graph embedding generally learns graph representations in an isolated stage and the learned representations are then used for the target task. Therefore, the GNN deep models can better leverage the graph features for the specific learning task compared to the graph embedding methods. Moreover, GNNs are easily adapted and extended for various graph related tasks, including deep graph similarity learning tasks in different domains. For instance, in brain connectivity network analysis in neuroscience, community structure among the nodes (i.e., brain regions) within the brain network is an essential factor that should be considered when learning node representations for cross-subject similarity analysis. However, none of the traditional graph embedding methods are able to capture such special structure and jointly leverage the learned node representations for similarity learning on brain networks. In Ma et al. (2019), a higher-order GNN model is developed to encode the community-structure of brain networks during the representation learning and leverage it for the similarity learning task on these brain networks. Some more examples from other domains include the GNN-based graph similarity predictive models introduced for chemical compound queries in computational chemistry (Bai et al. 2019a), and the deep graph matching networks proposed for binary function similarity search and malware detection in computer security (Li et al. 2019; Wang et al. 2019c).

In this survey paper, we provide a systematic review of the existing work in deep graph similarity learning. Based on the different graph representation learning strategies and how they are leveraged for the deep graph similarity learning task, we propose to categorize deep graph similarity learning models into three groups: Graph Embedding based-methods, GNN-based methods, and Deep Graph Kernel-based methods. Additionally, we sub-categorize the models based on their properties. Table 2 shows our proposed taxonomy, with some example models for each category as well as the relevant applications. In this survey, we will illustrate how these different categories of models approach the graph similarity learning problem. We will also discuss the loss functions used for the graph similarity learning task.

Scope and contributions. This paper is focused on surveying the recently emerged deep models for graph similarity learning, where the goal is to use deep strategies on graphs for learning the similarity of given pairs of graphs, instead of computing similarity scores based on predefined measures. We emphasize that this paper does not attempt to survey the extensive literature on graph representation learning, graph neural networks, and graph embedding. Prior work has focused on these topics (see Cai et al. 2018; Goyal and Ferrara 2018; Lee et al. 2019; Wu et al. 2020; Rossi et al. 2020b; Cui et al. 2018; Zhang et al. 2018a for examples). Here instead, we focus on deep graph representation learning methods that explicitly focus on modeling graph similarity. To the best of our knowledge, this is the first survey paper on this problem. We summarize the main contributions of this paper as follows:

- Two comprehensive taxonomies to categorize the literature of the emerging field of deep graph similarity learning, based on the type of models and the type of features adopted by the existing methods, respectively.
- Summary and discussion of the key techniques and building blocks of the models in each category.
- Summary and comparison of the different deep graph similarity learning models across the taxonomy.
- Summary and discussion of the real-world applications that can benefit from deep graph similarity learning in a variety of domains.
- Summary and discussion of the major challenges for deep graph similarity learning, the future directions, and the open problems.

Organization. The rest of the paper is organized as follows. In Sect. 2, we introduce notation, preliminary concepts, and define the graph similarity learning problem. In Sect. 3, we introduce the taxonomy with detailed illustrations of the existing deep models. In Sect. 4, we summarize the datasets and evaluations adopted in the existing works. In Sect. 5, we present the applications of deep graph similarity learning in various domains. In Sect. 6, we discuss the remaining challenges in this area and highlight future directions. Finally, we conclude in Sect. 7.

2 Notation and preliminaries

In this section, we provide the necessary notation and definitions of the fundamental concepts pertaining to the graph similarity problem that will be used throughout this survey. The notation is summarized in Table 1.

Let $G = (V, E, \mathbf{A})$ denote a graph, where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$ is the adjacency matrix of the graph. This is a general notation for graphs that covers different types of graphs, including unweighted/weighted graphs, undirected/directed graphs, and attributed/non-attributed graphs.

We are also assuming a set of graphs as input, $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$, and the goal is measure/model their pairwise similarity. This relates to the classical problem of graph isomorphism and its variants. In graph isomorphism (Miller 1979), two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are isomorphic (i.e., $G \cong H$), if there is a

Table 1 Summary of notation

G	Input graph
V	The set of nodes in a graph G
E	The set of edges in a graph G
a, $\mathbf{a}$, $\mathbf{A}$	Scalar, vector, matrix
$\mathcal{G}$	Graph set $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$
$\mathcal{M}$	Similarity function
s_{ij}	Similarity score between two graphs $G_i, G_j \in \mathcal{G}$
$\mathbb{R}^{m \times m}$	$m-$dimensional Euclidean space
$\mathbf{I}_m$	Identity matrix of dimension m
$\mathbf{A}^T$	Matrix transpose
$\mathbf{L}$	Laplacian matrix
$g_\theta * \mathbf{x}$	Convolution of g_θ and $\mathbf{x}$

mapping function $\pi : V_G \to V_H$, such that $(u, v) \in E_G$ iff $(\pi(u), \pi(v)) \in E_H$. The graph isomorphism is an NP problem, and no efficient algorithms are known for it. Subgraph isomorphism is a generalization of the graph isomorphism problem. In subgraph isomorphism, the goal is to answer for two input graphs G and H, if there is a subgraph of G ($G' \subset G$) such that G' is isomorphic to H (i.e., $G' \cong H$). This is suitable in a setting in which the two graphs have different sizes. The subgraph isomorphism problem has been proven to be NP-complete (unlike the graph isomorphism problem) (Garey and Johnson 1979). The maximum common subgraph problem is another less-restrictive measure of graph similarity, in which the similarity between two graphs is defined based on the size of the largest common subgraph in the two input graphs. However, this problem is also NP-complete (Garey and Johnson 1979).

Definition 1 (**Graph Similarity Learning**) Let $\mathcal{G}$ be an input set of graphs, $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ where $G_i = (V_i, E_i, \mathbf{A}_i)$. Let $\mathcal{M}$ denote a learnable similarity function, such that $\mathcal{M} : (G_i, G_j) \to \mathbb{R}$, for any pair of graphs $G_i, G_j \in \mathcal{G}$. Assume $s_{ij} \in \mathbb{R}$ denote the similarity score computed using $\mathcal{M}$ between pairs G_i and G_j. Then $\mathcal{M}$ is symmetric if and only if $s_{ij} = s_{ji}$ for any pair of graphs $G_i, G_j \in \mathcal{G}$. $\mathcal{M}$ should satisfy the property that: $s_{ii} >= s_{ij}$ for any pair of graphs $G_i, G_j \in \mathcal{G}$. And, s_{ij} is minimum if G_i is the complement of G_j, i.e, $G_i = \bar{G}_j$, for any graph $G_j \in \mathcal{G}$.

Clearly, graph isomorphism and its related variants (e.g., subgraph isomorphism, maximum common subgraphs, etc.) are focused on measuring the topological equivalence of graphs, which gives rise to a binary similarity measure that outputs 1 if two graphs are isomorphic and 0 otherwise. While these methods may sound intuitive, they are actually more restrictive and difficult to compute for large graphs. Here instead, we focus on a relaxed notion of graph similarity that can be measured using machine learning models, where the goal is to learn a model that quantifies the degree of structural similarity and relatedness between two graphs. This is slightly similar to the work done on modeling the structural similarity between nodes in the same

graph (Ahmed et al. 2020; Rossi and Ahmed 2014; Ahmed et al. 2018). We formally state the definition of graph similarity learning (GSL) in Definition 1. Note that in the case of deep graph similarity learning, the similarity function $\mathcal{M}$ is a neural network model that can be trained in an end-to-end fashion.

3 Taxonomy of models

In this section, we describe the taxonomy for the literature of deep graph similarity learning. As shown in Fig. 1, we propose two intuitive taxonomies for categorizing the various deep graph similarity learning methods based on the model architecture and the type of features used in these methods.

First, we start by discussing the categorization based on which model architecture has been used. There are three main categories of deep graph similarity learning methods (see Fig. 1a): (1) graph embedding based methods, which apply graph embedding techniques to obtain node-level or graph-level representations and further use the representations for similarity learning (Tixier et al. 2019; Nikolentzos et al. 2017; Narayanan et al. 2017; Atamna et al. 2019; Wu et al. 2018; Wang et al. 2019a; Xu et al. 2017; Liu et al. 2019b); (2) graph neural network (GNN) based models, which are based on using GNNs for similarity learning, including GNN-CNNs (Bai et al. 2018, 2019a), Siamese GNNs (Ktena et al. 2018; Ma et al. 2019; Liu et al. 2019a; Wang et al. 2019c; Chaudhuri et al. 2019) and GNN-based graph matching networks (Li et al. 2019; Ling et al. 2019; Bai et al. 2019b; Wang et al. 2019b; Jiang et al. 2019; Guo et al. 2018); and (3) deep graph kernels that first map graphs into a new feature space, where kernel functions are defined for similarity learning on graph pairs, including sub-structure based deep kernels (Yanardag and Vishwanathan 2015) and deep neural
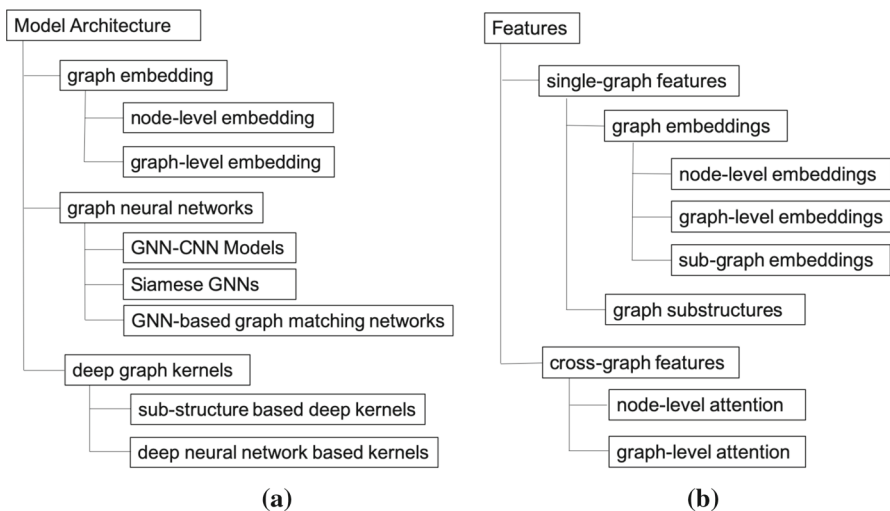

Fig. 1 Proposed taxonomy for categorizing the literature of deep graph similarity learning based on **a** model architecture, **b** type of features

Table 2 A taxonomy of deep graph similarity learning methods

Category		Methods	Weighted graphs	Heterogeneous graphs	Attributed graphs	Feature propagation	Cross-graph interaction	Applications
Graph embedding based GSL	Node-level embedding	Tixier et al. (2019)	×	×	✓	×	×	Social network analysis, Bioinformatics
		Nikolentzos et al. (2017)	×	✓	×	×	×	Chemoinformatics
	Graph-level embedding	Narayanan et al. (2017), Atamna et al. (2019), Wu et al. (2018)	×	✓	×	×	×	Chemoinformatics, Bioinformatics
		Wang et al. (2019a)	×	✓	×	×	×	Social network analysis, Chemoinformatics
		Xu et al. (2017)	×	×	✓	×	×	Binary code similarity
		Liu et al. (2019b)	×	×	✓	×	×	Chemoinformatics
GNN-based GSL	GNN-CNN models	Bai et al. (2018)	×	✓	✓	✓	×	Chemoinformatics
		Bai et al. (2019a)	×	✓	✓	✓	×	Chemoinformatics
	Siamese GNNs	Ktena et al. (2018), Ma et al. (2019), Liu et al. (2019a)	✓	×	✓	✓	×	Brain Network Analysis
		Wang et al. (2019c)	×	✓	✓	✓	×	Malware detection
		Chaudhuri et al. (2019)	✓	×	✓	✓	×	Image retrieval

Table 2 continued

Category	Methods	Weighted graphs	Heterogeneous graphs	Attributed graphs	Feature propagation	Cross-graph interaction	Applications
GNN-based graph matching networks	Li et al. (2019); Ling et al. (2019)	×	×	✓	✓	✓	Binary Code Similarity
	Bai et al. (2019b)	×	✓	✓	✓	✓	Chemoinformatics
	Wang et al. (2019b), Jiang et al. (2019)	×	×	✓	✓	✓	Image Matching
	Guo et al. (2018)	✓	✓	✓	✓	×	3D Action recognition
Deep Graph Kernel based GSL	Sub-structure based deep kernels Yanardag and Vishwanathan (2015)	×	✓	×	×	×	Chemoinformatics Bioinformatics Social network analysis
	Deep neural network based kernels Al-Rfou et al. (2019)	×	✓	✓	×	✓	Chemoinformatics Bioinformatics
	Du et al. (2019)	×	✓	✓	✓	×	Social network analysis Chemoinformatics

network based kernels (Al-Rfou et al. 2019; Du et al. 2019). In the meantime, different methods may use different types of features in the learning process.

