Graph Neural Networks: Foundations, State-of-the-Art, Applications for Data Management Research

Wei Zhuo
Sun Yat-sen University
Shenzhen, China
zhuow5@mail2.sysu.edu.cn

Zhengtong Yan
University of Helsinki
Helsinki, Finland
zhengtong.yan@helsinki.fi

Guang Tan
Sun Yat-sen University
Shenzhen, China
tanguang@mail.sysu.edu.cn

Jiaheng Lu
University of Helsinki
Helsinki, Finland
jiaheng.lu@helsinki.fi

ABSTRACT

Recent years have witnessed rapid advances in Graph Neural Networks (GNNs) that have benefited myriads of graph analytical tasks ranging from node classification to community detection. In particular, GNNs generalize deep learning models (e.g., CNN and RNN) to graph data to map graph nodes into a low-dimensional vector space, such that complicated graph analysis can be conducted based on the learned representation vectors. Thus, GNNs inherit the representation power of deep neural networks and leverage the structure information of graph data simultaneously, which significantly helps them in capturing the complex information of graph data. However, compared to GNN applications from other domains, database management systems offer many unique problems. Thus, GNN4DB has received increasing attention, which motivates us to present a comprehensive tutorial on this popular yet challenging topic.

Based on the techniques employed, GNNs have generally fallen into two main categories. The first is Spectral GNNs which are motivated by spectral graph theory and graph signal processing; the second is Spatial GNNs which borrow the idea from standard CNNs to directly perform neighborhood aggregation on graph structure. In this tutorial, we will introduce the theoretical foundations of these two GNN categories and state-of-the-art GNN methods on various applications in detail. Besides, we will highlight the applications of GNNs in database management systems. Finally, we will discuss some current issues and future directions.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Information systems** → **Data management systems**.

1 INTRODUCTION

Graphs, as a ubiquitous data structure in computer science and related fields, naturally represent complex interactions between elements of composite objects. Many real-world data come in the form of graphs, such as social networks [6, 37, 49], citation networks [38, 58], molecular graphs [35, 52], point clouds [39, 46], and transportation networks [3]. The aforementioned abundance of graph structure data raises requirements for intelligent methods that can automatically mine and analyze information from graphs and gain valuable insights.

Representation learning as an important aspect of machine learning has been extended to graph structure data to improve the efficiency and effectiveness of graph-based analytical tasks, such as node classification, link prediction [17, 33, 40], and community detection [45]. Specifically, graph representation learning (GRL) is to learn a mapping that projects nodes, edges, subgraphs, or graphs to a low-dimensional continuous space while topological relationships and inherent properties can be well preserved, and the dimension of embedding space is normally far smaller than the number of nodes. Fig. 1 shows an intuitive example of GRL. In the last decade, network embedding (NE) [8, 16, 19] and graph neural networks (GNNs) [51, 61, 63] have been two mainstream approaches for graph representation learning. NE aims at representing graphs by preserving the pre-defined structural proximity measure between nodes, where the proximity measure can be the co-occurrence probability in random walk sequences [13, 17, 33, 40], communities [34, 45], or metapaths [11]. However, such a learning paradigm suffers from two inherent drawbacks. Firstly, NE methods construct supervision signals only from the graph structure, and

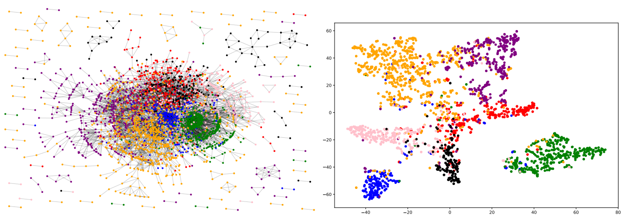
(a) Input: Cora graph. **(b) Output: representations in the embedding space $\mathbb{R}^2$.**

Figure 1: An example of graph representation learning on Cora graph. Representations are learned by a GRL method PE-GCL [67].

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: State-of-the-art solutions for classic graph analytical tasks.

Task	State-of-the-Art	Type
Node Classification	GLEM [62]	GNN-based
Graph Classification	Graphormer [59]	GNN-based
Link Prediction	GIDN [47]	GNN-based
Community Detection	CLARE [50]	GNN-based

thus they are task-independent and insensitive to attributes of the graph. Secondly, NE methods are also regarded as "shallow" models since they directly treat node representations as learnable parameters and the encoder function is just a simple embedding looking-up, without multilayer weight matrices to capture high-order nonlinear graph structures.

