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

This is the published version of:

Sun, Ke, Wang, Lei, Xu, Bo, Zhao, Wenhong, Teng, Shyh Wei, & Xia, Feng. (2020). Network Representation Learning: From Traditional Feature Learning to Deep Learning. *IEEE Access, 8*, 205600–205617.

Available online at: https://doi.org/10.1109/ACCESS.2020.3037118

See this record in Federation ResearchOnline at:
https://researchonline.federation.edu.au/vital/access/manager/Index

Network Representation Learning: From Traditional Feature Learning to Deep Learning

KE SUN[1], **LEI WANG**[1], **BO XU**[1], **(Member, IEEE), WENHONG ZHAO**[2], **(Member, IEEE), SHYH WEI TENG**[3], **(Member, IEEE), AND FENG XIA**[3], **(Senior Member, IEEE)**

[1]Key Lab for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian 116620, China
[2]Ultraprecison Machining Center, Zhejiang University of Technology, Hangzhou 310014, China
[3]School of Engineering, IT and Physical Sciences, Federation University Australia, Ballarat, VIC 3353, Australia

Corresponding author: Feng Xia (f.xia@ieee.org)

ABSTRACT Network representation learning (NRL) is an effective graph analytics technique and promotes users to deeply understand the hidden characteristics of graph data. It has been successfully applied in many real-world tasks related to network science, such as social network data processing, biological information processing, and recommender systems. Deep Learning is a powerful tool to learn data features. However, it is non-trivial to generalize deep learning to graph-structured data since it is different from the regular data such as pictures having spatial information and sounds having temporal information. Recently, researchers proposed many deep learning-based methods in the area of NRL. In this survey, we investigate classical NRL from traditional feature learning method to the deep learning-based model, analyze relationships between them, and summarize the latest progress. Finally, we discuss open issues considering NRL and point out the future directions in this field.

INDEX TERMS Traditional feature learning, network representation learning, deep learning, graph analytics.

I. INTRODUCTION

Representation learning is a new paradigm in the machine learning field aiming at representing information efficiently. For example, in the linguistic domain, word vectors generated from Word2vec framework [1]–[3] embed semantic information into low dimensional vectors so that machines can better understand the words after word embeddings. In the setting of network representation learning (NRL), a better representation method can make subsequent learning tasks easier. In general, a better representation of a network can preserve the graph topology and cluster similar nodes together in the embedding space. Additionally, the representation learning is beneficial for lots of downstream tasks, e.g., clustering [4], node classification [5], and link-prediction [6]. They have been widely applied in bioinformatics [7], linguistics [1], transport network [8], [9] and social sciences [10]–[13], etc. Many information processing tasks mentioned above depend on how the data are represented. Meanwhile, with the development of big data, which presents Volume, Variety, and Velocity characteristics, a more effective data representation

method is required to achieve low cost and tractable computation.

In the past few decades, many traditional feature learning (TFL) algorithms e.g., principal component analysis (PCA) [14], isometric feature mapping (Isomap) [15], and local linear embedding (LLE) [16], have been proposed for reducing dimensions of data. Instead, NRL focuses on learning the vector representation of a node or a graph. In general, these two kinds of algorithms are then combined to take advantage of both of them.

However, these traditional methods could not effectively extract complex and nonlinear structured relationships of data. It is widely recognized that deep learning [17] has emerged as a powerful tool for extracting data features and has been applied in many fields, such as image processing and speech recognition. Network science researchers have applied deep learning models, including convolutional neural network [18], autoencoder neural network [19], recurrent neural network [20], and generative adversarial network [21], into graph-structured data and proposed the graph neural network (GNN) models [22]–[26]. In recent, many researchers try to introduce deep learning models, such as reinforcement learning, adversarial methods to graph learning [27], [28].

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.
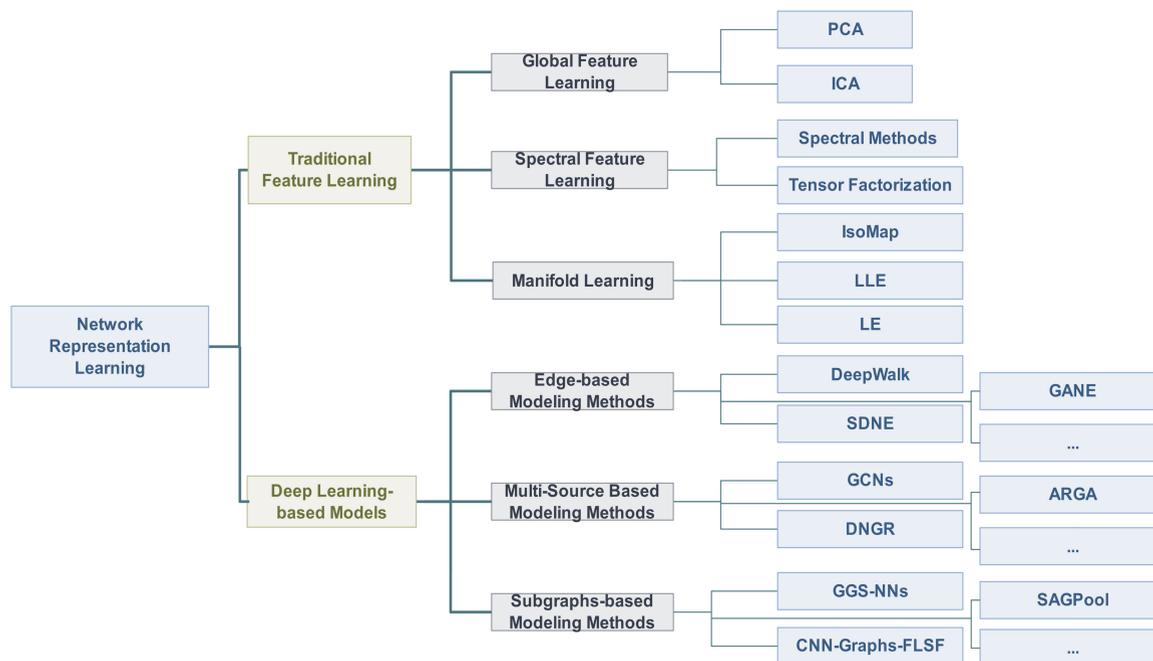
FIGURE 1. Categories of network representation learning algorithms.

In this survey, we provide a brief introduction to traditional representation algorithms and then mainly review NRL algorithms associated with deep learning models.

Representation learning models can be divided into "shallow" model and "deep" model. In this survey, we try to go through the development of data representation in graph-structured data from TFL to recent NRL based on deep learning. We do not intend to thoroughly summarize the various types of representation learning models in the literature. We will instead review TFL and the state-of-the-art representation learning methods mainly focusing on NRL technologies and network embedding algorithms. For understanding related algorithms, we first introduce the Word2vec framework as a basic model used by a large number of NRL algorithms to help understand what NRL is and the relationship between NRL and deep learning. Furthermore, we classify NRL into two categories, including TFL models and deep learning-based models. The overall organization of the categories of network representation learning algorithms are shown as Fig 1.

A. RELATED SURVEYS
There are already a few papers that summarized the algorithms about NRL. Our survey is different from all of them. We review several traditional feature learning algorithms and classical deep learning-based NRL models. We present deep learning-based NRL models based on the graph information they are keeping. A short introduction to the other related surveys is given as follows.

Hamilton *et al.* [29] focus on methods and applications of NRL. They discussed NRL in two main parts: embedding nodes and embedding sub-graphs. They proposed an encoder-decoder framework to organize various NRL models. Goyal *et al.* [30] mainly reviewed applications and performances of NRL, and gave detailed performance comparisons of NRL. Zhang *et al.* [31] gave a comprehensive categorization of NRL, including unsupervised methods, semi-supervised methods, and methods preserving network structure or vertex labels. The survey [32] did not review the whole variety of NRL algorithms but focused on methods for structure-preserving and property-preserving. In recent, many researchers turn attentions to graph neural network based models, and some surveys [33]–[36] specially investigated these works in the aspects of graph learning models and aggregation methods. These models are basically generated from deep learning models to graph, such as graph attention networks, graph autoencoders, and graph reinforcement learning. There are some surveys focusing on some special cases of NRL. For example, Yang *et al.* [37] reviewed heterogeneous NRL with analysis over benchmark and evaluation; Xie *et al.* [38] introduced dynamic network embedding from aspects of models.

The rest of our survey is organized as follows. We first present notations and graph related concepts in Section II. Then, we review the models of TFL in Section III and NRL based on deep learning models in Section IV. In the following two sections, we discuss the application of representation learning, and list several open issues. We provide our conclusions, draw our prospects of network representation learning in future research in Section VII.

II. NOTATIONS
This section presents TFL methods and NRL models based on deep learning methods. In the following, we first give the

TABLE 1. Terms and Notations.

G	The representation of a graph includes nodes and edges
V	The set of vertices of a graph
v	A vertice of a graph
E	The set of edges of a graph
L	Laplacian matrix of a graph
D	The degree matrix of a graph
A	Adjacency matrix of graph
i, j	The index of vertices
u	The new representation vector
W	The weight matrix of a graph
$\mathcal{L}$	The symbol of loss function

definitions of a Graph, Network embedding, and Laplacian matrix. Also, we list some terms and notations used in this article in Table 1.

Definition 1 (Graph): A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{Y})$ is a collection consisting of nodes (vertices or points) $V = \{v_1, v_2, \ldots, v_n\}$, edges $E = \{e_{i,j}\}_{i,j=1}^n$ and labels or side information associated to nodes Y. The edges between nodes can be directed or undirected.

Definition 2 (Network Embedding): Network embedding includes node embedding and edge embedding. Given a graph $\mathbf{G}$, the embedding function $f : \mathbf{V} \mapsto \mathbf{U}$ maps node $v_i \in \mathbf{V}$ to embedding vector $u_i \in \mathbf{U}$, where $\mathbf{V}$ represents vectors in the original space and $\mathbf{U}$ represents vectors in the projected space. Vector u_i is the newly learned node representation which often has low dimensions and preserves relevant network properties.

Definition 3 (Laplacian Matrix): In the graph theory, Laplacian matrix is a matrix representation of a graph. The Laplacian matrix of a simple graph is represented as: $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(\sum_{j:j \neq i} \mathbf{W}_{ij})$ is the degree matrix, $\mathbf{W}_{ij}$ is the weight between node i and j, and $\mathbf{A}$ is the adjacency matrix of a graph.