Second, we discuss the categorization of methods based on the type of features used in them. Existing GSL approaches can be generally grouped into two categories (see Fig. 1b): (1) methods that uses single-graph features (Ktena et al. 2018; Ma et al. 2019; Liu et al. 2019a; Wang et al. 2019c; Chaudhuri et al. 2019); (2) methods that uses cross-graph features for similarity learning (Li et al. 2019; Ling et al. 2019; Bai et al. 2019b; Al-Rfou et al. 2019; Wang et al. 2019b; Bai et al. 2019b). The main difference between these two categories of methods is that for methods using single-graph features, the representation of each graph is learned individually, while those methods that use cross-graph features allow graphs to learn and propagate features from each other and the cross-graph interaction is leveraged for pairs of graphs. The single-graph features mainly includes graph embeddings at different granularity (i.e.,node-level, graph-level, and subgraph-level), while the cross-graph features includes the cross-graph node-level features and cross-graph graph-level features, which are usually obtained by node-level attention and graph-level attention across the two graphs in each pair.

Next, we detail the description of the methods based on the taxonomy in Fig. 1a, b. We summarize the general characteristics and applications of all the methods in Table 2, including the type of graphs they are developed for, the type of features, and the domains/applications where they could be applied. We describe these methods in the following order:

1. Graph embedding based GSL
2. Graph Neural Network based GSL
3. Deep graph kernel based GSL

3.1 Graph embedding based graph similarity learning

Graph embedding has received considerable attention in the past decade (Cui et al. 2018; Zhang et al. 2018a), and a variety of deep graph embedding models have been proposed in recent years (Huang et al. 2019; Narayanan et al. 2017; Gao and Ji 2019b), for example the popular *DeepWalk* model proposed in (Perozzi et al. 2014) and the *node2vec* model from (Grover and Leskovec 2016). Similarity learning methods based on graph embedding seek to utilize node-level or graph-level representations learned by these graph embedding techniques for defining similarity functions or predicting similarity scores (Tsitsulin et al. 2018; Tixier et al. 2019; Narayanan et al. 2017). Given a collection of graphs, these works first aim to convert each graph G into a $d-$dimensional space ($d \ll \|V\|$), where the graph is represented as either a set of $d-$dimensional vectors with each vector representing the embedding of one node (i.e.,node-level embedding) or a $d-$dimensional vector for the whole graph as the graph-level embedding (Cai et al. 2018). The graph embeddings are usually learned in an unsupervised manner in a separate stage prior to the similarity learning stage, where the graph embeddings obtained are used for estimating or predicting the similarity score between each pair of graphs.

3.1.1 Node-level embedding based methods

Node-level embedding based methods compare graphs using the node-level representations learned from the graphs. The similarity scores obtained by these methods mainly capture the similarity between the corresponding nodes in two graphs. Therefore they focus on the local node-level information on graphs during the learning process.

node2vec-PCA. In Tixier et al. (2019), the node2vec approach (Grover and Leskovec 2016) is employed for obtaining the node-level embeddings of graphs. To make the embeddings of all the graphs in the given collection comparable, they apply the principal component analysis (PCA) on the embeddings to retain the first $d \ll D$ principal components (where D is the dimensionality of the original node embedding space). Afterwards, the embedding matrix of each graph is split into $d/2$ 2D slices. Suppose there are n nodes in each graph G and the embedding matrix for graph G is $F \in \mathbb{R}^{n \times d}$, then $d/2$ 2D slices each with $\mathbb{R}^{n \times 2}$ will be obtained, which are viewed as $d/2$ *channels*. Then each 2D slice from the embedding space is turned into regular grids by discretizing them into a fixed number of equally-sized bins, where the value associate with each bin is the count of the number of nodes falling into that bin. These bins can be viewed as *pixels*. Then, the graph is represented as a stack of 2D histograms of its node embeddings. The graphs are then compared in the grid space and input into a 2D CNN as multi-channel image-like structures for a graph classification task.

Bag-of-vectors. In Nikolentzos et al. (2017), the nodes of the graphs are first embedded in the Euclidean space using the eigenvectors of the adjacency matrices of the graphs, and each graph is then represented as a bag-of-vectors. The similarity between two graphs is then measured by computing a matching based on the Earth Mover's Distance (Rubner et al. 2000) between the two sets of embeddings.

Although node embedding based graph similarity learning methods have been extensively developed, a common problem with these methods is that, since the comparison is based on node-level representations, the global structure of the graphs tends to be ignored, which actually is very important for comparing two graphs in terms of their structural patterns.

3.1.2 Graph-level embedding based methods

The graph-level embedding based methods aim to learn a vector representation for each graph and then learn the similarity score between graphs based on their vector representations.

(1) graph2vec. In Narayanan et al. (2017), a graph2vec was proposed to learn distributed representations of graphs, similar to Doc2vec (Le and Mikolov 2014) in natural language processing. In graph2vec each graph is viewed as a document and the rooted subgraphs around every node in the graph are viewed as words that compose the document. There are two main components in this method: first, a procedure to extract rooted subgraphs around every node in a given graph following the Weisfeiler-Lehman relabeling process and second, the procedure to learn embeddings of the given graphs

by skip-gram with negative sampling. The Weisfeiler-Lehman relabeling algorithm takes the root node of the given graph and degree of the intended subgraph d as inputs, and returns the intended subgraph. In the negative sampling phase, given a graph and a set of rooted subgraphs in its context, a set of randomly chosen subgraphs are selected as negative samples and only the embeddings of the negative samples are updated in the training. After the graph embedding is obtained for each graph, the similarity or distance between graphs are computed in the embedding space for downstream prediction tasks (e.g., graph classification, clustering, etc.).

(2) Neural networks with Structure2vec. In Xu et al. (2017), a deep graph embedding approach is proposed for cross-platform binary code similarity detection. A Siamese architecture is applied to enable the pair-wise similarity learning, and the graph embedding network based on Structure2vec (Dai et al. 2016) is used for learning graph representations in the twin networks, which share weights with each other. The Structure2vec is a neural network approach inspired by graphical model inference algorithms where node-specific features are aggregated recursively according to graph topology. After a few steps of recursion, the network will produce a new feature representation for each node which considers both graph characteristics and long-range interaction between node features. Given is a set of K pairs of graphs $< G_i, G_i' >$, with ground truth pair label $y_i \in \{+1, -1\}$, where $y_i = +1$ indicates that G_i and G_i' are similar, and $y_i = -1$ indicates they are dissimilar. With the Structure2vec embedding output for G_i and G_i', represented as $\mathbf{f}_i$ and $\mathbf{f}_i'$ respectively, they define the Siamese network output for each pair as

$$Sim(G_i, G_i') = \cos(\mathbf{f}_i, \mathbf{f}_i') = \frac{\langle \mathbf{f}_i, \mathbf{f}_i' \rangle}{\|\mathbf{f}_i\| \cdot \|\mathbf{f}_i'\|} \tag{1}$$

and the following loss function is used for training the model.

$$L = \sum_{i=1}^{K} (Sim(G_i, G_i') - y_i)^2 \tag{2}$$

(3) Simple permutation-invariant GCN. In Atamna et al. (2019), a graph representation learning method based on a simple permutation-invariant graph convolutional network is proposed for the graph similarity and graph classification problem. A graph convolution module is used to encode local graph structure and node features, after which a sum-pooling layer is used to transform the substructure feature matrix computed by the graph convolutions into a single feature vector representation of the input graphs. The vector representation is then used as features for each graph, based on which the graph similarity or graph classification task can be performed.

(4) SEED: sampling, encoding, and embedding distributions. In Wang et al. (2019a), an inductive and unsupervised graph representation learning approach called SEED is proposed for graph similarity learning. The proposed framework consists of three components: sampling, encoding, and embedding distribution. In the sampling stage, a number of subgraphs called WEAVE are sampled based on the random

walk with earliest visit time. Then in the encoding stage, an autoencoder (Hinton and Salakhutdinov 2006) is used to encode the subgraphs into dense low-dimensional vectors. Given a set of k sampled WEAVEs $\{X_1, X_2, X_3, \ldots, X_k\}$, for each subgraph X_i the autoencoder works as follows.

$$\mathbf{z}_i = f(X_i; \theta_e), \quad \hat{X}_i = g(\mathbf{z}_i; \theta_d), \tag{3}$$

where $\mathbf{z}_i$ is the dense low-dimensional representation for the input WEAVE subgraph X_i, $f(\cdot)$ is the encoding function implemented with an Multi-layer Perceptron (MLP) with parameters θ_e, and $g(\cdot)$ is the decoding function implemented by another MLP with parameters θ_d. A reconstruction loss is used to train the autoencoder:

$$L = ||X - \hat{X}||_2^2 \tag{4}$$

After the autoencoder is well trained, the final subgraph embedding vectors $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \ldots,$ and $\mathbf{z}_k$ can be obtained for each graph. Finally, in the embedding distribution stage, the distance between the subgraph distributions of two input graphs G and H is evaluated using the maximum mean discrepancy (MMD) (Gretton et al. 2012) on the embeddings. Assume the k subgraphs sampled from G are encoded into embeddings $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k$, and the k subgraphs of H are encoded into embeddings $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k$, the MMD distance between G and H is:

$$\widehat{\text{MMD}}(G, H) = ||\hat{\mu}_G - \hat{\mu}_H||_2^2 \tag{5}$$

where $\hat{\mu}_G$ and $\hat{\mu}_H$ are empirical kernel embeddings of the two distributions, which are defined as:

$$\hat{\mu}_G = \frac{1}{k} \sum_{i=1}^{k} \phi(\mathbf{z}_i), \quad \hat{\mu}_H = \frac{1}{k} \sum_{i=1}^{k} \phi(\mathbf{h}_i) \tag{6}$$

where $\phi(\cdot)$ is the feature mapping function used for the kernel function for graph similarity evaluation. An identity kernel is applied in this work.