Recent development in Deep Neural Networks (DNNs), such as Convolutional Neural Networks (CNN) [32], Recurrent Neural Networks (RNN) [36], and Transformer [42], have made significant progress towards modeling for complex domains, such as computer vision and natural language processing. Inspired by them, Graph Neural Networks (GNNs) generalize DNN approaches to graph structure data, inheriting the advantages of DNNs. An important aspect is that GNNs follow the spirit of parameter sharing in DNNs, i.e., the learnable parameter matrices of GNNs are shared across all nodes, which are analogous to the filters in CNNs. Hence it allows stack multi-layer parameter matrices with nonlinearity to extract high-level node representations. On the other hand, GNNs can be trained in an end-to-end fashion and thus the learned representations are task-related. Besides, in the GNN framework, the representation of a node is computed by a GNN layer aggregating information from the node's topological neighbors via non-linear transformation and aggregation functions, which naturally capture the interactions between local structures and neighborhood features for the target node. Hence, GNNs are capable of handling attributed graphs. So the key limitations of NE can be well resolved by GNNs. Since the semi-supervised Graph Convolution Networks (GCNs) were first proposed by [21], GNNs have been the most popular GRL methods and outperform NE-based methods in most classic graph analytical tasks, as shown in Table 1. In this tutorial, we will give a brief introduction to NE and a comprehensive review of GNNs. Particularly, besides the traditional graph-based tasks, we also feature the applications of GNNs for Database Management Systems.

Related tutorials. Several tutorials with a similar topic have been presented at recent conferences. However, those tutorials either solely focus on GNNs or general machine learning for databases [25–27, 56, 57]. Our tutorial can be considered as an extension of the following tutorials: (1) Graph Neural Networks: Foundation, Frontiers, and Applications (IJCAI 2022 and KDD 2022); (2) Graph Neural Networks: Models and Applications (SIGIR 2021); (3) Representation Learning on Networks (WWW 2018); (4) AI Meets Database: AI4DB and DB4AI (SIGMOD 2021); (5) Machine Learning for Databases (VLDB 2021 and AIMLs 2021). Different from previous tutorials focusing on general representation learning or GNNs or general machine learning for databases, our tutorial has a strong focus on the most recent techniques in GNNs for database management.

Overview of this tutorial. This tutorial on Graph Neural Networks will cover a broad range of topics in GNNs. Specifically, we will introduce the fundamental concepts and advanced algorithms of GNNs, new research frontiers of GNNs, broad and emerging applications with GNNs, and current open challenges of GNNs. In addition, we emphasize the applications of GNNs in Database Management. To the best of our knowledge, this is the first tutorial to discuss the GNNs in Database Management and related conferences.

2 GRAPH NEURAL NETWORKS

It is well-known that GNNs have been one of the most popular technologies in deep learning. Its widespread use makes the division of the current GNNs can follow different standards. For example, GNNs can be classified based on different downstream tasks (e.g., node classification, link prediction, recommendation, etc.); they can also be classified into semi-supervised GNNs and unsupervised GNNs according to whether external labels are involved in training. In this tutorial, we broadly divide GNNs into Spectral GNNs and Spatial GNNs based on their theoretical frameworks.

2.1 Preliminaries

In this tutorial, we consider attributed graphs represented as $G = \left(V, E, \mathbf{X}^V, \mathbf{X}^E\right)$ where $V = \{v_1, ..., v_N\}$ is a set of $N = |V|$ nodes and $E \subseteq V \times V$ is a set of $M = |E|$ edges between nodes. $\mathbf{X}^V \in \mathbb{R}^{N \times F_v}$ and $\mathbf{X}^E \in \mathbb{R}^{M \times F_e}$ denote the feature matrix for N nodes and M edges respectively, which characterize the attributes of the graph. V and E jointly form the adjacency matrix of G denoting as $\mathbf{A} \in \mathbb{R}^{N \times N}$ with (i, j) entry equaling to 1 if there is an edge between nodes v_i and v_j and 0 otherwise. For an undirected graph, it has a graph Laplacian matrix which is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{N \times N}$ where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ is a diagonal degree matrix. Let $\mathbf{I}$ denote the identity matrix, the symmetric normalized graph Laplacian is often used in GNNs, which is defined as $\tilde{\mathbf{L}}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. Another version is random walk normalization: $\tilde{\mathbf{L}} = \mathbf{D}^{-1}\mathbf{L}$. Based on graph signal processing, $\mathbf{x} \in \mathbb{R}^N$ is a graph signal that corresponds to one dimension of $\mathbf{X}^V$. $\mathbf{L}$ can be eigendecomposed as $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$ where each column of $\mathbf{U} \in \mathbb{R}^{N \times N}$ is an eigenvector of $\mathbf{L}$ and $\Lambda = \mathrm{diag}([\lambda_1, \cdots, \lambda_N])$ is a diagonal matrix whose diagonal elements are eigenvalues of $\mathbf{L}$.