Representation Learning relies on an essential assumption in the manifold hypothesis [39], which refers to the real-world high dimensional data, such as images with two-dimensional manifolds embedding in the high dimensional space. Graph-structured data often has high dimensions and various types. Based on the manifold hypothesis, NRL algorithm could reduce the dimensions of graph data but keeps the internal relationship of nodes.

III. TRADITIONAL FEATURE LEARNING MODELS

Learning the intrinsic characteristics of data is always an enormous requirement for data science. In the past decade, a great deal of traditional feature learning or representation learning algorithms have been proposed in the machine learning domain such as kernel PCA [40] and kernel k-means [41]. They are a set of techniques allowing a system to automatically learn the latent features from raw data. These techniques are different from feature engineering which manually sets feature parameters. In this section, we mainly focus on TFL on graphs, and we separate them into three parts: global feature learning models, spectral learning based models, and manifold learning models. Global feature learning mainly focuses on preserving global information of data. Manifold learning aims to preserve local features and information, i.e., preserving property with the neighborhood of each node in a network.

A. GLOBAL FEATURE LEARNING

As mentioned above, global feature learning methods primarily focus on preserving global information of raw data in the learning feature space. In the following, we will present several algorithms belonging to this category.

1) PCA ALGORITHMS

PCA algorithm [40], is one of the earliest and most popular methods used to reduce the dimensions of the data. PCA is a linear, unsupervised, generative, and global feature learning method. There are lots of global feature learning algorithms, e.g., the variants of PCA, including sparse PCA [42] and robust PCA [43]. PCA can be used in NRL, such as dimension-reduced, and low-rank recovery of data. It can be applied for network data visualization and clustering in network science as well.

The classical PCA algorithm has several weaknesses, e.g., it lacks the ability to scale well to the number of data samples and is sensitive to outliers. In general, the reason that the classical PCA algorithm is susceptible to outliers data is caused by the quadratic term. Robust PCA can well overcome these shortcomings of classical PCA algorithm mentioned above and it is robust to occlusions and missing values by recovering the low-rank representation [43]. With the graph smoothness assumption, Shahid *et al.* [44] incorporated spectral graph regularization into robust PCA algorithm to improve the quality of clustering and dimension-reduced. The normalized graph Laplacian is defined as:

$$\mathbf{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \quad (1)$$

where $\mathbf{D} = \text{diag}(d_i)$ is the degree matrix and $\mathbf{A}$ is an adjacency matrix. In general, graph-Laplacian [45] is often combined with PCA methods [46]–[48] for feature extraction in the area of bioinformatics. The final experiments showed that the model can achieve better performance than the state-of-the-art models for the clustering and low-rank recovery tasks.

To accelerate the speed of robust PCA, Shahid *et al.* [49] proposed fast robust PCA (FRPCA) algorithms, which has the same advantages as the previous robust PCA. FRPCA has lower computational complexity for processing large scale datasets ($O(nlog(n))$) by using the FLANN library [50]. Motivated by the emerging field of signal processing on graphs [51], FRPCA adopted the idea of graph similarity, including feature similarity and data similarity, to enhance the

clustering quality in the new representation space. One problem of PCA algorithms is its high computational complexity. Here, FRPCA utilized Fast Iterative Soft Thresholding Algorithm (FISTA) [52] to solve this problem.

There are still many variants of PCA, such as sparse PCA (SPCA) [42] which extends classical PCA algorithm by adding sparsity constraint on input variables and can be well applied for multivariate datasets. Megasthenis *et al.* [53] introduced a variant of SPCA which accommodates the graph constraints to analyze financial data and the data in neuroscience. Min *et al.* [54] proposed Edge-group Sparse PCA (ESPCA) which combines the prior gene network with the PCA method for dimension-reduced and feature extraction.

2) ICA ALGORITHMS

Independent component analysis (ICA) [55] is a classical and powerful tool in signal processing, and also has been used to analyze the structured graph data. It is widely used to brain network analysis [56]–[58]. Park *et al.* [56] proposed a variant ICA, Graph ICA, to explore the changes of cognitive networks in the brain after completing a task. The concepts of Graph-ICA can be shown as follows:

$$g = [g_1, \ldots, g_M]^\top = W[s_1, \ldots, s_M]^\top = Ws, \qquad (2)$$

where weight matrix W represents the relationship strength between source (graph) s to compose g. The main ideas of the algorithm are to decompose measured graphs into common source graphs and then find these canonical network components from limited sets of data in neuroimaging. Diana *et al.* [57] directly utilized the ICA algorithm to extract features of different brain networks on the fMRI data and found the relationship among word learning with different parts of brain networks.

Ziegler *et al.* [58] designed a method combined with the ICA algorithm and then applied it for analyzing the resting-state fMRI. In short, the authors first used ICA to deal with neuronal components and then reconstructed them as weigh graphs. The method of calculating edge weighting in the graphs depends on the contribution to the specific component.

B. SPECTRAL LEARNING ON GRAPHS

The spectral learning, which is one kind of machine learning algorithm based on spectral methods, utilizes information in the eigenvectors of the target matrix to extract hidden structure. Most of the methods based on spectral learning just consider the structural information, so they could not apply to networks with complex information. In the following, we will discuss several common spectral learning frameworks, including spectral methods, singular value decomposition, and tensor factorization.

1) SPECTRAL METHODS

Spectral methods are fundamental for solving problems in engineering, applied mathematics, and statistics. More specifically, network researchers have used spectral methods to solve problems in network science, such as analyzing and visualizing networks.

Community detection is a popular problem in social networks. To address this problem, Newman *et al.* [59] proposed a spectral method as they find that tradition community detection methods based on maximum modularity and likelihood methods can be treated as spectral algorithms. The main idea of the algorithm is to utilize the matrix eigenvectors to represent the networks. The proposed algorithm contains three parts: (1) modularity maximization, (2) degree-corrected block model, and (3) normalized-cut graph partitioning. Zhang *et al.* [60] also applied the spectral method to detect communities while focused on overlapping communities in social networks. They adopted the K-medians algorithm to address the overlaps for graph clustering. In the real world, data are collected in different forms with various features and different structures. In order to deal with this situation, Li *et al.* [61] designed a spectral clustering algorithm based on the bipartite graph. The solution is the first k smallest eigenvectors. Moreover, the authors used a fast approximation algorithm to reduce the cost of computation of multi-view spectral clustering, so as to face the requirements of large-scale graph construction. Some researchers try to combine spectral methods with deep learning [62] by adopting stochastic optimization, which is successfully used to find meaningful subgoals in reinforcement learning environments.

2) TENSOR FACTORIZATION

The knowledge graph is a hot topic in network science, which can express data or information in the form of a graph with edges representing relations and nodes representing entities. It can be applied for recommender systems, search engines, etc. A group of researchers have applied tensor factorization methods to study NRL. Trouillon *et al.* [63] designed a method for link prediction called ComplEx, which is linear in both space and time. Besides, the algorithm exploited complex embeddings and utilized Hermitian dot product. Therefore, it is much simpler than neural tensor networks and holographic embeddings. Also, the algorithm is suitable for large datasets. Trouillon *et al.* [64] extended previous work [63] and utilized factorization models to study the knowledge graph. Moreover, the authors gave several proofs for the proposed model and more experiments, especially related to the training time of the models. The main idea of the algorithm is decomposing tensors into a product of embedding matrices with much lower dimensions.

C. MANIFOLD LEARNING

Manifold learning methods focus on preserving local similarity among data when the new representations are learned. In recent years, this algorithm is often used to deal with network analysis tasks. We will present several manifold learning algorithms, such as Isomap, Local Linear Embedding (LLE), and Laplacian Eigenmaps (LE). They are all based on the graph construction by exploiting manifold learning.

1) ISOMAP

Similar to PCA, Isomap algorithm [15] is a classical-dimension reduced approach, which is building on classical Metric Multidimensional Scaling (MDS) [65]. It is more powerful than other classical reduction methods as it could keep the nonlinear relations of original source. The main ideas of Isomap contain three steps: (1) constructing a neighborhood graph by using connectivity algorithm (such as KNN) from adjacency matrix; (2) computing the shortest path of entries as the geodesic distance; (3) finally, using MDS algorithm to obtain coordinate vector. The objective function is shown as follows:

$$\min \Sigma_{i \neq j,\dots,N}(d_{i,j} - ||u_i - u_j||)^2, \qquad (3)$$

where d represents the shortest path obtained from step (2), and u is the new representation vector that can be learned when minimizing Eq. (3). From Eq. (3), we can see that the optimized objective function is to make the distance between nodes in the new learned space similar to the distance in the original space. In other words, new low dimensional vectors approximately preserve the geodesic distance of the original data in the high dimensional space.

2) LOCAL LINEAR EMBEDDING

Local Linear Embedding (LLE) [16] is an another classical nonlinear dimension-reduced approach. This algorithm relies on the manifold hypothesis, and each node lies on its neighbors. Node features can be obtained from the summation of neighbor features, so that the algorithm has the ability to preserve the locally linear structure of neighborhood. Although LLE could preserve the structural information, it could only be used to undirected graph. LLE includes three main steps: (1) selecting neighbors for each node; (2) computing the weight $\mathbf{W}_{ij}$ which is the edge weight between the node and its neighbors:

$$\min \Sigma_i ||x_i - \Sigma_j \mathbf{W}_{ij} x_j||^2, \qquad (4)$$

(3) computing the new low dimensional representation from weights obtained from step (2), which is expressed as:

$$\min \Sigma_i ||u_i - \Sigma_j \mathbf{W}_{ij} u_j||^2. \qquad (5)$$

When minimizing (5), we can obtain the representation matrix U. In summary, LLE encodes the local information at each point into the reconstruction weights of its neighbors and then uses these weights to compute the low dimensional embeddings.

3) LAPLACIAN EIGENMAPS

Laplacian Eigenmaps (LE) [45] is a popular approach to find the low dimensional representation. Similar to the first step of LLE, LE first constructs a graph $\mathbf{G}$ by using the k nearest neighbors, and then uses the graph $\mathbf{G}$ to derive Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where the weight matrix W is generated by heat kernel method. The authors defined an objective function that makes connected points stay closer to each other, which is expressed as:

$$\Sigma ||u_i - u_j||^2 \mathbf{W}_{ij} = \text{tr}(\mathbf{U}^\top \mathbf{L} \mathbf{U}). \qquad (6)$$

When we minimize the above equation, the new representation matrix $\mathbf{U}$ can be obtained. In addition, TFL methods, such as Isomap, LLE, and LE, are all just applied for the undirected graph without external node information and focus on local features of the graph. However, they are not suitable for large-scale networks because obtaining eigenvector from large scale matrices has high computational complexity both in time and space.