(5) DGCNN: disordered graph CNN. In Wu et al. (2018), another graph-level representation learning approach called DGCNN is introduced based on graph CNN and mixed Gaussian model, where a set of key nodes are selected from each graph. Specifically, to ensure the number of neighborhoods of the nodes in each graph is consistent, the same number of key nodes are sampled for each graph in a key node selection stage. Then a convolution operation is performed over the kernel parameter matrix and the nodes in the neighborhood of the selected key nodes, after which the graph CNN takes the output of the convolutional layer as the input data of the overall connection layer. Finally, the output of the dense hidden layer is used as the feature vector for each graph in the graph similarity retrieval task.

(6) N-Gram graph embedding. In Liu et al. (2019b), an unsupervised graph representation based method called N-gram is proposed for similarity learning on molecule

graphs. It first views each node in the graph as one token and applies an analog of the CBOW (continuous bag of words) (Mikolov et al. 2013) strategy and trains a neural network to learn the node embeddings for each graph. Then it enumerates the walks of length n in each graph, where each walk is called an n-gram, and obtains the embedding for each n-gram by assembling the embeddings of the nodes in the n-gram using element-wise product. The embedding for the n-gram walk set is defined as the sum of the embeddings for all n-grams. The final n-gram graph-level representation up to lenght T is then constructed by concatenating the embeddings of all the n-gram sets for $n \in \{1, 2, \ldots, T\}$ in the graph. Finally, the graph-level embeddings are used for the similarity prediction or graph classification task for molecule analysis.

By summarizing the embedding based methods, we find the main advantage of these methods is their speed and scalability, due to the fact that the graph representations learned through these factorized models are developed on each single graph where there is no feature interactions across graphs. This property makes these methods a great option for graph similarity learning applications such as graph retrieval, where similarity search becomes a nearest neighbor search in a database of the precomputed graph representations by these factorized methods. Moreover, these embedding based methods provide a variety of perspectives and strategies for learning representations from graphs and demonstrate that these representations can be used for graph similarity learning. However, there are also shortcomings in these solutions, a common one being that the embeddings are learned independently on the individual graphs in a separate stage from the similarity learning, therefore the graph-graph proximity is not considered or utilized in the graph representation learning process, and the representations learned by these models may not be suitable for graph-graph similarity prediction compared to the methods that integrate the similarity learning with the graph representation learning in an end-to-end framework.

3.2 GNN-based graph similarity learning

The similarity learning methods based on Graph Neural Networks (GNNs) seek to learn graph representations by GNNs while doing the similarity learning task in an end-to-end fashion. Figure 2 illustrates a general workflow of GNN-based graph similarity learning models. Given pairs of input graphs $< G_i, G_j, y_{ij} >$, where y_{ij} denotes the ground-truth similarity label or score of $< G_i, G_j >$, the GNN-based GSL methods first employ multi-layer GNNs with weights W to learn the representations for G_i and G_j in the encoding space, where the learning on each graph in a pair could influence each other by some mechanisms such as weight sharing and cross-graph interactions between the GNNs for the two graphs. A matrix or vector representation will be output for each graph by the GNN layers, after which a dot product layer or fully connected layers can be added to produce or predict the similarity scores between two graphs. Finally, the similarity estimates for all pairs of graphs and their ground-truth labels are used in a loss function for training the model M with parameters W.

Before introducing the methods in this category, we provide the necessary background on GNNs.

GNN preliminaries. Graph neural networks (GNNs) were first formulated in Gori et al. (2005), which proposed to use a propagation process to learn node representations for graphs. It has then been further extended by Scarselli et al. (2008) and Gallicchio and Micheli (2010). Later, graph convolutional networks were proposed which compute node updates by aggregating information in local neighborhoods (Bruna et al. 2013; Defferrard et al. 2016; Kipf and Welling 2016), and they have become the most popular graph neural networks, which are widely used and extended for graph representation learning in various domains (Zhou et al. 2018; Zhang et al. 2018b; Gao et al. 2018; Gao and Ji 2019a, b).

With the development of graph neural networks, researchers began to build graph similarity learning models based on GNNs. In this section, we will first introduce the workflow of GCNs with the spectral GCN (Shuman et al. 2013) as an example, and then describe the GNN-based graph similarity learning methods covering three main categories.

Given a graph $G = (V, E, \mathbf{A})$, where V is the set of vertices, $E \subset V \times V$ is the set of edges, and $\mathbf{A} \in \mathbb{R}^{m \times m}$ is the adjacency matrix, the diagonal degree matrix $\mathbf{D}$ will have elements $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. The graph Laplacian matrix is $\mathbf{L} = \mathbf{D} - \mathbf{A}$, which can be normalized as $\mathbf{L} = \mathbf{I}_m - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}_m$ is the identity matrix. Assume the orthonormal eigenvectors of $\mathbf{L}$ are represented as $\{u_l\}_{l=0}^{m-1} \in \mathbb{R}^{m \times m}$, and their associated eigenvalues are $\{\lambda_l\}_{l=0}^{m-1}$, the Laplacian is diagonalized by the Fourier basis $[u_0, \ldots, u_{m-1}](= \mathbf{U}) \in \mathbb{R}^{m \times m}$ and $\mathbf{L} = \mathbf{U3U^T}$ where $3 = diag([\lambda_0, \ldots, \lambda_{m-1}]) \in \mathbb{R}^{m \times m}$. The graph Fourier transform of a signal $x \in \mathbb{R}^m$ can then be defined as $\hat{x} = \mathbf{U^T} x \in \mathbb{R}^m$ (Shuman et al. 2013). Suppose a signal vector $\mathbf{x} : V \to \mathbb{R}$ is defined on the nodes of graph G, where $\mathbf{x}_i$ is the value of $\mathbf{x}$ at the ith node. Then the signal $\mathbf{x}$ can be filtered by g_θ as

$$y = g_\theta * \mathbf{x} = g_\theta(\mathbf{L})\mathbf{x} = g_\theta(\mathbf{U3U^T})\mathbf{x} = \mathbf{U}g_\theta(\Lambda)\mathbf{U^T}\mathbf{x} \qquad (7)$$

where the filter $g_\theta(\Lambda)$ can be defined as $g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$, and the parameter $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients (Defferrard et al. 2016). GCNs can be constructed by stacking multiple convolutional layers in the form of Eq. (7), with a non-linearity activation (ReLU) following each layer.

Based on how graph-graph similarity/proximity is leveraged in the learning, we summarize the existing GNN-based graph similarity learning work into three main categories: (1) GNN-CNN mixed models for graph similarity prediction, (2) Siamese GNNs for graph similarity prediction, and (3) GNN-based graph matching networks.

3.2.1 GNN-CNN models for graph similarity prediction

The works that use GNN-CNN mixed networks for graph similarity prediction mainly employ GNNs to learn graph representations and leverage the learned representations into CNNs for predicting similarity scores, which is approached as a classification or
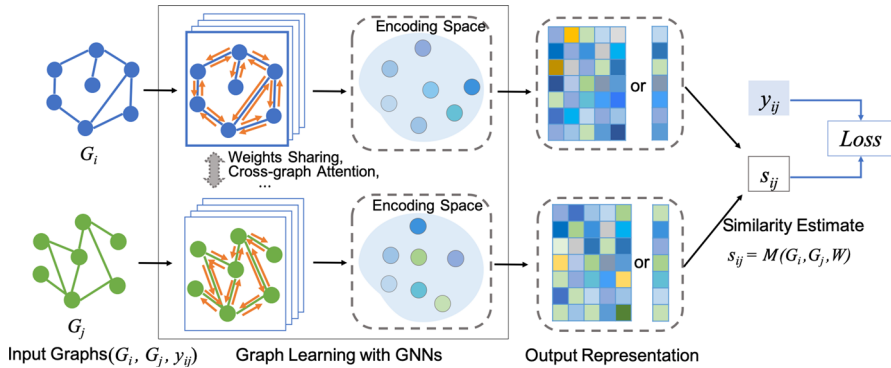
Fig. 2 Illustration of GNN-based graph similarity learning

regression problem. Fully connected layers are often added for the similarity score prediction in an end-to-end learning framework.

(1) GSimCNN. In Bai et al. (2018), a method called GSimCNN is proposed for pairwise graph similarity prediction, which consists of three stages. In Stage 1, node representations are first generated by multi-layer GCNs, where each layer is defined as

$$conv(\mathbf{x}_i) = ReLU \left(\sum_{j \in N(i)} \frac{1}{\sqrt{d_i d_j}} \mathbf{x}_j \mathbf{W}^{(l)} + \mathbf{b}^{(l)} \right) \tag{8}$$

where $N(i)$ is the set of first-order neighbors of node i plus node i itself, d_i is the degree of node i plus 1, $\mathbf{W}^{(l)}$ is the weight matrix for the $l-$th GCN layer, $\mathbf{b}^{(l)}$ is the bias, and $ReLU(x) = max(0, x)$ is the activation function. In Stage 2, the inner products between all possible pairs of node embeddings between two graphs from different GCN layers are calculated, which results in multiple similarity matrices. Finally, the similarity matrices from different layers are processed by multiple independent CNNs, where the output of the CNNs are concatenated and fed into fully connected layers for predicting the final similarity score s_{ij} for each pair of graphs G_i and G_j.

(2) SimGNN. In Bai et al. (2019a), a SimGNN model is introduced based on the GSimCNN from (Bai et al. 2018). In addition to pairwise node comparison with node-level embeddings from the GCN output, neural tensor networks (NTN) (Socher et al. 2013) are utilized to model the relation between the graph-level embeddings of two input graphs, whereas the graph embedding for each graph is generated via a weighted sum of node embeddings, and a global context-aware attention is applied on each node, such that nodes similar to the global context receive higher attention weights. Finally, both the comparison between node-level embeddings and graph-level embeddings are considered for the similarity score prediction in the CNN fully connected layers.
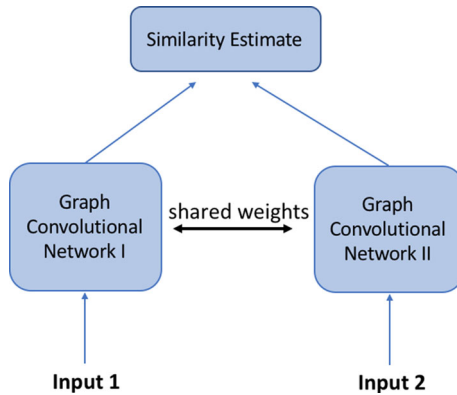
Fig. 3 Siamese architecture with graph convolutional networks

3.2.2 Siamese GNN models for graph similarity learning

This category of works uses the Siamese network architecture with GNNs as twin networks to simultaneously learn representations from two graphs, and then obtain a similarity estimate based on the output representations of the GNNs. Figure 3 shows an example of Siamese architecture with GCNs in the twin networks, where the weights of the networks are shared with each other. The similarity estimate is typically leveraged in a loss function for training the network.

(1) Siamese GCN. The work in Ktena et al. (2018) proposes to learn a graph similarity metric using the Siamese graph convolutional neural network (S-GCN) in a supervised setting. The S-GCN takes a pair of graphs as inputs and employs spectral GCN to get graph embedding for each input graph, after which a dot product layer followed by a fully connected layer is used to produce the similarity estimate between the two graphs in the spectral domain.