2.2 Spectral Graph Neural Networks

Spectral GNNs rely on spectral graph theory [7] and graph signal processing [31]. The basic idea of this framework is that it first assumes the graph is undirected and then leverages graph Fourier transform to convert graph signals defined in spatial domain into spectral domain. The spectral domain can be the space spanned by the eigenvectors of the graph Laplacian in practice because it carries the smooth geometry of the graph [2]. By transposing the convolution theorem to graphs, the graph convolution is defined as the multiplication of the graph signals and the parameterized convolution filter in the spectral domain, such that the parameter sharing property of traditional CNN can be satisfied.

Spectral Graph CNN [2] is the first work of Spectral GNNs. Given a graph signal $\mathbf{x}$ and the graph Laplacian $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$, we have Fourier transform $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$ and inverse Fourier transform $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. Taking the other signal $\boldsymbol{\omega} \in \mathbb{R}^N$ as a parameterized filter, the graph convolution of $\mathbf{x}$ with $\boldsymbol{\omega}$ is

$$\mathbf{x} * \boldsymbol{\omega} = \mathbf{U}\left(\left(\mathbf{U}^\top\mathbf{x}\right) \odot \left(\mathbf{U}^\top\boldsymbol{\omega}\right)\right) = \mathbf{U}g_\theta\mathbf{U}^\top\mathbf{x}, \tag{1}$$

where $\odot$ is Hadamard product and $\mathbf{U}^\top\boldsymbol{\omega} = [\theta_1, \ldots, \theta_n]^T$ is a parameterized vector in spectral domain. Hence, $g_\theta = \text{diag}([\theta_1, \ldots, \theta_n])$ is a diagonal matrix as spectral filter. By stacking multiple convolution layers, the l-th layer Spectral Graph CNN is:

$$\mathbf{H}_j^{(l+1)} = \sigma\left(\sum_{i=1}^{d_l} \mathbf{U}\boldsymbol{\Omega}_i^{(l,j)}\mathbf{U}^\top\mathbf{H}_i^{(l)}\right), \quad \text{for } j \in \{1, \ldots, d_{l+1}\}, \tag{2}$$

where d_l is the number of signals (feature dimension) at the l-th layer and $d_0 = F_v$, and d_{l+1} is the number of filters (output dimension) at the l-th layer. $\mathbf{H}_i^{(l)}$ is the i-th input signal at the l-th layer, $\boldsymbol{\Omega}_i^{(l,j)}$ is the j-th filter for the i-th signal at the l-th layer, and σ denotes a non-linear activation function e.g., ReLU. Such a layer of Spectral Graph CNN defined in Eq. (2) transforms the dimension of node feature from d_l to d_{l+1}. While spectral graph theory lends theoretical support to Spectral Graph CNN, this approach suffers from several key bottlenecks. Firstly, since the graph Fourier transform needs to compute the full eigendecompositions of the Laplacian matrix and the parameter size is linearly related to the number of nodes, the time complexity and parameter complexity of the algorithm is high; secondly, the convolution operation defined in the spectral domain is not satisfied with spatial localization; finally, the graph filter depends on the bases of the Fourier transform of the specific graph, so it is hard to parallelly handle multiple graphs of different sizes.

ChebNet [10]. Note that the parameterized spectral filter g_θ is not localized and its learning complexity is $O(N)$. These two limitations contrast with traditional CNNs where convolution filters are localized in space and the learning complexity is independent of the input shape. To tackle these issues, ChebNet first proposes to use polynomials of the diagonal matrix of eigenvalues to approximate g_θ with $g_\theta(\Lambda)$, where $g_\theta(\Lambda)_{ii} = \sum_{j=0}^{L}\theta_j\lambda_i^j$ and $L \ll N$. Then we have the graph convolution:

$$g_\theta(\Lambda) * \mathbf{x} = \mathbf{U}g_\theta(\Lambda)\mathbf{U}^\top\mathbf{x} = \sum_{j=0}^{L}\theta_j\mathbf{L}^j\mathbf{x}. \tag{3}$$