IV. DEEP LEARNING-BASED MODELS

We have witnessed the superior performance of deep learning in many fields, and they have been widely applied for image classification, speech recognition, and object detection, etc. The deep architecture can extract latent information layer by layer from data, which contributes to the performance of data processing. More precisely, the original data is transformed by a nonlinear model to a more higher-level feature representation so as to achieve more abstract representation of data. There are several deep learning-based NRL models [66]–[70] proposed in recent years. The timeline of some representative methods of them are illustrated as Fig 2. Even though most of them are based on advanced models of deep learning, there have some methods having connections with traditional feature learning. We will introduce these in the related parts. In the following, we will focus on reviewing them from three subsections, as outlined in Table 2.

A. A TAXONOMY OF DEEP LEARNING-BASED NRL MODELS

The deep learning-based NRL models have different categories. We assign them into three categories: (1) Edge-based Modeling Methods, (2) Multi-source Based Modeling Methods, (3) Subgraphs Based Modeling Methods. As deep learning-based NRL models are the major concern in this article, we discuss and give a brief summary of them in Table 2. In the following subsections, We detail the characteristics of each algorithm belonging to the listed categories and provide a summary of them.

B. EDGE-BASED MODELING METHODS

Graph-structured data is a complex data type containing edges and nodes. In real world, graph edge can represent the link between users and products or links of friends. Lots of NRL algorithms just consider the structure of the graph, such as first-order proximity of nodes and second-order proximity of nodes. We cluster these NRL models by focusing on the graph structure as edge-based modeling methods. In addition, there are several NRL models based on Skip-Gram model [1], which is a powerful model in natural language processing. Moreover, the random walk approach [83] has been applied to capture graph structure. To understand these NRL algorithms deduced from Skip-Gram model, we start
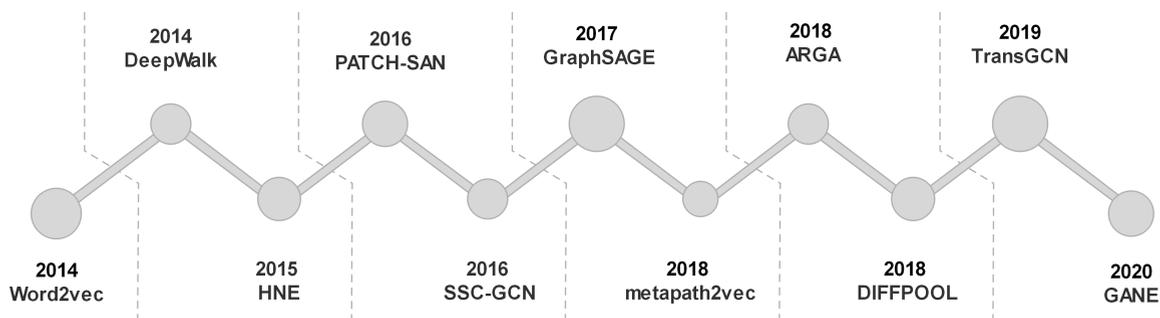
FIGURE 2. A timeline of some representative network representation learning methods.

TABLE 2. A Summary of NRL Algorithms According to the Information They are Preserving.

Category	Algorithms	Neural components	Proximities	Structure Sequence Technology
Edge-based Modeling Methods	Word2vec [1]	Neural Probabilistic Language Model	None	Random walk
	DeepWalk [71]	Skip-Gram Model	First-order, Second-order	
	node2vec [5]		Second-order, Higher-order	
	AIDW [72]	GANs		
	GraphGAN [68]			
	GANE [73]			None
	LINE [6]	None	First-order, Second-order	Adjacency matricx
	SDNE [67]	Autoencoder		
	DNGR [74]		Second-order, Higher-order	Surfing model
Multi-source Based Modeling Methods	HNE [23]	(Graph)Convolutional Neural Network	None	Linear transformation matrices
	PATCH-SAN [66]			Graph normaliztion
	SSC-GCN [22]		First-order	Layer-wise propagation rule
	Planetoid [75]	Feed-forward neural networks	Second-order, Higher-order	None
	TransNet [76]	Autoencoder	None	
	ARGA [77]	Autoencoder, GANs		Adjacency matrix
Subgraphs Based Modeling Methods	GGS-NNs [78]	Graph Neural Networks		None
	CNN-Graphs-FLSF [79]	Convolutional Neural Network		
	DIFFPOOL [80]	Graph Neural Network,Pooling	First-order	Layer-wise propagation rule
	HGP-SL [81]			
	SAGPool [82]			

with a brief introduction of Word2vec model [1] in this subsection.

1) Word2vec

Word2vec model [1], [3] is recognized as a powerful tool in natural language processing. It can reconstruct one-hot vector representations of words (word embedding). Actually, the framework of the model is a variant of neural probabilistic language model with three layers, and the word embedding representations are the matrices between input layer and hidden layer. After that, several literatures [84]–[86] have been proposed other variants to explain the principle of Word2vec. Levy *et al.* [85] pointed out that the neural word embedding is one kind of implicit matrix factorization. Given training words set $\{w_1, w_2, w_t, \ldots, w_n\}$, where t is the position in a text and the aim of Word2vec is to learn an estimated model:

$$F(\Theta, w_t, w_{t-1}, \ldots, w_{t-n+1}) = P(w_t|w_{t-1}, \ldots, w_{t-n+1}), \quad (7)$$

where function P is the conditional probability, function F is the function carried out by using a neural network and its free parameters, and Θ denotes the feature vector matrix (neural network matrix). The weight matrices can be learned by training the model when maximizing the empirical conditional probability of model F:

$$\max \sum_t F(\Theta, w_t, w_{t-1}, \ldots, w_{t-n+1}). \quad (8)$$

There are some variants of the model based on the way of normalization, e.g., softmax normalization and hierarchical softmax. The word vector is attracting interest due to the feature that semantic similarity words are located close to each other in the word vector space (representation space). In addition, the vectors can be computed by linear mathematical operations, for example, "king"−"queen"="man"−"women" as shown in Fig. 3. These intriguing property of the word vector shows that it contains semantic information existing in the real world.
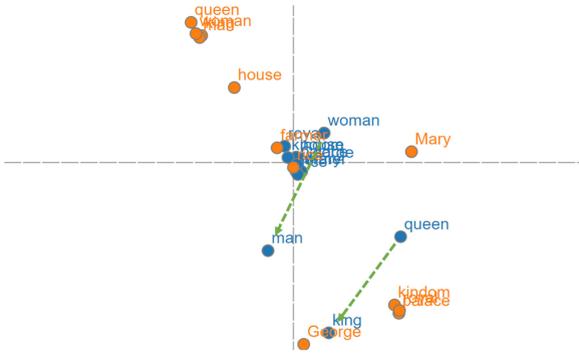
FIGURE 3. An example of word vectors embedding in two dimensional space.

FIGURE 4. Node2vec random walk strategy.

2) DeepWalk

DeepWalk [71] is a NRL algorithm, and it can learn latent representation of vertices in networks. The algorithm is the first generalization of Word2vec to networks. The truncated random walk approach is utilized to capture the graph structure, and then generate a sequence of vertices. However, the random walk approach is unbiased, which means it can not conduct breadth-first search or depth-first search on graph based on preferences. That provides a chance to improve the embedding performance by node2vec [5]. The frequency of the vertex appears in the sequences following power-law distributions, which is similar to the distribution of words in natural language. This is the main reason that Word2vec algorithm can be used to generalize the network-structured data. Given a random walk sequence $v_1, v_2, \ldots, v_l$, l is the length of word sequence. The training objective of DeepWalk is the same to the Word2vec algorithm. Given a previous vertex v_i, the likelihood of observing vertices $v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}$ is expressed as

$$P(v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}|v_i). \quad (9)$$

Now, learning an effective vertex representation $\Phi(v_i)$ ($\Phi(v_i) = \Theta^T \cdot v_i$) becomes an optimization problem:

$$\max \log P(v_{i-w}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+w}|\Phi(v_i)). \quad (10)$$

Different from the form of the adjacency matrix, vertex vector representation can avoid the data sparse problem, which can achieve higher computational efficiency. Furthermore, the random walk approach is leveraged to generate sequences of vertices based on local information. This characteristic enables DeepWalk to run on the distributed systems so as to meet the requirement of large-scale data processing.

3) Node2vec

Analogous to DeepWalk based on Word2vec, node2vec [5] algorithm was proposed by extending the Skip-gram architecture [1] to networks. The algorithm introduced a flexible neighborhood sampling strategy than DeepWalk, which captures network structure controlled by two hyperparameters p and q. They are used to interpolate random walk with breadth-first sampling or depth-first sampling. Given several
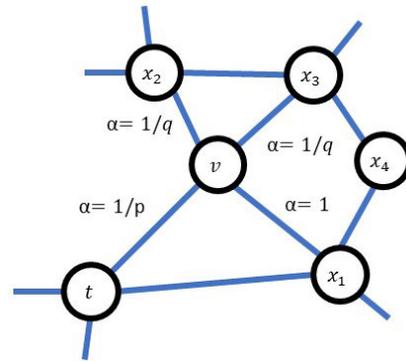
nodes t, v, x_1, x_2, x_3 as shown in Fig. 4, the unnormalized transition probability between v and x is decided by $\alpha_{pq}(t, x) \cdot w_{vx}$. The piecewise function $\alpha_{pq}(t, x)$ is expressed as

$$\alpha_{pq}(t, x) = \begin{cases} \dfrac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \dfrac{1}{q} & \text{if } d_{tx} = 2, \end{cases} \quad (11)$$

where w_{vx} is the static edge weight and d_{tx} denotes the shortest path distance between nodes t and x. Actually, the unbiased random walk strategy of DeepWalk is a special case of node2vec with $p = 1$ and $q = 1$. When tuning these parameters, the model can trade off the preference of focusing on the local structure or the global structure and therefore learns high quality and more information embeddings compared with DeepWalk.