(2) Higher-order Siamese GCN. Higher-order Siamese GCN (HS-GCN) is proposed in Ma et al. (2019), which incorporates higher-order node-level proximity into graph convolutional networks so as to perform higher-order convolutions on each of the input graphs for the graph similarity learning task. A Siamese framework is employed with the proposed higher-order GCN in each of the twin networks. Specifically, random walk is used for capturing higher-order proximity from graphs and refining the graph representations used in graph convolutions. Both this work and the S-GCN (Ktena et al. 2018) introduced above use the Hinge loss for training the Siamese similarity learning models:

$$L_{Hinge} = \frac{1}{K} \sum_{i=1}^{N} \sum_{j=i+1}^{N} max(0, 1 - y_{ij}s_{ij}), \tag{9}$$

where N is the total number of graphs in the training set, $K = N(N-1)/2$ is the total number of pairs from the training set, y_{ij} is the ground-truth label for the pair

of graphs G_i and G_j where $y_{ij} = 1$ for similar pairs and $y_{ij} = -1$ for dissimilar pairs, and s_{ij} is the similarity score estimated by the model. More general forms of higher-order information [e.g., motifs (Ahmed et al. 2015, 2017b)] have been used for learning graph representations (Rossi et al. 2018, 2020a) and would likely benefit the learning.

(3) Community-preserving Siamese GCN. In Liu et al. (2019a), another Siamese GCN based model called SCP-GCN is proposed for the similarity learning in functional and structural joint analysis of brain networks, where the graph structure used in the GCN is defined from the structural connectivity network while the node features come from the functional brain network. The contrastive loss (Eq. 10) along with a newly proposed community-preserving loss (Eq. 11) is used for training the model.

$$L_{Contrastive} = \frac{y_{ij}}{2}\|\mathbf{g}_i - \mathbf{g}_j\|_2^2 + (1 - y_{ij})\frac{1}{2}\{max(0, m - \|\mathbf{g}_i - \mathbf{g}_j\|_2)\}^2 \quad (10)$$

where $\mathbf{g}_i$ and $\mathbf{g}_j$ are the graph embeddings of graph G_i and graph G_j computed from the GCN, m is a margin value which is greater than 0. $y_{ij} = 1$ if G_i and G_j are from the same class and $y_{ij} = 0$ if they are from different classes. By minimizing the contrastive loss, the Euclidean distance between two graph embedding vectors will be minimized when the two graphs are from the same class, and maximized when they belong to different classes. The community-preserving loss is defined as follows.

$$L_{CP} = \alpha(\sum_c \frac{1}{|S_c|}\sum_{i \in S_c}\|\mathbf{z}_i - \hat{\mathbf{z}}_c\|_2^2) - \beta\sum_{c,c'}\|\hat{\mathbf{z}}_c - \hat{\mathbf{z}}_{c'}\|_2^2 \quad (11)$$

where S_c contains the indexes of nodes belonging to community c, $\hat{\mathbf{z}}_c = \frac{1}{|S_c|}\sum_{i \in S_c}\mathbf{z}_i$ is the community center embedding for each community c, where $\mathbf{z}_i$ is the embedding of the ith node, i.e., the ith row in the node embedding $\mathbf{Z}$ of the GCN output, and α and β are the weights balancing the intra/inter-community loss.

(4) Hierarchical Siamese GNN. In Wang et al. (2019c), a Siamese network with two hierarchical GNN models is introduced for the similarity learning of heterogeneous graphs for unknown malware detection. Specifically, they consider the path-relevant sets of neighbors according to meta-paths and generate node embeddings by selectively aggregating the entities in each path-relevant neighbor set. The loss function in Eq. (2) is used for training the model.

(5) Siamese GCN for image retrieval. In Chaudhuri et al. (2019), Siamese GCNs are used for content based remote sensing image retrieval, where each image is converted to a region adjacency graph in which each node represents a region segmented from the image. The goal is to learn an embedding space that pulls semantically coherent images closer while pushing dissimilar samples far apart. Contrastive loss is used in the model training.

Since the twin GNNs in the Siamese network share the same weights, an advantage of the Siamese GNN models is that the two input graphs are guaranteed to be processed in the same manner by the networks. As such, similar input graphs would be embedded
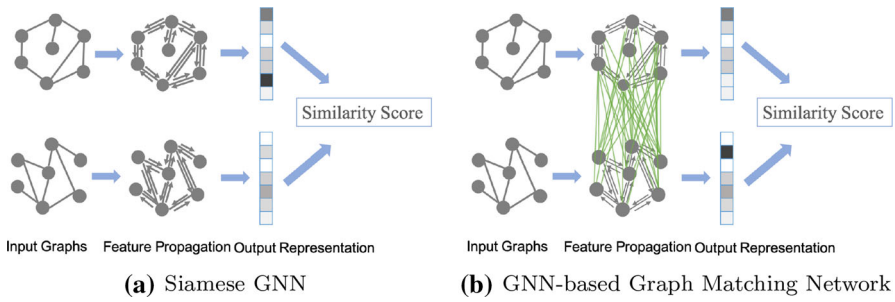
(a) Siamese GNN   **(b)** GNN-based Graph Matching Network

Fig. 4 Comparison of the learning process of Siamese GNN and GNN-based graph matching network

similarly in the latent space. Therefore, the Siamese GNNs are good for differentiating the two input graphs in the latent space or measuring the similarity between them.

In addition to choosing the appropriate GNN models in the twin networks, one needs to choose a proper loss function. Another widely used loss function for Siamese network is the triplet loss (Schroff et al. 2015). For a triplet (G_i, G_p, G_n), G_p is from the same class as G_i, while G_n is from a different class from G_i. The triplet loss is defined as follows.

$$L_{Triplet} = \frac{1}{K} \sum_K max(d_{ip} - d_{in} + m, 0) \tag{12}$$

where K is the number of triplets used in the training, d_{ip} represents the distance between G_i and G_p, d_{in} represents the distance between G_i and G_n, and m is a margin value which is greater than 0. By minimizing the triplet loss, the distance between graphs from same class (i.e., d_{ip}) will be pushed to 0, and the distance between graphs from different classes (i.e., d_{in} will be pushed to be greater than $d_{ip} + m$.

It is important to consider which loss function would be suitable for the targeted problem when applying these Siamese GNN models for the graph similarity learning task in practice.

3.2.3 GNN-based graph matching networks

The work in this category adapts Siamese GNNs by incorporating matching mechanisms during the learning with GNNs, and cross-graph interactions are considered in the graph representation learning process. Figure 4 shows this difference between the Siamese GNNs and the GNN-based graph matching networks.

(1) GMN: graph matching network. In Li et al. (2019), a GNN based architecture called Graph Matching Network (GMN) is proposed, where the node update module in each propagation layer takes into account both the aggregated messages on the edges for each graph and a cross-graph matching vector which measures how well a node in one graph can be matched to the nodes in the other graph. Given a pair of graphs as input, the GMN jointly learns graph representations for the pair through the cross-graph attention-based matching mechanism, which propagates node representations

by using both the neighborhood information within the same graph and cross-graph node information. A similarity score between the two input graphs is computed in the latent vector space.

(2) NeuralMCS: neural maximum common subgraph GMN. Based on the graph matching network in Li et al. (2019) and Bai et al. (2019b) proposes a neural maximum common subgraph (MCS) detection approach for learning graph similarity. The graph matching network is adapted to learn node representations for two input graphs G_1 and G_2, after which a likelihood of matching each node in G_1 to each node in G_2 is computed by a normalized dot product between the node embeddings. The likelihood indicates which node pair is most likely to be in the MCS, and the likelihood for all pairs of nodes constitutes the matching matrix $\mathbf{Y}$ for G_1 and G_2. Then a guided subgraph extraction process is applied, which starts by finding the most likely pair and iteratively expands the extracted subgraphs by selecting one more pair at a time until adding more pairs would lead to non-isomorphic subgraphs. To check the subgraph isomorphism, subgraph-level embeddings are computed by aggregating the node embeddings of the neighboring nodes that are included in the MCS, and Euclidean distance between the subgraph embeddings are computed. Finally, a similarity/match score is obtained based on the subgraphs extracted from G_1 and G_2.

(3) Hierarchical graph matching network. In Ling et al. (2019), a hierarchical graph matching network is proposed for graph similarity learning, which consists of a Siamese GNN for learning global-level interactions between two graphs and a multi-perspective node-graph matching network for learning the cross-level node-graph interactions between parts of one graph and one whole graph. Given two graphs G_1 and G_2 as inputs, a three-layer GCN is utilized to generate embeddings for them, and aggregation layers are added to generate the graph embedding vector for each graph. In particular, cross-graph attention coefficients are calculated between each node in G_1 and all the nodes in G_2, and between each node in G_2 and all the nodes in G_1. Then the attentive graph-level embeddings are generated using the weighted average of node embeddings of the other graph, and a multi-perspective matching function is defined to compare the node embeddings of one graph with the attentive graph-level embeddings of the other graph. Finally, the BiLSTM model (Schuster and Paliwal 1997) is used to aggregate the cross-level interaction feature matrix from the node-graph matching layer, followed by the final prediction layers for the similarity score learning.

(4) NCMN: neural graph matching network. In Guo et al. (2018), a Neural Graph Matching Network (NGMN) is proposed for few-shot 3D action recognition, where 3D data are represented as interaction graphs. A GCN is applied for updating node features in the graphs and an MLP is employed for updating the edge strength. A graph matching metric is then defined based on both node matching features and edge matching features. In the proposed NGMN, edge generation and graph matching metric are learned jointly for the few-shot learning task.

Recently, deep graph matching networks were introduced for the graph matching problem for image matching (Fey et al. 2019; Zanfir and Sminchisescu 2018; Jiang et al. 2019; Wang et al. 2019b). Graph matching aims to find node correspondence

between graphs, such that the corresponding node and edge's affinity is maximized. Although the problem of graph matching is different from the graph similarity learning problem we focus on in this survey and is beyond the scope of this survey, some work on deep graph matching networks involves graph similarity learning and thus we review some of this work below to provide some insights into how deep similarity learning may be leveraged for graph matching applications, such as image matching.

(5) GMNs for image matching. In Jiang et al. (2019), a Graph Learning-Matching Network is proposed for image matching. A CNN is first utilized to extract feature descriptors of all feature points for the input images, and graphs are then constructed based on the features. Then the GCNs are used for learning node embeddings from the graphs, in which both intra-graph convolutions and cross-graph convolutions are conducted. The final matching prediction is formulated as node-to-node affinity metric learning in the embedding space, and the constraint regularized loss along with cross-entropy loss is used for the metric learning and the matching prediction. In Wang et al. (2019b), another GNN-based graph matching network is proposed for the image matching problem, which consists of a CNN image feature extractor, a GNN-based graph embedding component, an affinity metric function and a permutation prediction component, as an end-to-end learnable framework. Specifically, GCNs are used to learn node-wise embeddings for intra-graph affinity, where a cross-graph aggregation step is introduced to aggregate features of nodes in the other graph for incorporating cross-graph affinity into the node embeddings. The node embeddings are then used for building an affinity matrix that contains the similarity scores at the node level between two graphs, and the affinity matrix is further used for the matching prediction. The cross-entropy loss is used to train the model end-to-end.

3.3 Deep graph kernels

Graph kernels have become a standard tool for capturing the similarity between graphs for tasks such as graph classification (Vishwanathan et al. 2010). Given a collection of graphs, possibly with node or edge attributes, the work in graph kernel aim to learn a kernel function that can capture the similarity between any two graphs. Traditional graph kernels, such as random walk kernels, subtree kernels, and shortest-path kernels have been widely used in the graph classification task (Nikolentzos et al. 2019). Recently, deep graph kernel models have also emerged, which build kernels based on the graph representations learned via deep neural networks.

3.3.1 Deep graph kernels

In Yanardag and Vishwanathan (2015), a Deep Graph Kernel approach is proposed. For a given set of graphs, each graph is decomposed into its sub-structures. Then the sub-structures are viewed as words and neural language models in the form of CBOW (continuous bag-of-words) and Skip-gram are used to learn latent representations of sub-structures from the graphs, where corpora are generated for the Shortest-path graph and Weisfeiler-Lehman kernels in order to measure the co-occurrence relationship
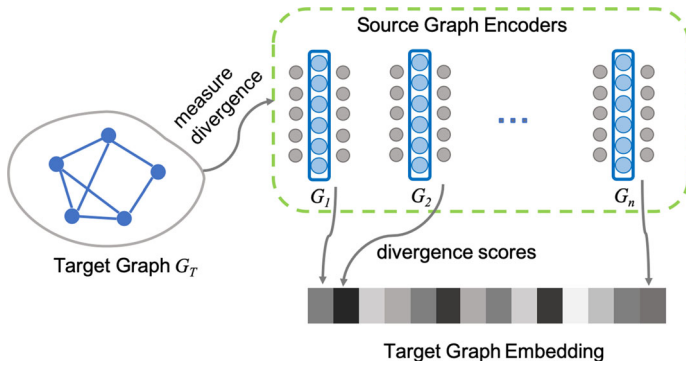
Fig. 5 The graph representation learning in the deep divergence graph kernels (Al-Rfou et al. 2019)

between substructures. Finally, the kernel between two graphs is defined based on the similarity of the sub-structure space.

3.3.2 Deep divergence graph kernels

In Al-Rfou et al. (2019), a model called Deep Divergence Graph Kernels (DDGK) is introduced to learn kernel functions for graph pairs. Given two graphs G_1 and G_2, they aim to learn an embedding based kernel function $k()$ as a similarity metric for graph pairs, defined as:

$$k(G_1, G_2) = \|\Psi(G_1) - \Psi(G_2)\|^2 \tag{13}$$

where $\Psi(G_i)$ is a representation learned for G_i. This work proposes to learn graph representation by measuring the divergence of the target graph across a population of source graph encoders. Given a source graph collection $\{G_1, G_2, \ldots, G_n\}$, a graph encoder is first trained to learn the structure of each graph in the source collection. Then, for a target graph G_T, the divergence of G_T from each source graph is measured, after which the divergence scores are used to compose the vector representation of the target graph G_T. Figure 5 illustrates the above graph representation learning process. Specifically, the divergence score between a target graph $G_T = (V_T, E_T)$ and a source graph $G_S = (V_S, E_S)$ is computed as follows:

$$\mathcal{D}'(G_T \| G_S) = \sum_{v_i \in V_T} \sum_{\substack{j \\ e_{ij} \in E_T}} -log\mathrm{Pr}(v_j | v_i, H_S) \tag{14}$$

where H_S is the encoder trained on graph S.

3.3.3 Graph neural tangent kernel

In Du et al. (2019), a Graph Neural Tangent Kernel (GNTK) is proposed for fusing GNNs with the neural tangent kernel, which is originally formulated for fully-

connected neural networks in Jacot et al. (2018) and later introduced to CNNs in Arora et al. (2019). Given a pair of graphs $< G, G' >$, they first apply GNNs on the graphs. Let $f(\theta, G) \in \mathbb{R}$ be the output of the GNN under parameters $\theta \in \mathbb{R}^m$ on input Graph G, where m is the dimension of the parameters. To get the corresponding GNTK value, they calculate the expected value of

$$\left\langle \frac{\partial f(\theta, G)}{\partial \theta}, \frac{\partial f(\theta, G')}{\partial \theta} \right\rangle \tag{15}$$

in the limit that $m \to \infty$ and θ are all Gaussian random variables.

Meanwhile, there are also some deep graph kernels proposed for the node representation learning on graphs for node classification and node similarity learning. For instance, in Tian et al. (2019), a learnable kernel-based framework is proposed for node classification, where the kernel function is decoupled into a feature mapping function and a base kernel. An encoder-decoder function is introduced to project each node into the embedding space and reconstructs pairwise similarity measurements from the node embeddings. Since we focus on the similarity learning between graphs in this survey, we will not discuss this work further.

4 Datasets and evaluation

In this section, we summarize the characteristics of the datasets that are frequently used in deep graph similarity learning methods and the experimental evaluation adopted by these methods.

4.1 Datasets

Graph data from various domains have been used to evaluate graph similarity learning methods (Rossi and Ahmed 2015), for example, protein-protein graphs from bioinformatics, chemical compound graphs from chemoinformatics, and brain networks from neuroscience, etc. We summarize the benchmark datasets that are frequently used in deep graph similarity learning methods in Table 3.

In addition to these datasets, synthetic graph datasets or other domain-specific datasets are also widely used in some graph similarity learning works. For example, in Li et al. (2019) and Fey et al. (2019), control flow graphs of binary functions are generated and used to evaluate graph matching networks for binary code similarity search. In Wang et al. (2019c), attacks are conducted on testing machines to generate malware data, which are then merged with normal data to evaluate the Siamese GNN model for malware detection. In Jiang et al. (2019), images are collected from multiple categories and keypoints are annotated in the images to evaluate the proposed model for graph matching.

Table 3 Summary of benchmark datasets that are frequently used in deep graph similarity learning

Graph type	Datasets	Source	Number of graphs	Number of classes	Avg. number of nodes	Avg. number of edges	References
Social Networks	COLLAB	Yanardag and Vishwanathan (2015)	5000	3	74.49	2457.78	Yanardag and Vishwanathan (2015) Tixier et al. (2019) Wu et al. (2018) Wang et al. (2019a) Du et al. (2019)
	IMDB-BINARY	Yanardag and Vishwanathan (2015)	1000	2	19.77	96.53	Yanardag and Vishwanathan (2015) Atamna et al. (2019) Wang et al. (2019a) Du et al. (2019)
	IMDB-MULTI	Yanardag and Vishwanathan (2015)	1500	3	13.00	65.94	Yanardag and Vishwanathan (2015) Atamna et al. (2019) Wang et al. (2019a) Bai et al. (2018) Bai et al. (2019a) Du et al. (2019)
	REDDIT-BINARY	Yanardag and Vishwanathan (2015)	2000	2	429.63	497.75	Yanardag and Vishwanathan (2015) Tixier et al. (2019)
	REDDIT-MULTI-5K	Yanardag and Vishwanathan (2015)	4999	5	508.52	594.87	Yanardag and Vishwanathan (2015) Tixier et al. (2019)
	REDDIT-MULTI-12K	Yanardag and Vishwanathan (2015)	11929	11	391.41	456.89	Yanardag and Vishwanathan (2015) Tixier et al. (2019)
Bioinformatics	D&D	Dobson and Doig (2003)	1178	2	284.32	715.66	Al-Rfou et al. (2019) Nikolentzos et al. (2017) Wu et al. (2018)
	ENZYMES	Borgwardt et al. (2005)	600	6	32.63	62.14	Atamna et al. (2019) Nikolentzos et al. (2017) Yanardag and Vishwanathan (2015)

Table 3 continued

Graph type	Datasets	Source	Number of graphs	Number of classes	Avg. number of nodes	Avg. number of edges	References
Chemoinformatics	PROTEINS	Borgwardt et al. (2005)	1113	2	39.06	72.82	Du et al. (2019), Narayanan et al. (2017), Nikolentzos et al. (2017), Yanardag and Vishwanathan (2015)
	AIDS	Riesen and Bunke (2008)	2000	2	15.69	16.20	Bai et al. (2018), Bai et al. (2019a), Fey et al. (2019)
	MUTAG	Debnath et al. (1991)	188	2	17.93	19.79	Al-Rfou et al. (2019), Du et al. (2019), Atamna et al. (2019), Narayanan et al. (2017), Nikolentzos et al. (2017), Wang et al. (2019a), Yanardag and Vishwanathan (2015)
	NCI1	Wale et al. (2008)	4110	2	29.87	32.30	Al-Rfou et al. (2019), Atamna et al. (2019), Du et al. (2019), Nikolentzos et al. (2017), Wang et al. (2019a), Yanardag and Vishwanathan (2015)
	NCI109	Wale et al. (2008)	4127	2	29.68	32.13	Narayanan et al. (2017), Nikolentzos et al. (2017), Yanardag and Vishwanathan (2015)
	PTC_MR	Helma et al. (2001)	344	2	14.29	14.69	Narayanan et al. (2017), Nikolentzos et al. (2017), Yanardag and Vishwanathan (2015)

Table 3 continued

Graph type	Datasets	Sourc	Number of graphs	Number of classes	Avg. number of nodes	Avg. number of edges	References
Brain Networks	ABIDE	Di Martino et al. (2014)	871	2	110	–	Ktena et al. (2018) Ma et al. (2019)
	UK Biobank	Biobank (2014)	2500	2	55	–	Ktena et al. (2018)
	HCP	Van Essen et al. (2012)	1200	2	360	–	Ma et al. (2019)
Image Graphs	COIL-DEL	Riesen and Bunke (2008)	3900	100	21.54	54.24	Li et al. (2019)

4.2 Evaluation

During evaluation, most GSL methods take pairs or triplets of graphs as input during training with various objective functions used for various graph similarity tasks. The existing evaluation tasks mainly include pair classification (Xu et al. 2017; Ktena et al. 2018; Ma et al. 2019; Li et al. 2019; Fey et al. 2019), graph classification (Tixier et al. 2019; Nikolentzos et al. 2017; Narayanan et al. 2017; Atamna et al. 2019; Wu et al. 2018; Wang et al. 2019a; Liu et al. 2019b; Yanardag and Vishwanathan 2015; Al-Rfou et al. 2019; Du et al. 2019), graph clustering (Wang et al. 2019a), graph distance prediction (Bai et al. 2018, 2019a; Fey et al. 2019), and graph similarity search (Wang et al. 2019c). Classification AUC (i.e., Area Under the ROC Curve) or accuracy are used as the most popular metric for the evaluation of graph-pair classification or graph classification task (Ma et al. 2019; Li et al. 2019). Mean squared error (MSE) is used as evaluation metric for the regression task in graph distance prediction (Bai et al. 2018, 2019a).

According to the evaluation results reported in the above works, the deep graph similarity learning methods tend to outperform the traditional methods. For example, Al-Rfou et al. (2019) shows that the deep divergence graph kernel approach achieves higher classification accuracy scores compared to traditional graph kernels such as the shortest-path kernel (Borgwardt and Kriegel 2005) and Weisfeiler–Lehman kernel (Kriege et al. 2016) in most cases for the graph classification task. Meanwhile, among the deep methods, methods that allow for cross-graph feature interaction tend to achieve a better performance compared to the factorized methods that relies only on single graph features. For instance, the experimental evaluations in Li et al. (2019) and Fey et al. (2019) have demonstrated that the GNN-based graph matching networks have superior performance than the Siamese GNNs in pair classification and graph edit distance prediction tasks.