4) LINE

Large-scale Information Network Embedding (LINE) algorithm [6] is not a deep learning based model, but is often compared with DeepWalk and node2vec [5]. Qiu *et al.* [87] pointed out that LINE, Node2vec and LINE can be implicitly categorized as matrix factorization frameworks. LINE is able to preserve the first-order proximity and second-order proximity. But it could not preserve the high-order proximity like node2vec. The first-order proximity refers to the proximity of two nodes connected with one-hop, and it can be measured by the joint probability distribution:

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i^\top \cdot \vec{u}_j)}, \quad (12)$$

where $\vec{u}_i$ and $\vec{u}_j$ stand for the vector representation of the nodes v_i and v_j, respectively. The second-order is similar to the first-order, but considers two nodes with a range of two-hop. Its proximity is the probability of the context node v_j generated by node v_i, i.e.,

$$p_2(v_j|v_i) = \frac{\exp(\vec{u}_j'^\top \cdot \vec{u}_i)}{\sum_k \exp(\vec{u}_k'^\top \cdot \vec{u}_i)}. \quad (13)$$
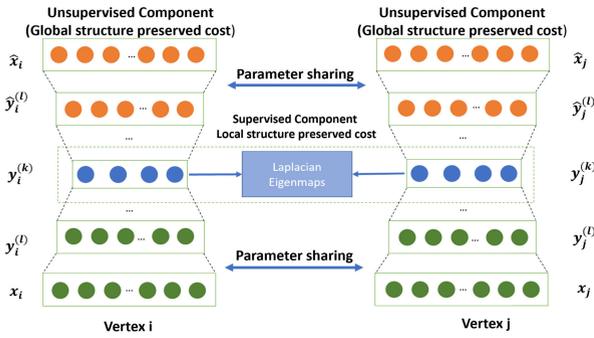
FIGURE 5. The structure of deep network embedding.

The second-order proximity means that nodes with similar distribution are similar to each other. To preserve the first-order proximity or the second-order proximity, the optimization objective of the algorithm tries to minimize the loss functions derived from KL-divergence between probability distribution and empirical distribution.

5) SDNE

Most NRL algorithms cannot extract the high nonlinear network-structured feature. Wang *et al.* [67] designed a semi-supervised model named Structure Deep Network Embedding (SDNE), which is a representative NRL model based on deep autoencoder approach [88]. The framework of SDNE is shown in Fig. 5. Similar to LINE which focuses on the graph-structured proximity of nodes, SDNE also preserves the first-order and second-order proximity of nodes. To address these structure-preserving and sparsity problems, the basic ideas of the algorithm are stated as two parts: (1) utilizing unsupervised component combining with deep autoencoder to preserve second-order proximity, which means that vertices with similar neighborhood stay close in the latent representation space; (2) using first-order proximity as the supervised information to make similar vertices more similar in the embedding space, where the objective function is based on Laplacian eigenmaps [89]. The loss function for second-order proximity is given by:

$$\mathcal{L}_{2nd} = \sum_{i=1}^{n} ||(\hat{r}_i - r_i) \odot b_i||_2^2, \tag{14}$$

where $\hat{r}_i$ denotes the reconstructed representation, and r_i is the input representation representing the neighborhood structure of the vertex. Notation $\odot$ represents the Hadamard product and $\mathbf{b_i} = \{b_{i,j}\}_{j=1}^{n}$ is used to impose more penalty to the reconstruction error of non-zero elements than zero elements, where

$$b_{i,j} = \begin{cases} \beta > 1 & s_{i,j} = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{15}$$

SDNE can preserve the local structure of network. The first-order proximity is adopted to represent the local network

structure and the loss function is expressed as

$$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{ij}||y_i^{(K)} - y_j^{(K)}||_2^2, \tag{16}$$

where s_{ij} is an instance from the the adjacency matrix $\mathbf{S}$, and y_i is the latent representation of node. As mentioned above, the objective of the above loss function is to make similar vertices more similar in the embedding space by utilizing the supervised information. The two loss functions are all distance-based model similar to Isomap [15], LE [45], and LLE [16]. To preserve the first-order and second-order proximity, the mix loss function combined $\mathcal{L}_{1st}$ with $\mathcal{L}_{2nd}$ is defined as

$$\mathcal{L} = \mathcal{L}_{2nd} + \alpha \mathcal{L}_{1st} + \nu \mathcal{L}_{reg}, \tag{17}$$

where $\mathcal{L}_{reg}$ is a regularizer term to avoid overfitting. The final latent representation of vertices can be achieved when the above mix loss function is minimized.

6) DNGR

Analogous to SDNE depending on the deep neural network model, Deep Neural Graph Learning (DNGR) [74] is another NRL algorithm incorporating deep autoencoders with network features. In contrast to algorithms using a truncated random walk, such as DeepWalk and node2vec, the DNGR algorithm utilizes random surfing model to overcome the drawback that they cannot capture weighted graphs and cope with evolved graphs. Two important contributions are stated in [74]: (1) designing a random surfing model motivated by PageRank mode, which can be directly applied for weighted graphs and product the probabilistic co-occurrence (PCO) matrix; (2) demonstrating a novel model for accurately learning vertex representation of weighted graphs. The main ideas of DNGR model are to transform the PCO matrix captured by random surfing model to positive pointwise mutual information (PPMI) matrix and then feed them into stacked denoising autoencoder [90] so as to learn the vertex latent representation. The main processes are shown in Fig. 6.

7) AIDW

Most NRL algorithms ignore the robustness of representation. To overcome this weakness, The authors [72] proposed an Adversarial Inductive DeepWalk (AIDW) model consisting of structure preserving component and an adversarial learning component. AIDW could well preserve structure information while having robustness to representation. The trick behind AIDW is introducing the adversarial learning model including a discriminator and a generator (structure preserving component) to enhance the representation from the structure preserving component of AIDW. However, the adversarial learning model always has high computation time because it conducts game playing between discriminator and generator. Here, the discriminator is trained to differentiate between feature vectors and prior samples. The loss
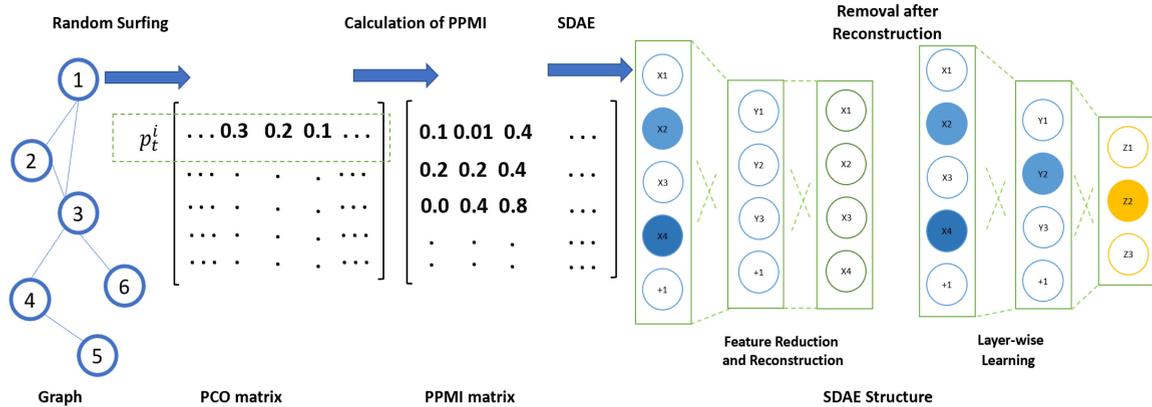
FIGURE 6. Deep neural graph learning components.

function of discriminator is expressed as:

$$O_D(\theta_2) = \mathbb{E}_{z \sim p(z)}[\log \mathbf{D}(\mathbf{Z}; \theta_2)]$$
$$+ \mathbb{E}_x[\log(1 - \mathbf{D}(\mathbf{G}(x; \theta_1); \theta_2))], \quad (18)$$

where $\mathbf{G}(*; *)$ is a generator. Here, the two models improve their performance by using the minimax game mechanism.

The authors [25] adopted a similar policy with AIDW that use GAN to enhance the performance of embedding, and proposed a novel NRL algorithm with adversarially regularized autoencoders (NetRA). Specifically, NetRA involves LSTM to product positive samples to feed generative model. However, the regularization of NetRA is static. To further improve the ability of adversarial training on graph, the authors [91] developed a framework, which could dynamically regularize with graph structure.

C. MULTI-SOURCE BASED MODELING METHODS

Besides graph-structured data, there are other types of information, including vertex attributes and vertex labels, etc. We call this information as multi-source of nodes. It is no doubt that efficiently using these data can enormously improve the performance of network representation. In recent years, many studies have focused on the multi-source embedding of graphs, such as considering labels information [92] and heterogeneous network embedding [23]. In this section, we will illustrate the NRL algorithms, which consider both graph structure and the features of nodes.

1) HNE

Research in NRL focuses on homogeneous networks rather than heterogeneous networks. Chang *et al.* [23] designed Heterogeneous Network Embedding (HNE) algorithm to leverage deep learning architectures. Besides, HNE has several key advantages than traditional linear embedding models, such as being able to handle the dynamic networks, being suitable for network-oriented data mining applications. To transform different types of data into a uniform representation space,

a relatively linear transform matrix is introduced:

$$\tilde{x} = \mathbf{U}^\top x, \ and \ \tilde{z} = \mathbf{V}^\top z, \quad (19)$$

where $\mathbf{U}$ and $\mathbf{V}$ denote the linear transformation matrices, and $\tilde{x}$ and $\tilde{z}$ are the transformed samples. The way of linear transformation is often used to transform embedding space by traditional feature learning based on linear projection like PCA [40], LLE [16]. Importantly, to represent the similarity between two data points, the inner product is used in the projected space. Based on that, to denote relationship of heterogeneous linkages in networks, a decision function is designed

$$d(x_i, x_j) = s(x_i, x_j) - t_{II}, \quad (20)$$

where $s(*, *)$ denotes the inner product of two samples of x and z respectively., and t_{II} is a bias-based value. Generally speaking, most representation learning algorithms can be seen as mathematical optimization problems, and the loss function of HNE is defined as:

$$\mathcal{L}(x_i, x_j) = \log(1 + \exp(-\mathbf{A}_{i,j} \cdot d(x_i, x_j))). \quad (21)$$

Actually, the above equation can be regarded as a binary logistic regression. Another fundamental characteristic of HNE is a deep structure including a CNN structure with fully connected layers to learn features of image and text, and thus it can model complex networks with heterogeneous components.

There are various deep learning-based methods for dealing with heterogeneous networks. For example, metapath2vec [93] utilizes random walks method to capture graph structure information and then feeds them to HeterogeneousSkipGram to embed vectors. HAN [94] involves attention mechanism to improve embedding performance. Zhang *et al.* [95] proposed a GNN-based heterogeneous networks embedding algorithm, namely HetGNN. The authors consider that GNN could capture the rich neighborhood information. Existing approaches focus primarily on static networks, while a HIN in reality is usually changing with time.

Zhang *et al.* [96] developed a dynamic heterogeneous network embedding algorithm utilizing hierarchical attentions mechanism. There are some traditional methods for heterogeneous network embedding, such as TransN [97] based on dual-learning mechanism, RHINE [98] based on euclidean distance and translation-based distance.