The efficiency of different methods is also analyzed and evaluated in some of these works. In Bai et al. (2019a), some evaluations have been done for comparing the efficiency of the GNN based graph similarity learning approach SimGNN with traditional GED approximation methods including A*-Beamsearch (Neuhaus et al. 2006), Hungarian (Riesen and Bunke 2009) and VJ (Fankhauser et al. 2011), where the core operation for GED approximation may take polynomial or sub-exponential to the number of nodes in the graphs. For the GNN based model like SimGNN, to compute similarity scores for pairs of graphs, the time complexity mainly involves two parts: (1) the node-level and graph-level embedding computation stages, where the time complexity is $O(|E|)$, and $|E|$ is the number of edges of the graph (Kipf and Welling 2016); and (2) the similarity score computation stage, where the time complexity is $O(D^2 K)$ (D is the dimension of the graph-level embedding, and K is the feature map dimension used in the graph-graph interaction stage) for the strategy of using graph-level embedding interaction, and the time complexity is $O(DN^2)$ (N is the number of nodes in the larger graph). The experimental evaluations in Bai et al. (2019a) show that the GNN based models consistently achieve the best results in efficiency and effectiveness for the pairwise GED computation (Bai et al. 2019a) on

multiple graph datasets, demonstrating the benefit of using these deep models for the similarity learning tasks.

5 Applications

Graph similarity learning is a fundamental problem in domains where data are represented as graph structures, and it has various applications in the real world.

5.1 Computational chemistry and biology

An important application of graph similarity learning in the chemistry and biology domain is to learn the chemical similarity, which aims to learn the similarity of chemical elements, molecules or chemical compounds with respect to their effect on reaction partners in inorganic or biological settings (Brown 2009). An example is the compounds query for in-silico drug screening, where searching for similar compounds in a database is the key process.

In the literature of graph similarity learning, quite a number of models have been proposed and applied to similarity learning for chemical compounds or molecules. Among these work, the traditional models mainly employ sub-graph based search strategies or graph kernels to solve the problem (Zhang et al. 2013; Zeng et al. 2009; Swamidass et al. 2005; Mahé and Vert 2009). However, these methods tend to have high computational complexity and strongly rely on the sub-graph or kernels defined, making it difficult to use them in real applications. Recently, a deep graph similarity learning model SimGNN is proposed in Bai et al. (2019a) which also aims to learn similarity for chemical compounds as one of the tasks. Instead of using sub-graphs or other explicit features, the model adopts GCNs to learn node-level embeddings, which are fed into an attention module after multiple layers of GCNs to generate the graph-level embeddings. Then a neural tensor network (NTN) (Socher et al. 2013) is used to model the relation between two graph-level embeddings, and the output of the NTN is used together with the pairwise node embedding comparison output in the fully connected layers for predicting the graph edit distance between the two graphs. This work has shown that the proposed deep learning model outperforms the traditional methods for graph edit distance computation in prediction accuracy and with much less running time, which indicates the promising application of the deep graph similarity learning models in the chemo-informatics and bio-informatics.

5.2 Neuroscience

Many neuroscience studies have shown that structural and functional connectivity of the human brain reflects the brain activity patterns that could be indicators of the brain health status or cognitive ability level (Badhwar et al. 2017; Ma et al. 2017a, b). For example, the functional brain connectivity networks derived from fMRI neuroimaging data can reflect the functional activity across different brain regions, and people with brain disorder like Alzheimer's disease or bipolar disorder tend to have functional

activity patterns that differ from those of healthy people (Badhwar et al. 2017; Syan et al. 2018; Ma et al. 2016). To investigate the difference in brain connectivity patterns for these neuroscience problems, researchers have started to study the similarity of brain networks among multiple subjects with graph similarity learning methods (Lee et al. 2020; Ktena et al. 2018; Ma et al. 2019).

The organization of functional brain networks is complicated and usually constrained by various factors, such as the underlying brain anatomical network, which plays an important role in shaping the activity across the brain. These constraints make it a challenging task to characterize the structure and organization of brain networks while performing similarity learning on them. Recent work in Ktena et al. (2018), Ma et al. (2019) and Liu et al. (2019a) have shown that the deep graph models based on graph convolutional networks have a superior ability to capture brain connectivity features for the similarity analysis compared to the traditional graph embedding based approaches. In particular, Ma et al. (2019) proposes a higher-order Siamese GCN framework that leverages higher-order connectivity structure of functional brain networks for the similarity learning of brain networks.

In view of the work introduced above and the trending research problems in the field of neuroscience, we believe that deep graph similarity learning will benefit the clinical investigation of many brain diseases and other neuroscience applications. Promising research directions include, but are not limited to, deep similarity learning on resting-state or task-related fMRI brain networks for multi-subject analysis with respect to brain health status or cognitive abilities, deep similarity learning on the temporal or multi-task fMRI brain networks of individual subjects for within-subject contrastive analysis over time or across tasks for neurological disorder detection. Some example fMRI brain network datasets that can be used for such analysis have been introduced in Table 3.

5.3 Computer security

In the field of computer security, graph similarity has also been studied for various application scenarios, such as the hardware security problem (Fyrbiak et al. 2019), the malware indexing problem based on function-call graphs (Hu et al. 2009), and the binary function similarity search for identifying vulnerable functions (Li et al. 2019).

In Fyrbiak et al. (2019), a graph similarity heuristic is proposed based on spectral analysis of adjacency matrices for the hardware security problem, where evaluations are done for three tasks, including gate-level netlist reverse engineering, Trojan detection, and obfuscation assessment. The proposed method outperforms the graph edit distance approximation algorithm proposed in Hu et al. (2009) and the neighbor matching approach (Vujošević-Janičić et al. 2013), which matches neighboring vertices based on graph topology. Li et al. (2019) is the work that introduced GNN-based deep graph similarity learning models to the security field to solve the binary function similarity search problem. Compared to previous models, the proposed deep model computes similarity scores jointly on pairs of graphs rather than first independently mapping each graph to a vector, and the node representation update process uses an attention-based module which considers both within-graph and cross-graph informa-

tion. Empirical evaluations demonstrate the superior performance of the proposed deep graph matching networks compared to the Google's open source function similarity search tool (Dullien 2018), the basic GNN models, and the Siamese GNNs.

5.4 Computer vision

Graph similarity learning has also been explored for applications in computer vision. In Wu et al. (2014), context-dependent graph kernels are proposed to measure the similarity between graphs for human action recognition in video sequences. Two directed and attributed graphs are constructed to describe the local features with intra-frame relationships and inter-frame relationships, respectively. The graphs are decomposed into a number of primary walk groups with different walk lengths, and a generalized multiple kernel learning algorithm is applied to combine all the context-dependent graph kernels, which further facilitates human action classification. In Guo et al. (2018), a deep model called Neural Graph Matching Network is first introduced for the 3D action recognition problem in the few-shot learning setting. Interaction graphs are constructed from the 3D scenes, where the nodes represent physical entities in the scene and edges represent interactions between the entities. The proposed NGM Networks jointly learn a graph generator and a graph matching metric function in an end-to-end fashion to directly optimize the few-shot learning objective. It has been shown to significantly improve the few-shot 3D action recognition over the holistic baselines.

Another emerging application of graph similarity learning in computer vision is the image matching problem, where the goal is to find consistent correspondences between the sets of features in two images. As introduced at the end of Sect. 3.2, recently some deep graph matching networks have been developed for the image matching task (Jiang et al. 2019; Wang et al. 2019b), where images are first converted to graphs and the image matching problem is then solved as a graph matching problem. In the graph converted from an image, the nodes represent the unary descriptors of annotated feature points in images, and edges encode the pairwise relationships among different feature points in that image. Based on the new graph representation, the feature matching can be reformulated as graph matching problem. However, it is worth noting that, this graph matching is actually the graph node matching, as the goal is to match the nodes between graphs instead of two entire graphs. Therefore, the graph based image matching problem is a special case or a sub-problem of the general graph matching problem.

The two application problems discussed above are both promising directions of applying deep graph similarity learning models for the practical learning tasks in computer vision. A key advice we provide on applying graph similarity learning methods for these image applications is to first find an appropriate mapping for converting the images to graphs, so that the learning tasks on images can be formulated as the graph similarity learning based tasks.

6 Challenges

6.1 Various graph types

In most of the work discussed above, the graphs involved consist of unlabeled nodes/edges and undirected edges. However, there are many variants of graphs in real-world applications. How to build deep graph similarity learning models for these various graph types is a challenging problem.

Directed graphs. In some application scenarios, the graphs are directed, which means all the edges in the graph are directed from one vertex to another. For instance, in a knowledge graph, edges go from one entity to another, where the relationship is directed. In such cases, we should treat the information propagation process differently according to the direction of the edge. Recently some GCN based graph models have suggested some strategies for dealing with such directed graphs. In Kampffmeyer et al. (2019), a dense graph propagation strategy is proposed for the propagation on knowledge graphs, where two kinds of weight matrices are introduced for the propagation based on a node's relationship to its ancestors and descendants, respectively. However, to the best of our knowledge, no work has been done on deep similarity learning specifically for directed graphs, which arises as a challenging problem for this community.

Labeled graphs. Labeled graphs are graphs where vertices or edges have labels. For example, in chemical compound graphs where vertices denote the atoms and the edges represent the chemical bonds between the atoms, each node and edge have labels representing the atom type and bond type, respectively. These labels are important for characterizing the node-node relationship in the graphs, therefore it is important to leverage these label information for the similarity learning. In Bai et al. (2019a) and Ahmed et al. (2018), the node label information are used as the initial node representations encoded by a one-hot vector and used in the node embedding stage. In this case, the nodes with the same type share the same one-hot encoding vector. This should guarantee that even if the node ids are permuted, the aggregation results would be the same. However, the label information is only used for the node embedding process within each graph, and the comparison of the node or edge labels across graphs is not considered during the similarity learning stage. In Al-Rfou et al. (2019), both node labels and edge labels in the chemo- and bio-informatic graphs have been used as attributes for learning better alignment across graphs, which has been shown to lead to better performance. Therefore, how to leverage the node/edge attributes of the labeled graphs into the similarity learning process is a critical problem.

Dynamic and streaming graphs. Another type of graphs is the dynamic graph, which has a static graph structure and dynamic input signals/features. For example, the 3D human action or motion data can be represented as graphs where the entities are rep-

resented as nodes and the actions as edges connecting the entities. Then similarity learning on these graphs is an important problem for action and motion recognition. Moreover, another type of graph is the streaming graph, where both the structure and/or features are continuously changing (Ahmed et al. 2019; Ahmed and Duffield 2019). For example, online social networks (Ahmed et al. 2017a, 2014a, b). The similarity learning would be important for change/anomaly detection, link prediction, relationship strength prediction, etc. Although some work has proposed variants of GNN models for spatio-temporal graphs (Yu et al. 2017; Manessi et al. 2020), and other learning methods for dynamic graphs (Nguyen et al. 2018a, b; Tong et al. 2008; Li et al. 2017), the similarity learning problem on dynamic and streaming graphs has not been well studied. For example, in the multi-subject analysis of task-related fMRI brain networks as mentioned in Sect. 5.2, for each subject, a set of brain connectivity networks can be collected for a given time period, which forms a spatio-temporal graph. It would be interesting to conduct similarity learning on the spatio-temporal graphs of different subjects to analyze their similarity in cognitive abilities, which is an important problem in the neuroscience field. However, to the best of our knowledge, none of the existing similarity learning methods is able to deal with such spatio-temporal graphs. The main challenge in such problems is how to leverage the temporal updates of the node-level representations and the interactions between the nodes on these graphs while modeling their similarity.

6.2 Interpretability

The deep graph models, such as GNNs, combine node feature information with graph structure by recursively passing neural messages along edges of the graph, which is a complex process and makes it challenging to explain the learning results from these models. Recently, some work has started to explore the interpretability of GNNs (Ying et al. 2019; Baldassarre and Azizpour 2019). In Ying et al. (2019), a GNNEXPLAINER is proposed for providing interpretable explanations for predictions of GNN-based models. It first identifies a subgraph structure and a subset of node features that are crucial in a prediction. Then it formulates an optimization task that maximizes the mutual information between a GNN's prediction and the distribution of possible subgraph structures. Baldassarre and Azizpour (2019) explores the explainability of GNNs using gradient-based and decomposition-based methods, respectively, on a toy dataset and a chemistry task. Although these works have provided some insights into the interpretability of GNNs, they are mainly for node classification or link prediction tasks on a graph. To the best of our knowledge, the explainability of GNN-based graph similarity models remains unexplored.

6.3 Few-shot learning

The task of few-shot learning is to learn classifiers for new classes with only a few training examples per class. A big branch of work in this area is based on metric learning (Wang and Yao 2019). However, most of the existing work proposes few-shot learning problems on images, such as image recognition (Koch et al. 2015) and

image retrieval (Triantafillou et al. 2017). Little work has been done on metric learning for few-shot learning on graphs, which is an important problem for areas in which data are represented as graphs and data gathering is difficult, for example, brain connectivity network analysis in neuroscience. Since graph data usually has complex structure, how to learn a metric so that it can facilitate generalizing from a few graph examples is a big challenge. Some recent work (Guo et al. 2018) has begun to explore the few-shot 3D action recognition problem with graph-based similarity learning strategies, where a neural graph matching network is proposed to jointly learn a graph generator and a graph matching metric function to optimize the few-shot learning objective of 3D action recognition. However, since the objective is defined specifically based on the 3D action recognition task, the model can not be directly used for other domains. The remaining problem is to design general deep graph similarity learning models for the few-shot learning task for a multitude of applications.

7 Conclusion

Recently, there has been an increasing interest in deep neural network models for learning graph similarity. In this survey paper, we provided a comprehensive review of the existing work on deep graph similarity learning, and categorized the literature into three main categories: (1) graph embedding based graph similarity learning models, (2) GNN-based models, and (3) Deep graph kernels. We discussed and summarized the various properties and applications of the existing literature. Finally, we pointed out the key challenges and future research directions for the deep graph similarity learning problem.

References

Ahmed NK, Duffield N (2019) Network shrinkage estimation. arXiv preprint arXiv:1908.01087

Ahmed NK, Duffield N, Neville J, Kompella R (2014a) Graph sample and hold: a framework for big-graph analytics. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 1446–1455

Ahmed NK, Neville J, Kompella R (2014b) Network sampling: from static to streaming graphs. ACM Trans Knowl Discov Data 8(2):7

Ahmed NK, Neville J, Rossi RA, Duffield N (2015) Efficient graphlet counting for large networks. In: 2015 IEEE international conference on data mining, IEEE, pp 1–10

Ahmed NK, Duffield N, Willke TL, Rossi RA (2017a) On sampling from massive graph streams. Proc VLDB Endow 10(11):1430–1441

Ahmed NK, Neville J, Rossi RA, Duffield NG, Willke TL (2017b) Graphlet decomposition: framework, algorithms, and applications. Knowl Inf Syst 50(3):689–722

Ahmed NK, Rossi R, Lee JB, Willke TL, Zhou R, Kong X, Eldardiry H (2018) Learning role-based graph embeddings. arXiv preprint arXiv:1802.02896

Ahmed NK, Duffield N, Rossi RA (2019) Temporal network sampling. arXiv preprint arXiv:1910.08657

Ahmed NK, Rossi R, Lee J, Willke T, Zhou R, Kong X, Eldardiry H (2020) Role-based graph embeddings. IEEE Trans Knowl Data Eng

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

Arora S, Du SS, Hu W, Li Z, Salakhutdinov R, Wang R (2019) On exact computation with an infinitely wide neural net. arXiv preprint arXiv:1904.11955

Atamna A, Sokolovska N, Crivello JC (2019) SPI-GCN: a simple permutation-invariant graph convolutional network

Badhwar A, Tam A, Dansereau C, Orban P, Hoffstaedter F, Bellec P (2017) Resting-state network dysfunction in Alzheimer's disease: a systematic review and meta-analysis. Alzheimer's Dement Diagn Assess Dis Monit 8:73–85

Bai Y, Ding H, Sun Y, Wang W (2018) Convolutional set matching for graph similarity. arXiv preprint arXiv:1810.10866

Bai Y, Ding H, Bian S, Chen T, Sun Y, Wang W (2019a) Simgnn: a neural network approach to fast graph similarity computation. In: Proceedings of the 12th ACM international conference on web search and data mining, ACM, pp 384–392

Bai Y, Xu D, Gu K, Wu X, Marinovic A, Ro C, Sun Y, Wang W (2019b) Neural maximum common subgraph detection with guided subgraph extraction. https://openreviewnet/pdf?id=BJgcwh4FwS

Baldassarre F, Azizpour H (2019) Explainability techniques for graph convolutional networks. arXiv preprint arXiv:1905.13686

Berretti S, Del Bimbo A, Vicario E (2001) Efficient matching and indexing of graph models in content-based retrieval. IEEE Trans Pattern Ana Mach Intell 23(10):1089–1105

Biobank U (2014) About UK biobank. Available at https://www.ukbiobank.ac.uk/about-biobank-uk

Borgwardt KM, Kriegel HP (2005) Shortest-path kernels on graphs. In: Fifth IEEE international conference on data mining, IEEE, pp 8–pp

Borgwardt KM, Ong CS, Schönauer S, Vishwanathan S, Smola AJ, Kriegel HP (2005) Protein function prediction via graph kernels. Bioinformatics 21(suppl–1):i47–i56

Borgwardt KM, Kriegel HP, Vishwanathan S, Schraudolph NN (2007) Graph kernels for disease outcome prediction from protein–protein interaction networks. In: Biocomputing 2007, World Scientific, pp 4–15

Brown N (2009) Chemoinformatics—an introduction for computer scientists. ACM Comput Surv CSUR 41(2):8

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

Bunke H, Allermann G (1983) Inexact graph matching for structural pattern recognition. Pattern Recognit Lett 1(4):245–253

Bunke H, Shearer K (1998) A graph distance metric based on the maximal common subgraph. Pattern Recognit Lett 19(3–4):255–259

Cai H, Zheng VW, Chang KCC (2018) A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637

Chaudhuri U, Banerjee B, Bhattacharya A (2019) Siamese graph convolutional network for content based remote sensing image retrieval. Comput Vis Image Underst 184:22–30

Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27

Cui P, Wang X, Pei J, Zhu W (2018) A survey on network embedding. IEEE Trans Knowl Data Eng 31(5):833–852

Dai H, Dai B, Song L (2016) Discriminative embeddings of latent variable models for structured data. In: International conference on machine learning, pp 2702–2711

Debnath AK, Lopez de Compadre RL, Debnath G, Shusterman AJ, Hansch C (1991) Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. J Med Chem 34(2):786–797

Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the 30th international conference on neural information processing systems, pp 3844–3852

Di Martino A, Yan CG, Li Q, Denio E, Castellanos FX, Alaerts K, Anderson JS, Assaf M, Bookheimer SY, Dapretto M et al (2014) The autism brain imaging data exchange: towards a large-scale evaluation of the intrinsic brain architecture in autism. Mol Psychiatry 19(6):659–667

Dijkman R, Dumas M, García-Bañuelos L (2009) Graph matching algorithms for business process model similarity search. In: International conference on business process management, Springer, pp 48–63

Dobson PD, Doig AJ (2003) Distinguishing enzyme structures from non-enzymes without alignments. J Mol Biol 330(4):771–783

Douglas BL (2011) The weisfeiler-lehman method and graph isomorphism testing. arXiv preprint arXiv:1101.5211

Du SS, Hou K, Salakhutdinov RR, 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 5724–5734

Dullien T (2018) Functionsimsearch. https://github.com/google/functionsimsearch. Accessed 14 May 2018

Fankhauser S, Riesen K, Bunke H (2011) Speeding up graph edit distance computation through fast bipartite matching. In: International workshop on graph-based representations in pattern recognition, Springer, pp 102–111

Fey M, Lenssen JE, Morris C, Masci J, Kriege NM (2019) Deep graph matching consensus. In: International conference on learning representations

Fröhlich H, Wegner JK, Sieker F, Zell A (2006) Kernel functions for attributed molecular graphs-a new similarity-based approach to ADME prediction in classification and regression. QSAR Combin Sci 25(4):317–326

Fyrbiak M, Wallat S, Reinhard S, Bissantz N, Paar C (2019) Graph similarity and its applications to hardware security. IEEE Trans Comput 69(4):505–519

Gallicchio C, Micheli A (2010) Graph echo state networks. In: The 2010 international joint conference on neural networks, IEEE, pp 1–8

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 and data mining, ACM, pp 741–749

Gao H, Ji S (2019b) Graph u-nets. ICML

Gao H, Wang Z, Ji S (2018) Large-scale learnable graph convolutional networks. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 1416–1424

Garey MR, Johnson DS (1979) Computers and intractability, vol 174. freeman San Francisco

Gori M, Monfardini G, Scarselli F (2005) A new model for learning in graph domains. In: Proceedings. 2005 IEEE international joint conference on neural networks, 2005., IEEE, vol 2, pp 729–734

Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: a survey. Knowl Based Syst 151:78–94

Gretton A, Borgwardt KM, Rasch MJ, Schölkopf B, Smola A (2012) A kernel two-sample test. J Mach Learn Res 13:723–773

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, ACM, pp 855–864

Guillaumin M, Verbeek J, Schmid C (2009) Is that you? Metric learning approaches for face identification. In: 2009 IEEE 12th international conference on computer vision, IEEE, pp 498–505

Guo M, Chou E, Huang DA, Song S, Yeung S, Fei-Fei L (2018) Neural graph matching networks for fewshot 3d action recognition. In: Proceedings of the 15th European conference on computer vision, pp 653–669

Helma C, King RD, Kramer S, Srinivasan A (2001) The predictive toxicology challenge 2000–2001. Bioinformatics 17(1):107–108

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507

Horváth T, Gärtner T, Wrobel S (2004) Cyclic pattern kernels for predictive graph mining. In: Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 158–167

Hu X, Chiueh Tc, Shin KG (2009) Large-scale malware indexing using function-call graphs. In: Proceedings of the 16th ACM conference on computer and communications security, ACM, pp 611–620

Huang X, Cui P, Dong Y, Li J, Liu H, Pei J, Song L, Tang J, Wang F, Yang H, et al. (2019) Learning from networks: Algorithms, theory, and applications. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 3221–3222

Jacot A, Gabriel F, Hongler C (2018) Neural tangent kernel: convergence and generalization in neural networks. In: Advances in neural information processing systems, pp 8571–8580

Jiang B, Sun P, Tang J, Luo B (2019) Glmnet: graph learning-matching networks for feature matching. arXiv preprint arXiv:1911.07681

Jiang N, Liu W, Wu Y (2012) Order determination and sparsity-regularized metric learning adaptive visual tracking. In: 2012 IEEE conference on computer vision and pattern recognition, IEEE, pp 1956–1963

Johansson FD, Dubhashi D (2015) Learning with similarity functions on graphs using matchings of geometric embeddings. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 467–476

Kampffmeyer M, Chen Y, Liang X, Wang H, Zhang Y, Xing EP (2019) Rethinking knowledge graph propagation for zero-shot learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 11487–11496

Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907

Koch G, Zemel R, Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop, vol 2

Kriege NM, Giscard PL, Wilson R (2016) On valid optimal assignment kernels and applications to graph classification. In: Advances in neural information processing systems, pp 1623–1631

Ktena SI, Parisot S, Ferrante E, Rajchl M, Lee M, Glocker B, Rueckert D (2018) Metric learning with spectral graph convolutions on brain connectivity networks. NeuroImage 169:431–442

Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: International conference on machine learning, pp 1188–1196

Lee JB, Rossi RA, Kim S, Ahmed NK, Koh E (2019) Attention models in graphs: a survey. ACM Trans Knowl Discov Data 13(6):62

Lee JB, Kong X, Moore CM, Ahmed NK (2020) Deep parametric model for discovering group-cohesive functional brain regions. In: Proceedings of the 2020 SIAM international conference on data mining, SIAM, pp 631–639

Lee JE, Jin R, Jain AK (2008) Rank-based distance metric learning: an application to image retrieval. In: 2008 IEEE conference on computer vision and pattern recognition, IEEE, pp 1–8

Li J, Dani H, Hu X, Tang J, Chang Y, Liu H (2017) Attributed network embedding for learning in a dynamic environment. In: Proceedings of the 2017 ACM on conference on information and knowledge management, ACM, pp 387–396

Li Y, Gu C, Dullien T, Vinyals O, Kohli P (2019) Graph matching networks for learning the similarity of graph structured objects. In: Proceedings of the 36th international conference on machine learning, pp 3835–3845

Lim D, Lanckriet G, McFee B (2013) Robust structural metric learning. In: The 30th international conference on machine learning, pp 615–623

Ling X, Wu L, Wang S, Ma T, Xu F, Wu C, Ji S (2019) Hierarchical graph matching networks for deep graph similarity learning. https://openreviewnet/pdf?id=rkeqn1rtDH

Liu J, Ma G, Jiang F, Lu CT, Philip SY, Ragin AB (2019a) Community-preserving graph convolutions for structural and functional joint embedding of brain networks. In: 2019 IEEE international conference on big data, IEEE, pp 1163–1168

Liu S, Demirel MF, Liang Y (2019b) N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. In: Advances in neural information processing systems, pp 8464–8476

Ma G, He L, Cao B, Zhang J, Philip SY, Ragin AB (2016) Multi-graph clustering based on interior-node topology with applications to brain networks. In: Joint European conference on machine learning and knowledge discovery in databases, Springer, pp 476–492

Ma G, He L, Lu CT, Shao W, Yu PS, Leow AD, Ragin AB (2017a) Multi-view clustering with graph embedding for connectome analysis. In: Proceedings of the 2017 ACM on conference on information and knowledge management, ACM, pp 127–136

Ma G, Lu CT, He L, Philip SY, Ragin AB (2017b) Multi-view graph embedding with hub detection for brain network analysis. In: 2017 IEEE international conference on data mining, IEEE, pp 967–972

Ma G, Ahmed NK, Willke TL, Sengupta D, Cole MW, Turk-Browne NB, Yu PS (2019) Deep graph similarity learning for brain data analysis. In: Proceedings of the 28th ACM international conference on information and knowledge management, ACM, pp 2743–2751

Mahé P, Vert JP (2009) Graph kernels based on tree patterns for molecules. Mach Learn 75(1):3–35

Manessi F, Rozza A, Manzo M (2020) Dynamic graph convolutional networks. Pattern Recognit 97:107000

Mensink T, Verbeek J, Perronnin F, Csurka G (2012) Metric learning for large scale image classification: generalizing to new classes at near-zero cost. In: European conference on computer vision, Springer, pp 488–501

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119

Miller GL (1979) Graph isomorphism, general remarks. J Comput Syst Sci 18(2):128–142

Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y, Jaiswal S (2017) graph2vec: learning distributed representations of graphs. arXiv preprint arXiv:1707.05005

Neuhaus M, Riesen K, Bunke H (2006) Fast suboptimal algorithms for the computation of graph edit distance. In: Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR), Springer, pp 163–172

Nguyen GH, Lee JB, Rossi RA, Ahmed NK, Koh E, Kim S (2018a) Continuous-time dynamic network embeddings. In: Companion proceedings of the web conference 2018, international world wide web conferences steering committee, pp 969–976

Nguyen GH, Lee JB, Rossi RA, Ahmed NK, Koh E, Kim S (2018b) Dynamic network embeddings: from random walks to temporal random walks. In: 2018 IEEE international conference on big data, IEEE, pp 1085–1092

Nikolentzos G, Meladianos P, Vazirgiannis M (2017) Matching node embeddings for graph similarity. In: Thirty-first AAAI conference on artificial intelligence

Nikolentzos G, Siglidis G, Vazirgiannis M (2019) Graph kernels: a survey. arXiv preprint arXiv:1904.12218

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, ACM, pp 701–710

Riesen K, Bunke H (2008) Iam graph database repository for graph based pattern recognition and machine learning. In: Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR), Springer, pp 287–297

Riesen K, Bunke H (2009) Approximate graph edit distance computation by means of bipartite graph matching. Image Vis Comput 27(7):950–959

Rossi R, Ahmed N (2015) The network data repository with interactive graph analytics and visualization. In: Twenty-ninth AAAI conference on artificial intelligence

Rossi RA, Ahmed NK (2014) Role discovery in networks. IEEE Trans Knowl Data Eng 27(4):1112–1131

Rossi RA, Ahmed NK, Koh E (2018) Higher-order network representation learning. In: Companion proceedings of the the web conference 2018, international world wide web conferences steering committee, pp 3–4

Rossi RA, Ahmed NK, Koh E, Kim S, Rao A, Abbasi-Yadkori Y (2020a) A structural graph representation learning framework. In: Proceedings of the 13th international conference on web search and data mining, pp 483–491

Rossi RA, Jin D, Kim S, Ahmed NK, Koutra D, Lee JB (2020b) On proximity and structural role-based embeddings in networks: misconceptions, techniques, and applications. ACM Trans Knowl Discov Data

Rubner Y, Tomasi C, Guibas LJ (2000) The earth mover's distance as a metric for image retrieval. Int J Comput Vis 40(2):99–121

Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80

Schroff F, Kalenichenko D, Philbin J (2015) Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on Computer vision and pattern recognition, pp 815–823

Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681

Shuman DI, Narang SK, 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 Process Mag 30(3):83–98

Socher R, Chen D, Manning CD, Ng A (2013) Reasoning with neural tensor networks for knowledge base completion. In: Advances in neural information processing systems, pp 926–934

Swamidass SJ, Chen J, Bruand J, Phung P, Ralaivola L, Baldi P (2005) Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. Bioinformatics 21(suppl-1):i359–i368

Syan SK, Smith M, Frey BN, Remtulla R, Kapczinski F, Hall GB, Minuzzi L (2018) Resting-state functional connectivity in individuals with bipolar disorder during clinical remission: a systematic review. J Psychiatry Neurosci JPN 43(5):298

Tian Y, Zhao L, Peng X, Metaxas D (2019) Rethinking kernel methods for node representation learning on graphs. In: Advances in neural information processing systems, pp 11681–11692

Tixier AJP, Nikolentzos G, Meladianos P, Vazirgiannis M (2019) Graph classification with 2d convolutional neural networks. In: International conference on artificial neural networks, Springer, pp 578–593

Tong H, Papadimitriou S, Sun J, Yu PS, Faloutsos C (2008) Colibri: fast mining of large static and dynamic graphs. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 686–694

Triantafillou E, Zemel R, Urtasun R (2017) Few-shot learning through an information retrieval lens. In: Advances in neural information processing systems, pp 2255–2265

Tsitsulin A, Mottin D, Karras P, Müller E (2018) Verse: Versatile graph embeddings from similarity measures. In: Proceedings of the 2018 world wide web conference, international world wide web conferences steering committee, pp 539–548

Van Essen DC, Ugurbil K, Auerbach E, Barch D, Behrens T, Bucholz R, Chang A, Chen L, Corbetta M, Curtiss SW et al (2012) The human connectome project: a data acquisition perspective. Neuroimage 62(4):2222–2231

Vishwanathan SVN, Schraudolph NN, Kondor R, Borgwardt KM (2010) Graph kernels. J Mach Learn Res 11:1201–1242

Vujošević-Janičić M, Nikolić M, Tošić D, Kuncak V (2013) Software verification and graph similarity for automated evaluation of students' assignments. Inf Softw Technol 55(6):1004–1016

Wale N, Watson IA, Karypis G (2008) Comparison of descriptor spaces for chemical compound retrieval and classification. Knowl Inf Syst 14(3):347–375

Wallis WD, Shoubridge P, Kraetz M, Ray D (2001) Graph distances using graph union. Pattern Recognit Lett 22(6–7):701–704

Wang L, Zong B, Ma Q, Cheng W, Ni J, Yu W, Liu Y, Song D, Chen H, Fu Y (2019a) Inductive and unsupervised representation learning on graph structured objects. In: International conference on learning representations

Wang R, Yan J, Yang X (2019b) Learning combinatorial embedding networks for deep graph matching. arXiv preprint arXiv:1904.00597

Wang S, Chen Z, Yu X, Li D, Ni J, Tang LA, Gui J, Li Z, Chen H, Yu PS (2019c) Heterogeneous graph matching networks for unknown malware detection. In: Proceedings of the 28th international joint conference on artificial intelligence, pp 3762–3770

Wang Y, Yao Q (2019) Few-shot learning: a survey. arXiv preprint arXiv:1904.05046

Wu B, Yuan C, Hu W (2014) Human action recognition based on context-dependent graph kernels. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2609–2616

Wu B, Liu Y, Lang B, Huang L (2018) Dgcnn: disordered graph convolutional neural network based on the gaussian mixture model. Neurocomputing 321:346–356

Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst

Xu X, Liu C, Feng Q, Yin H, Song L, Song D (2017) Neural network-based graph embedding for cross-platform binary code similarity detection. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, ACM, pp 363–376

Yan X, Yu PS, Han J (2005) Substructure similarity search in graph databases. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, pp 766–777

Yanardag P, Vishwanathan S (2015) Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 1365–1374

Ying R, Bourgeois D, You J, Zitnik M, Leskovec J (2019) Gnn explainer: a tool for post-hoc explanation of graph neural networks. arXiv preprint arXiv:1903.03894

Yoshida T, Takeuchi I, Karasuyama M (2019) Learning interpretable metric between graphs: convex formulation and computation with graph mining. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 1026–1036

Yu B, Yin H, Zhu Z (2017) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875

Zanfir A, Sminchisescu C (2018) Deep learning of graph matching. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2684–2693

Zeng Z, Tung AK, Wang J, Feng J, Zhou L (2009) Comparing stars: on approximating graph edit distance. Proc VLDB Endow 2(1):25–36

Zhang D, Yin J, Zhu X, Zhang C (2018a) Network representation learning: a survey. IEEE Trans Big Data

Zhang S, Tong H, Xu J, Maciejewski R (2018b) Graph convolutional networks: Algorithms, applications and open challenges. In: International conference on computational social networks, Springer, pp 79–91

Zheng W, Zou L, Lian X, Wang D, Zhao D (2013) Graph similarity search with edit distance constraint in large graph databases. In: Proceedings of the 22nd ACM international conference on information and knowledge management, ACM, pp 1595–1600

Zhou J, Cui G, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2018) Graph neural networks: a review of methods and applications. arXiv preprint arXiv:1812.08434

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.