2) PLANETOID

In the real world, most datasets are composed of unlabeled data. How to leverage a large amount of unlabeled data to improve data analysis performance is still a considerable challenge. To represent unlabeled data in the graph, Yang *et al.* [75] designed a novel semi-supervised learning algorithm for graph embedding (Planetoid). The authors specially developed two variants methods containing transductive graph embedding and inductive graph embedding. The transductive graph embedding is applied for predicting class label and graph context based on the input feature of observed labeled data and embeddings extracted from graph structure. The loss function of transductive graph embedding is expressed as:

$$-\frac{1}{L}\Sigma_{i=1}^{L} \log \mathrm{p}(y_i|x_i, e_i) - \lambda\mathbb{E}_{(i,c,\gamma)}\log \sigma(\gamma w_c^\top e_i), \quad (22)$$

where the first term is the probability of predicting labels, and the second term is the loss function for predicting graph context. To generalize unobserved instances, the inductive learning relies on the input feature x and the embedding works as a parameterized function of the feature of x. Similar to the loss function of transductive learning, the loss function here is defined as:

$$-\frac{1}{L}\Sigma_{i=1}^{L} \log \mathrm{p}(y_i|x_i) - \lambda\mathbb{E}_{(i,c,\gamma)}\log \sigma(\gamma w_c^\top \mathrm{h}^{l_1}(x_i)). \quad (23)$$

Compared with Eq. (22), Eq. (23) replaces embedding of instance e_i with embedding $\mathrm{h}^{l_1}(x_i)$. In addition, Planetoid framework is based on feed-forward neural networks, of which the stochastic gradient descent (SGD) is adopted to train the model in mini-batch mode.

3) PATCHY-SAN

From the view that arbitrary graph can be seen as an image, Niepert *et al.* [66] proposed a deep learning-based algorithm: PATCHY-SAN, for learning arbitrary graph by integrating CNN. The algorithm opens up a novel perspective that deep learning methods can be used to solve graph embedding problems. The main idea of the algorithm is transforming graph data to a special form combined with existing convolutional network components. PATCHY-SAN model contains four steps: (1) node sequence selection; (2) neighborhood graph construction; (3) normalizing the extracting neighborhood graph; (4) combining with existing CNN components, which is illustrated in Fig. 7. In step (3), in order to optimize graph normalization, an optimal normalization problem is defined to find the optimal labeling approach $\hat{l}$, which can assign similar structural nodes to the same relative position in the
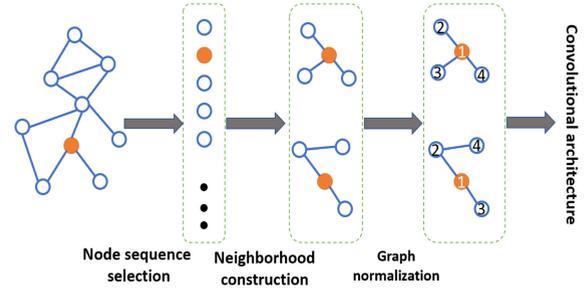
FIGURE 7. CNN for graph architecture.

adjacency matrices for a given collection of graphs

$$\hat{l} = \arg\min_{l} \mathbb{E}_g[|\mathrm{d}_A(\mathbf{A}^l(\mathbf{G}), \mathbf{A}^l(\mathbf{G}')) - \mathrm{d}_G(\mathbf{G}, \mathbf{G}')|], \quad (24)$$

where g is a collection of unlabeled graphs, l denotes an injective graph labeling procedure, $\mathbf{A}^l(\mathbf{G})$ is a unique adjacency matrix of graph $\mathbf{G}$ determined by labeling procedure l, d_G denotes the distance between graphs based on nodes and d_A based on matrices. From the equation, the optimal labeling produce $\hat{l}$ can be obtained when the expected difference between the above two types of distance is minimized.

4) GCNs

Different from the above frameworks based on Word2vec, Kipf and Welling [22] proposed a semi-supervised graph convolutional networks (GCNs) considering graph structure and node label information. The authors refined and optimized the previous GCN model proposed by Bruna *et al.* [99] and successfully draw attentions of researchers to graph neural network. This previous GCNs model is based on spectral graph regularization, which is often used by traditional feature learning methods like robust PCA [44]. GCNs is a scalable approach, which can be directly applied for graph-structured data. The key innovation of the algorithm is introducing an effective neural network layer-wise propagation rule for graphs

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \quad (25)$$

where $\sigma(\cdot)$ denotes the nonlinear activation function, such as function tanh, $\mathbf{W}^{(l)}$ is the free weight matrix of a layer, and $\tilde{\mathbf{A}}$ represents the adjacency matrix of an undirected graph with added self-connections. $\mathbf{H}^{(l)}$ is the matrix of activations in the *lth* layer, for example, $\mathbf{H}^0 = \mathbf{X}$ (feature matrix) and $\mathbf{H}^l = \mathbf{Z}$ (The final desired feature matrix). GCNs is a differentiable generalization of the Weisfeiler-Lehman algorithm [100] as its propagation rule can be interpreted as a variant of a hash function of that. To achieve semi-supervised learning, graph-based regularization [101] is adopted to learn graph feature, and it includes two steps: (1) getting embedding of nodes; (2) training classifier on the embeddings.

There are some GCNs-based variant methods for improving GCNs capability from different aspects. Li *et al.* [102] proved that GCNs is actually a special form of Laplacian

smoothing and then proposed the co-training and self-training approaches to improve the learning efficiency of GCNs framework. Chen *et al.* [103] developed control variate based algorithms to overcome the receptive field size growing problem of GCNs so as to arrive comparable convergence speed. Hamilton *et al.* [7] designed an improved GCNs model: GraphSAGE with learnable aggregation functions rather than using graph Laplacian, which allows GCNs to apply to a large graph. Chen *et al.* [104] further improves the sampling algorithm based on GraphSAGE and obtain a better computational efficiency to a large graph. There are many other variant GCNs-based models future enhancing performances like SGCN [105], mGCN [106], and Deep-GCNs [107].

Inspired by the success of GCNs, several researchers involved this model to knowledge graph representation. Schlichtkrull *et al.* [108] first utilized the GCN and proposed the R-GCN. The authors designed a matrix transform method to represent the relations between facts. This method could address the embedding problem caused by too many types of relationships. Cai *et al.* [109] combined the TransE [110] with GCN and proposed TransGCN, which could be directly used to link prediction of the heterogeneous relations knowledge graph, while has less parameters than R-GCN [108]. Wang *et al.* [111] further improved the propagation model, proposed the logical attention network (LAN). The model considers the disorder and inequality nature of entities, so as to well learn relations between the entities and the corresponding neighbors.

5) TransNet
Realizing that there are rich semantic information on edges, Tu *et al.* [76] proposed TransNet-based NRL model to extract social relationships from networks, and the interactions between nodes can be regarded as a translation operation. Instead of utilizing CNN, the algorithm designed an auto-encoder framework to learn edge latent representation. Autoencoder is utilized to reconstruct edge labels and vertex vectors. These vertex vectors of edges stay in a continued space and we have

$$\mathbf{u} + \mathbf{l} \approx \mathbf{v}', \qquad (26)$$

where $\mathbf{u}$ and $\mathbf{v}'$ denote the representations of vertices, and l is the edge representation derived from label set l. To minimize the distance $\mathbf{d}(*, *)$ at the left and right side of (26), a hinge-loss is defined as

$$\mathcal{L}_{\text{trans}} = \max(\gamma + \mathrm{d}(u + \mathrm{l}, \mathrm{v}') - \mathrm{d}(\hat{u} + \hat{\mathrm{l}}, \hat{\mathrm{v}'}), 0), \quad (27)$$

where $\mathbf{L}_1$ norm is adopted, $\gamma > 0$ is a margin hyper-parameter, and $(\hat{u}, \hat{v}, \hat{l})$ denotes a negative sample of original variables from the negative sampling set. Similar to SDNE framework, the deep model used by TransNet for learning edge representation is deep autoencoder, and the reconstructed loss function is a distance-based model similar to LE [45], LLE [16]. The loss function is expressed as

$$\mathcal{L}_{\text{rec}} = ||(s - \hat{s}) \odot \mathrm{x}||, \qquad (28)$$

where s and $\hat{s}$ denote input and output, respectively. Finally, a joint optimization objective is defined by integrating the loss functions mentioned above

$$\mathcal{L} = \mathcal{L}_{\text{trans}} + \alpha[\mathcal{L}_{\text{ae}}(l) + \mathcal{L}_{\text{ae}}(\hat{l})] + \eta\mathcal{L}_{\text{reg}}, \qquad (29)$$

where λ and α are two hyper-parameters to regularize the importance of different parts of the model.

6) ARGA
Previous works have proved that GCNs is a powerful tool to represent graph-structured data. Pan *et al.* [77] proposed a novel adversarial graph embedding framework, namely ARGA leveraging GCNs as a graph encoder. Similar to AIDW, ARGA utilized GAN to enhance robustness of embedding while preserving structure and node label information. To keep both structure and node label information, the authors developed a variant encoder based on GCNs, defined as follows:

$$\begin{aligned} \mathbf{Z}^1 &= \mathrm{f}_{\text{Relu}}(\mathbf{X}, \mathbf{A}|\mathbf{W}^{(0)}), \\ \mathbf{Z}^2 &= \mathrm{f}_{\text{linear}}(\mathbf{Z}, \mathbf{A}|\mathbf{W}^{(1)}), \end{aligned} \qquad (30)$$

where $\mathbf{X}$ represents the node content, $\mathbf{A}$ is the graph-structured information, such as adjacency matrix.

There are some other GAN-based network embedding algorithms. For example, HeGAN [112] utilize a generator to produce negative samples so as to achieve better embedding performance for heterogeneous information networks; Graphite [113] is variational autoencoders based embedding algorithm while involving iterative message passing procedure to raise performance, etc.

D. SUBGRAPHS-BASED MODELING METHODS
The NRL algorithms mentioned above just seek to address node embedding. In addition, there are some requirements for learning the representation of subgraphs or the whole graph, which refers to a set of nodes and edges, such as protein and molecules. Subgraphs embedding can be applied for learning molecular fingerprints [114] and predicting multicellular function [115], etc. In addition, the fixed-size subgraphs can be treated as motifs [116], [117] or graph kernel [118]. However, we will not discuss them but primarily focus on the deep learning-based NRL model.

1) GGS-NNs
To deal with graph-structured data, Li *et al.* [78] proposed a novel graph-based neural network model called Gated Graph Sequence Neural Networks (GGS-NNs) by extending the previous related GNNs model [119]. Being different from the feature learning algorithm GNNs, the modification specially used gated recurrent units [120] and modern optimization techniques. In addition, GGS-NN can produce sequence outputs, e.g., paths on a graph, rather than a single output. The basic component of GGS-NNs is GG-NNs containing three main parts: (1) node annotation process initializes node
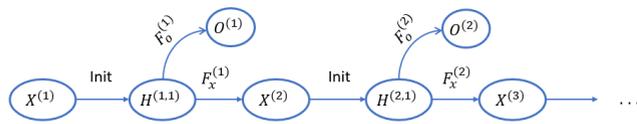
FIGURE 8. GGS-NNs architecture.

representation; (2) propagation model computes node representation of each of them; (3) output model, i.e., the model $\mathbf{o}_v = g(\mathbf{h}_v^{(T)} x_v)$ maps node representations with their labels to outputs, where the notations h_v and x_v denote a representation of node and node label, respectively. In graph level outputs, a graph level representation vector is defined as:

$$\mathbf{h}_g = \tanh(\sum_{v \in V} \sigma(\mathbf{i}(\mathbf{h}_v^{(T)}, x_v)) \odot \tanh(\mathbf{j}(\mathbf{h}_v^{(T)}, x_v))), \quad (31)$$

where $\sigma(*, *)$ works as a soft attention mechanism by deciding the relevant nodes to the current graph level task. $\mathbf{i}(*, *)$ and $\mathbf{j}(*, *)$ are neural networks for computing inputs h_v, x_v to real-valued vectors. The last two parts are the core processes, which map the graph to the output. As mentioned above, GGS-NNs can produce sequence outputs, e.g., $o^{(1)}, o^{(2)}, \ldots, o^{(K)}$, that is different from most graph representation learning algorithms. The architecture shown in Fig. 8 contains several GG-NNs operating in sequence to produce sequence outputs.

There are some new GNN-based models with innovations of involving new pooling strategies for graph-level representation. For example, Ying *et al.* [80] proposed a differentiable pooling strategy, which can learn hierarchical representations of graphs but it has the drawback of high computational complexity; Zhang *et al.* [81] also designed a hierarchical graph pooling method without parameter. The proposed method has good performance of keeping key substructures of graph. Lee *et al.* [82] designed a pooling method on graph with involving self attention mechanism. This mechanism could pilot model focusing on important features like a human.

2) CNN-GRAPHS-FLSF

Generalizing CNN to graph-structured data is always a great challenge. To overcome this problem, a novel convolutional neural network model was combined with fast localized spectral filtering for graphs (CNNs-Graphs) in [79]. CNN is a powerful tool in images, video tasks, and natural language processing. There are several contributions to generalize the classical CNN from low dimensional data to high dimensional irregular domains, e.g., social networks and brain connectomes. At first, a novel convolutional filter named fast localized spectral filter on graphs was proposed by enhancing the previous GCNs algorithm [99]. The fast localized spectral filter is a spectral approach based on spectral graph theory [121]. In general, the convolutional filter can be applied to recognize identical features of data. Spectral approaches are used here to offer a well-defined localization operator on graphs in spectral domain [51]. Here, the graph Laplacian matrix [121] is leveraged as an essential operator in spectral

graph analysis, and the formula definition has been given in Section 2 Definition 3.

In order to improve the computational efficiency of the CNN-Graphs model, the polynomial filter is first utilized to reduce the learning complexity, which can achieve the same complexity as classical CNN. In addition, to further reduce the model complexity, Chebyshev expansion [122] was utilized to overcome the high computational cost caused by the multiplication with the Fourier basis. Second, Graph coarsening is a necessary operation because pooling operation needs a valid group of data (they are neighborhoods for a graph) and coarsening phase of Graclus multilevel clustering algorithm [123] was employed to group similar vertices in graphs. Last, an efficient and fast pooling strategy was proposed by constructing a balanced binary tree and then applied pooling operation on the rearranged vertices of the graph.

V. APPLICATIONS

There are many NRL algorithms proposed based on deep learning models. In practice, NRL and network embedding are commonly used in graph analytic tasks, such as node classification, link prediction, clustering, and graph visualization. In this section, we will discuss the applications of these NRL algorithms in the following.

A. NODE CLASSIFICATION

Classification refers to dividing items into different categories. In network science, the most common applications in the graph analysis task include node classification [124] and graph classification. In addition, node or graph classification is often used as a benchmark to evaluate the performance of node embedding. In the node classification application, labels are able to indicate different information, such as interest and affiliations. However, due to the limited amount of labeled data in reality and only a few available labels, semi-supervised learning is often considered to enhance the performance of the node classification task [22], [125], or text classification [75]. The learned latent representation is a real-value vector, which is convenient to be combined with traditional classification algorithm to address the problems, such as multi-label classification [5], [71], [126]. Besides, Wang *et al.* [67] proposed a semi-supervised deep embedding algorithm, which can be used for link prediction and multi-label classification in blog-catalog networks. In heterogeneous networks, e.g., social networks, various types of network data take a great challenge for mining network data. Most representation learning algorithms just consider network structure. However, network vertices contain rich text information, which can be incorporated with NRL through matrix factorization so as to achieve high classification accuracy [92]. Another primary application of graph analysis task is graph classification, which is assigning graphs to several categories, such as classifying proteins based on biological function [7], [66], [127].

In order to present the general process of node classification, and give comparisons of performance with some

TABLE 3. Node Classification of Wiki Dataset by Different Network Embedding Algorithms.

Methods	Time	Micro-F1	Macro-F1
LE [89]	1.13s	0.36	0.15
Node2vec [5]	43.34s	0.66	0.53
LINE [6]	342s	0.39	0.28
SDNE [67]	2271s	0.63	0.51
HOPE [128]	1.62s	0.60	0.43

representative methods, we briefly conduct a node classification on wiki dataset.[1] The dataset is a webpage network consisting of 2,405 nodes, 17,981 links, and 20 classes. We utilize an open-source toolkit,[2] which packs common methods and provides flexible parameters to control algorithms. Here, we evaluate performance with metrics of time consumption, Micro-F1 score, and Macro-F1 score. The ratio of training is 0.5. The results are shown in Table 3. As shown in the table, network embedding can be used to node classification. Different methods have different performance in terms of computational efficiency and embedding accuracy.

B. LINK PREDICTION

Another popular application of node embedding is link prediction by finding explicit or implicit links between nodes in graphs, for example, mining the relationship links in social networks [6], [76], [128], [129]. Link prediction is widely used to predict unknown interactions between nodes to observe links and properties. In addition, it can be used to recommend items via establishing links between users, such as predicting affinities between users and movies [130]. Moreover, node embedding transforms a node into real-value vector, and similar node vectors tend to stay close in the latent learning space. This intrinsic characteristic helps a lot in the prediction of missing edges [5], [67] as close nodes are likely to have connections in the future. Recently, knowledge graph is becoming a hot topic in the network science domain, in which predicting missing relations of entities attracts lots of attention [131]. Predicting unknown interactions between proteins in computational biology is a fundamental problem, which can be treated as the link prediction problem in graphs.

C. CLUSTERING

Clustering is a traditional problem of machine learning. Graph clustering refers to nodes or graphs having similar affiliations or interests which are densely grouped together in a cluster. There are numerous applications of node clustering for community detection [4], text categorization [74], [79], computational biology, and recommender systems [132], etc. Because nodes after embedding process are real-valued vectors, density-based clustering is able to leverage the vector to perform node clustering tasks in graphs [133], [134].

Furthermore, graph clustering is a powerful tool for chemical analysis. For example, it can be used to divide certain wines based on its chemical analysis information from different categories [135]. Similar to classification, clustering also can be utilized to evaluate the performance of representation learning. In NRL algorithms [6], the empirical experiment showed that it has achieved high performance compared with the methods by using the metric of matching authors to the belonging communities.

D. GRAPH VISUALIZATION

Graph visualization is a great way to help human understand and analyze sophisticated networks. The basic form of graph visualization is to project high dimensional network data into a 2D picture, where the same group nodes have the same color, and different node categories can be easily distinguished. For example, LINE [6] is able to visualize the same group authors in the same field, and the data come from co-authorship networks. There are several benefits of graph visualization. When a graph is visualized as a 2D image, it will be easy to reveal the real intrinsic structure of graphs, such as discovering the hidden structure or finding communities. In real applications, graph visualization has many varieties of applications through social science [136], [137] and biology visualization [66]. Furthermore, similar nodes stay close to each other in the 2D visualization figure as well as node clustering. Several researchers utilized graph visualization to present document categories in the visual form [67], [124]. Because node embedding associates with real-value vector, it often connects with dimension-reduced methods, such as PCA and t-SNE [138], or other traditional methods.

We visualize 20Newsgroups dataset[3] using different NRL methods, i.e., LE [89], LLE [16], LINE [6], SDNE [67], and HOPE [128]. Different colors of nodes in pictures represent different classes. The results are shown in Fig 9. We can see that nodes with the same colors are clustering together. That means network embedding could keep the original structural information of the graph.

E. OTHER APPLICATIONS

Besides the general applications mentioned above, there are still a number of specific applications. Herein, we briefly give some examples in the following.

GGS-NNs [78] is a feature learning technique for graph-structured data. The novel characteristic of the algorithm is that it can produce a sequence of outputs rather than a single output. Different from the mentioned applications, this algorithm can be applied for BABI TASK, which contains 20 testing basic forms of reasoning tasks, such as deduction, induction, counting, and path-finding, etc. Duvenaud *et al.* [114] extended the application of NRL to predict the properties of new molecules. The algorithm is proposed based on circular fingerprints and has a better predictive

---

[1]https://linqs.soe.ucsc.edu/data
[2]https://github.com/thunlp/OpenNE

[3]http://qwone.com/ jason/20Newsgroups/
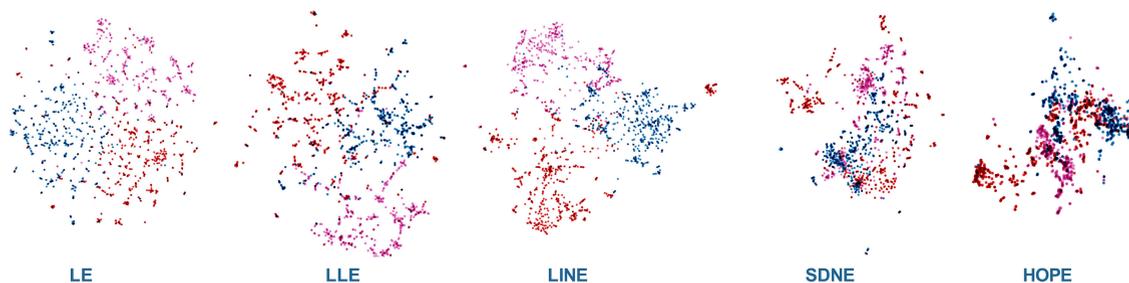
FIGURE 9. Graph visualization of 20Newsgroups dataset.

performance on a series of tasks. Predicting protein function is an interesting topic of bioinformatics, such as Ohm-Net [115] leveraging multi-layers tissue networks to predict multicellular function, which achieved remarkable prediction accuracy. NRL is also an important way to detect community. For example, Li *et al.* [139] proposed a novel embedding based method for community detection leveraging both attributes and structure information of graphs, Tu *et al.* [140] proposed unified framework for community detection considering NRL and text modeling.

VI. OPEN ISSUES
Even though NRL is a powerful and general technique for graph analysis. There still remain lots of concrete open research problems.

A. LARGE-SCALE GRAPHS MODELING
According to our knowledge, few NRL algorithms can better handle large-scale networks, and most of them are just suitable for small-scale networks rather than that having hundreds of millions of nodes and links. With large-scale networks, researchers usually care about computational efficiency and how to combine heterogeneous network structure and multi-type node information. In addition, with the scale of graph increasing, the reconstruction vectors of graphs are becoming vague or inaccurate. That is a big problem when dealing with classification of large-scale graphs, and we believe that how to deal with large-scale network embedding still needs further exploration within the domain of NRL in the future.

B. MODEL DEPTH
Deep learning has achieved a great performance improvement for image classification, handwriting recognition, etc., and has been widely applied for network representation learning in recent years, such as Graph Neural Networks, Graph Convolutional Networks, and Graph Autoencoders etc. Although there are many deep learning-based methods proposed, most of them are shallow models. Too many layers could cause an over-smooth problem and could not fully extract features of graphs. We still lack efficient network embedding approaches to cope with these problems. There are some NRL works

adopting ideas from deep learning models, such as DenseNet, graph convolution of different scales. Deep graph learning is still an open issue for researchers to further study.

C. INTERPRETABILITY
We know that many deep learning models behave as black-boxes, which causes the problem of interpretability. However, lots of graph learning methods derived from deep learning methods need interpretability. For example, graph learning methods are for recommend system or decision-making system. There are few researchers to address this problem with graph learning. So, interpretability is crucial and even more challenging as complex characteristics of graph data.

D. ROBUSTNESS TRAINING AND ADVERSARIAL ATTACKS
Most NRL algorithms rely on ideal graph-structured data. However, in most cases, data are often truncated, missing, fuzzy, lopsided and we cannot get ideal information. Few papers are discussing how to well handle robustness with deep learning-based NRL methods. Even though some works involve GAN mechanism to enhance the robustness of embedding, these algorithms are inefficient and only suitable for specific networks. In addition, deep learning model is sensitive to adversarial attacks. So, these deep learning-based NRL methods are inherently unable to overcome the attack problem. In summary, the robust graph learning techniques still need to be further explored.

VII. CONCLUSION
In this article, we review the NRL algorithms including TFL models and deep learning-based models. These NRL algorithms can learn to reconstruct the representation of graphs. We first give a brief introduction about TFL and then separately discuss the NRL algorithms by focusing on the graph sources (edges and node attributes). There are too many categories of NRL, and we mainly pay attention to these deep learning-based models. We propose three taxonomies of graphs embedding from the deep learning perspective as shown in Table 2. The most important contributions are the parts that introduce NRL algorithms. Also, we summarize the application of graphs embedding in the aspects of classification and semi-supervised learning, link prediction, clustering,

etc. Finally, we emphasize that NRL has great promising future research directions in the field of network science.

REFERENCES

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. ICLR Workshop Papers*, 2013, pp. 1–12.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[3] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. Int. Workshop Artif. Intell. Statist.*, 2005, pp. 246–252.

[4] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, Feb. 2010.

[5] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 855–864.

[6] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[7] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[8] F. Xia, J. Wang, X. Kong, D. Zhang, and Z. Wang, "Ranking station importance with human mobility patterns using subway network datasets," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 2840–2852, Jul. 2020.

[9] W. Wang, F. Xia, H. Nie, Z. Chen, Z. Gong, X. Kong, and W. Wei, "Vehicle trajectory clustering based on dynamic representation learning of Internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–10, 2020.

[10] S. Yu, F. Xia, K. Zhang, Z. Ning, J. Zhong, and C. Liu, "Team recognition in big scholarly data: Exploring collaboration intensity," in *Proc. IEEE 15th Int. Conf Dependable, Autonomic Secure Comput., 15th Int. Conf Pervas. Intell. Comput., 3rd Int. Conf Big Data Intell. Comput. Cyber Sci. Technol. Congress(DASC/PiCom/DataCom/CyberSciTech)*, Nov. 2017, pp. 925–932.

[11] D. Zhang, T. Guo, H. Pan, J. Hou, Z. Feng, L. Yang, H. Lin, and F. Xia, "Judging a book by its cover: The effect of facial perception on centrality in social networks," in *Proc. World Wide Web Conf.*, 2019, pp. 2290–2300.

[12] X. Kong, Y. Shi, S. Yu, J. Liu, and F. Xia, "Academic social networks: Modeling, analysis, mining and applications," *J. Netw. Comput. Appl.*, vol. 132, pp. 86–103, Apr. 2019.

[13] J. Xu, S. Yu, K. Sun, J. Ren, I. Lee, S. Pan, and F. Xia, "Multivariate relations aggregation learning in social networks," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries*, Aug. 2020, pp. 77–86.

[14] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[15] J. B. Tenenbaum, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[16] S. T. Roweis, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[19] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[20] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1045–1048.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[23] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 119–128.

[24] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2847–2856.

[25] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang, "Learning deep network representations with adversarially regularized autoencoders," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2663–2671.

[26] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proc. Int. Conf. Neural Inf. Process*. Cham, Switzerland: Springer, 2018, pp. 362–373.

[27] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 695–704.

[28] M. R. F. MendonÇa, A. Ziviani, and A. M. S. Barreto, "Graph-based skill acquisition for reinforcement learning," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–26, Feb. 2019.

[29] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bulletin.*, vol. 40, no. 3, pp. 52–74, 2017.

[30] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl.-Based Syst.*, vol. 151, pp. 78–94, Jul. 2018.

[31] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 3–28, Mar. 2020.

[32] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2019.

[33] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 24, 2020, doi: 10.1109/TNNLS.2020.2978386.

[34] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 17, 2020, doi: 10.1109/TKDE.2020.2981333.

[35] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Netw.*, vol. 129, pp. 203–221, Sep. 2020.

[36] W. Cao, Z. Yan, Z. He, and Z. He, "A comprehensive survey on geometric deep learning," *IEEE Access*, vol. 8, pp. 35929–35949, 2020.

[37] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond," 2020, *arXiv:2004.00216*. [Online]. Available: http://arxiv.org/abs/2004.00216

[38] Y. Xie, C. Li, B. Yu, C. Zhang, and Z. Tang, "A survey on dynamic network embedding," 2020, *arXiv:2006.08093*. [Online]. Available: http://arxiv.org/abs/2006.08093

[39] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[40] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *London, Edinburgh, Dublin Phil. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, Nov. 1901.

[41] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: Spectral clustering and normalized cuts," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 551–556.

[42] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *J. Comput. Graph. Statist.*, vol. 15, no. 2, pp. 265–286, Jun. 2006.

[43] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, p. 11, May 2011.

[44] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Robust principal component analysis on graphs," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2812–2820.

[45] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 585–591.

[46] C.-M. Feng, Y.-L. Gao, J.-X. Liu, J. Wang, D.-Q. Wang, and C.-G. Wen, "Joint $L_{1/2}$-norm constraint and graph-Laplacian PCA method for feature extraction," *BioMed Res. Int.*, vol. 2017, pp. 1–14, 2017.

[47] C.-M. Feng, Y.-L. Gao, J.-X. Liu, C.-H. Zheng, and J. Yu, "PCA based on graph Laplacian regularization and P-Norm for gene selection and clustering," *IEEE Trans. Nanobiosci.*, vol. 16, no. 4, pp. 257–265, Jun. 2017.

[48] D. Sun, H. Liang, M. Ge, Z. Ding, W. Cai, and B. Luo, "Protein functional annotation refinement based on graph regularized ℓ_1-norm PCA," *Pattern Recognit. Lett.*, vol. 87, pp. 212–221, Feb. 2017.

[49] N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst, "Fast robust PCA on graphs," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 4, pp. 740–756, Jun. 2016.

[50] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, Nov. 2014.

[51] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[52] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009.

[53] M. Asteris, A. Kyrillidis, A. Dimakis, H.-G. Yi, and B. Chandrasekaran, "Stay on path: PCA along graph paths," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1728–1736.

[54] W. Min, J. Liu, and S. Zhang, "Edge-group sparse PCA for network-guided high dimensional data analysis," *Bioinformatics*, vol. 34, no. 20, pp. 3479–3487, Oct. 2018.

[55] T.-W. Lee, "Independent component analysis," in *Independent Component Analysis*. Cham, Switzerland: Springer, 1998, pp. 27–66.

[56] B. Park, D.-S. Kim, and H.-J. Park, "Graph independent component analysis reveals repertoires of intrinsic network components in the human brain," *PLoS ONE*, vol. 9, no. 1, Jan. 2014, Art. no. e82873.

[57] D. López-Barroso, P. Ripollés, J. Marco-Pallarés, B. Mohammadi, T. F. Münte, A.-C. Bachoud-Lévi, A. Rodriguez-Fornells, and R. de Diego-Balaguer, "Multiple brain networks underpinning word learning from fluent speech revealed by independent component analysis," *NeuroImage*, vol. 110, pp. 182–193, Apr. 2015.

[58] D. R. de Paula, E. Ziegler, P. M. Abeyasinghe, T. K. Das, C. Cavaliere, M. Aiello, L. Heine, C. di Perri, A. Demertzi, Q. Noirhomme, V. Charland-Verville, A. Vanhaudenhuyse, J. Stender, F. Gomez, J.-F. L. Tshibanda, S. Laureys, A. M. Owen, and A. Soddu, "A method for independent component graph analysis of resting-state fMRI," *Brain Behav.*, vol. 7, no. 3, 2017, Art. no. e00626.

[59] M. E. J. Newman, "Spectral methods for community detection and graph partitioning," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 88, no. 4, Oct. 2013, Art. no. 042822.

[60] Y. Zhang, E. Levina, and J. Zhu, "Detecting overlapping communities in networks using spectral methods," 2014, *arXiv:1412.3432*. [Online]. Available: http://arxiv.org/abs/1412.3432

[61] Y. Li, F. Nie, H. Huang, and J. Huang, "Large-scale multi-view spectral clustering via bipartite graph," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2750–2756.

[62] D. Pfau, S. Petersen, A. Agarwal, D. G. T. Barrett, and K. L. Stachenfeld, "Spectral inference networks: Unifying deep and spectral learning," 2018, *arXiv:1806.02215*. [Online]. Available: http://arxiv.org/abs/1806.02215

[63] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2071–2080.

[64] T. Trouillon, C. R. Dance, E. Gaussier, J. Welbl, S. Riedel, and G. Bouchard, "Knowledge graph completion via complex tensor factorization," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 4735–4772, 2017.

[65] I. Borg and P. Groenen, "Modern multidimensional scaling: Theory and applications," *J. Educ. Meas.*, vol. 40, no. 3, pp. 277–280, Sep. 2003.

[66] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2014–2023.

[67] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1225–1234.

[68] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[69] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1269–1278.

[70] J. Liu, F. Xia, L. Wang, B. Xu, X. Kong, H. Tong, and I. King, "Shifu2: A network representation learning based model for advisor-advisee relationship mining," *IEEE Trans. Knowl. Data Eng.*, early access, Oct. 11, 2019, doi: 10.1109/TKDE.2019.2946825.

[71] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.

[72] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[73] H. Hong, X. Li, and M. Wang, "GANE: A generative adversarial network embedding," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2325–2335, Jul. 2020.

[74] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1145–1152.

[75] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, vol. 48, 2016, pp. 40–48.

[76] C. Tu, Z. Zhang, Z. Liu, and M. Sun, "TransNET: Translation-based network representation learning for social relation extraction," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, Melbourne, VIC, Australia, 2017, pp. 2864–2870.

[77] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2609–2615.

[78] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–20.

[79] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[80] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4800–4810.

[81] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, and C. Wang, "Hierarchical graph pooling with structure learning," 2019, *arXiv:1911.05954*. [Online]. Available: http://arxiv.org/abs/1911.05954

[82] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," 2019, *arXiv:1904.08082*. [Online]. Available: http://arxiv.org/abs/1904.08082

[83] F. Xia, J. Liu, H. Nie, Y. Fu, L. Wan, and X. Kong, "Random walks: A review of algorithms and applications," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 2, pp. 95–107, Apr. 2020.

[84] Y. Goldberg and O. Levy, "Word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method," 2014, *arXiv:1402.3722*. [Online]. Available: http://arxiv.org/abs/1402.3722

[85] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2177–2185.

[86] X. Rong, "Word2vec parameter learning explained," 2014, *arXiv:1411.2738*. [Online]. Available: http://arxiv.org/abs/1411.2738

[87] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 459–467.

[88] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, Jul. 2009.

[89] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.

[90] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[91] F. Feng, X. He, J. Tang, and T.-S. Chua, "Graph adversarial training: Dynamically regularizing based on graph structure," *IEEE Trans. Knowl. Data Eng.*, early access, Dec. 5, 2019, doi: 10.1109/TKDE.2019.2957786.

[92] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2015, pp. 2111–2117.

[93] Y. Dong, N. V. Chawla, and A. Swami, "Metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 135–144.

[94] X. Wang, H. Ji, C. Shi, B. Wang, P. Cui, P. Yu, and Y. Ye, "Heterogeneous graph attention network," 2019, *arXiv:1903.07293*. [Online]. Available: http://arxiv.org/abs/1903.07293

[95] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 793–803.

[96] X. Wang, Y. Lu, C. Shi, R. Wang, P. Cui, and S. Mou, "Dynamic heterogeneous information network embedding with meta-path based proximity," *IEEE Trans. Knowl. Data Eng.*, early access, May 11, 2020, doi: 10.1109/TKDE.2020.2993870.

[97] Z. Li, W. Zheng, X. Lin, Z. Zhao, Z. Wang, Y. Wang, X. Jian, L. Chen, Q. Yan, and T. Mao, "TransN: Heterogeneous network representation learning by translating node embeddings," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 589–600.

[98] C. Shi, Y. Lu, L. Hu, Z. Liu, and H. Ma, "RHINE: Relation structure-aware heterogeneous information network embedding," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 30, 2020, doi: 10.1109/TKDE.2020.2982898.

[99] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–14.

[100] B. L. Douglas, "The Weisfeiler-Lehman method and graph isomorphism testing," 2011, *arXiv:1101.5211*. [Online]. Available: http://arxiv.org/abs/1101.5211

[101] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 912–919.

[102] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.

[103] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," 2017, *arXiv:1710.10568*. [Online]. Available: http://arxiv.org/abs/1710.10568

[104] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," 2018, *arXiv:1801.10247*. [Online]. Available: http://arxiv.org/abs/1801.10247

[105] T. Derr, Y. Ma, and J. Tang, "Signed graph convolutional networks," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 929–934.

[106] Y. Ma, S. Wang, C. C. Aggarwal, D. Yin, and J. Tang, "Multi-dimensional graph convolutional networks," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2019, pp. 657–665.

[107] G. Li, M. Müller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?" 2019, *arXiv:1904.03751*. [Online]. Available: http://arxiv.org/abs/1904.03751

[108] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.* Cham, Switzerland: Springer, 2018, pp. 593–607.

[109] L. Cai, B. Yan, G. Mai, K. Janowicz, and R. Zhu, "TransGCN: Coupling transformation assumptions with graph convolutional networks for link prediction," in *Proc. 10th Int. Conf. Knowl. Capture*, Sep. 2019, pp. 131–138.

[110] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.

[111] P. Wang, J. Han, C. Li, and R. Pan, "Logic attention based neighborhood aggregation for inductive knowledge graph embedding," in *Proc. AAAI*, vol. 33, 2019, pp. 7152–7159.

[112] B. Hu, Y. Fang, and C. Shi, "Adversarial learning on heterogeneous information networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 120–129.

[113] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2434–2444.

[114] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.

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

[116] R. Milo, "Network motifs: Simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, Oct. 2002.

[117] L. Wang, J. Ren, B. Xu, J. Li, W. Luo, and F. Xia, "Model: Motif based network embedding for link prediction," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 2, pp. 503–516, 2020.

[118] S. V. N. Vishwanathan, N. N. Schraudolph, I. R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, Apr. 2010.

[119] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[120] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Comput. Sci.*, 2014, pp. 1–15.

[121] F. R. Chung and F. C. Graham, *Spectral Graph Theory*, no. 92. Providence, RI, USA: AMS, 1997.

[122] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[123] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[124] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1895–1901.

[125] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 731–739.

[126] T. D. Bui, S. Ravi, and V. Ramavajjala, "Neural graph machines: Learning neural networks using graphs," 2017, *arXiv:1703.04818*. [Online]. Available: http://arxiv.org/abs/1703.04818

[127] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1365–1374.

[128] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1105–1114.

[129] W. Wang, J. Liu, F. Xia, I. King, and H. Tong, "Shifu: Deep learning based advisor-advisee relationship mining in scholarly big data," in *Proc. 26th Int. Conf. World Wide Web Companion*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 303–310.

[130] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2017, *arXiv:1706.02263*. [Online]. Available: http://arxiv.org/abs/1706.02263

[131] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proc. IEEE*, vol. 104, no. 1, pp. 11–33, Jan. 2016.

[132] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu, "Heterogeneous information network embedding for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, Feb. 2019.

[133] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 891–900.

[134] S. Wang, J. Tang, F. Morstatter, and H. Liu, "Paired restricted Boltzmann machine for linked data," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 1753–1762.

[135] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1293–1299.

[136] B. Paul Chamberlain, J. Clough, and M. Peter Deisenroth, "Neural embeddings of graphs in hyperbolic space," 2017, *arXiv:1705.10359*. [Online]. Available: http://arxiv.org/abs/1705.10359

[137] L. Freeman, "Visualizing social networks," *Social Netw. Data Analytics*, vol. 6, no. 4, pp. 411–429, 2000.

[138] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 2605, pp. 2579–2605, 2008.

[139] Y. Li, C. Sha, X. Huang, and Y. Zhang, "Community detection in attributed graphs: An embedding approach," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 338–345.

[140] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin, "A unified framework for community detection and network representation learning," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 6, pp. 1051–1065, Jun. 2019.

KE SUN received the B.Sc. and M.Sc. degrees from Shandong Normal University, Jinan, China. He is currently pursuing the Ph.D. degree in software engineering with the Dalian University of Technology, Dalian, China. His research interests include deep learning, network representation learning, and knowledge graph.

LEI WANG received the B.Sc. degree in software engineering from the Dalian University of Technology, China, in 2018, where he is currently pursuing the master's degree with the School of Software. His research interests include data mining, the analysis of complex networks, and machine learning.

BO XU (Member, IEEE) received the B.Sc. and Ph.D. degrees from the Dalian University of Technology, China, in 2007 and 2014, respectively. She is currently an Associate Professor with the School of Software, Dalian University of Technology. Her current research interests include data science, network analysis, and natural language processing.

WENHONG ZHAO (Member, IEEE) received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2002. Since 1991, he has been with the Zhejiang University of Technology, Hangzhou, where he is currently a Full Professor with the Ultraprecison Machining Center. His research interests include big data, embedded systems, intelligent systems, and precision machining.

SHYH WEI TENG (Member, IEEE) received the Ph.D. degree from Monash University, in 2004. He held positions at Monash University. He is currently an Associate Professor and the Deputy Dean of the School of Engineering, IT and Physical Sciences, Federation University Australia. He is also the Residential Aged Care Lead of his University's partnership in the Australia's Digital Health Cooperative Research Centre and also a Selected Member of the City of Casey Smart City Advisory Committee. He has published more than 80 refereed research articles. His research interests include image/video processing, machine learning, and multimedia analytics.

FENG XIA (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees from Zhejiang University, Hangzhou, China. He is currently an Associate Professor and a Discipline Leader with the School of Engineering, IT and Physical Sciences, Federation University Australia. He has published two books and more than 300 scientific articles in international journals and conferences. His research interests include data science, social computing, and systems engineering. He is a Senior Member of ACM.

• • •