By this mean, the number of parameters of the filter can be reduced from $O(N)$ to $O(L)$, and this expression is L-localization since it is a L-order polynomial in the graph Laplacian, which depends only on nodes that are at most L hops away from the central node. However, computing the powers of $\mathbf{L}$ still needs computation complexity $O(N^3)$. To solve this problem, ChebNet further uses a truncated expansion in terms of Chebyshev polynomials $T_k(x)$ up to the L-th order to approximate $g_\theta(\Lambda)$. Eq. (3) can be approximated as follows:

$$g_\theta(\Lambda) * \mathbf{x} \approx \sum_{j=0}^{L}\theta_j T_k(\tilde{\mathbf{L}})\mathbf{x}, \tag{4}$$

where the Chebyshev polynomials are defined recursively by $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$ and $T_1(x) = x$, $\tilde{\mathbf{L}} = 2\tilde{\mathbf{L}}_{sym}/\lambda_{max} - \mathbf{I}$ where λ_{max} is the largest eigenvalue of $\tilde{\mathbf{L}}_{sym}$. Here $\tilde{\mathbf{L}}$ is a scaling of the graph Laplacian to map the eigenvalues from $[0, \lambda_{max}]$ to $[-1, 1]$ since the Chebyshev polynomial forms an orthogonal bases in $[-1, 1]$.

Graph Convolutional Network (GCN) [21]. As shown in Eq. (4), ChebNet uses a single convolution filter to achieve L-localization, i.e., ChebNet can be seen as using one neural network layer to preserve L-localization. GCN proposes a first-order approximation of ChebNet and achieves high-order localization by stacking multiple layers. Specifically, GCN limits the layer-wise convolution filter to $L = 1$ and λ_{max} is approximated by 2 to get a linear function. Then Eq. (4) can be simplified to:

$$g_\theta(\Lambda) * \mathbf{x} = \theta_0\mathbf{x} - \theta_1\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{x}. \tag{5}$$

Since $\{\theta_i\}$ are free parameters, GCN further assumes the parameters $\theta = \theta_0 = -\theta_1$, leading to the following graph convolution:

$$g_\theta(\Lambda) * \mathbf{x} = \theta(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{x}. \tag{6}$$

Then GCN further introduce the renormalization trick which replace $\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ with $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is a diagonal degree matrix of $\tilde{\mathbf{A}}$. Finally, by generalizing the convolution to work with multiple filters in a d channel input and layering the model with nonlinear activation functions between each layer, the l-th layer GCN is defined as:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{7}$$

wherer $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is a layer-specific trainable weight matrix and $\sigma(\cdot)$ denotes an activation function. Hence, we can stack L GCN layers defined in Eq. (7) to achieve L-localization.

Spectral Graph CNN, ChebNet, and GCN are the pioneering works of spectral GNNs, whereas GCN is also a spatial GNN which we will discuss later. These models provide a solid mathematical foundation that motivates the follow-up works to make improvements over them.

GWNN [53] is a more flexible spectral method that replaces the Fourier transform in spectral filters with the graph wavelet transform. GWNN uses a set of wavelets $\boldsymbol{\Psi}_s = \{\psi_{s1}, \psi_{s2}, \cdots, \psi_{sn}\}$ as bases of spectral domain, where each wavelet ψ_{si} represents the energy diffused from node v_i, which describes the local structure of v_i, and s is a scaling parameter. Then Eq. (1) can be rewritten as:

$$\mathbf{x} * \boldsymbol{\omega} = \boldsymbol{\Psi}_s\left(\left(\boldsymbol{\Psi}_s^\top\mathbf{x}\right) \odot \left(\boldsymbol{\Psi}_s^\top\boldsymbol{\omega}\right)\right) = \boldsymbol{\Psi}_s g_\theta\boldsymbol{\Psi}_s^\top\mathbf{x}. \tag{8}$$

Compared with Fourier bases, wavelet bases are localized and the range of localization can be controlled by s. The high spareness of wavelet bases leads to the computational complexity linear with respect to the number of edges.

Graph Diffusion Convolution (GDC) [22] proposes to use a generalized graph diffusion $\mathbf{M} = \sum_{k=0}^{\infty}\theta_k\mathbf{S}^k$ to define spectral polynomial filters, where $\{\theta_k\}$ are weight coefficients and $\mathbf{S}$ is the generalized transition matrix. $\mathbf{M}$ can be regarded as related to the Taylor expansion of matrix-valued functions. Thus, the choice of θ_k and $\mathbf{S}$ must at least ensure that $\mathbf{M}$ converges. ChebNet is an example of GDC, and GDC also provides two special cases as diffusion, i.e., personalized PageRank (PPR) and Heat Kernel.

2.3 Spatial Graph Neural Networks

Analogous to the convolution operation of CNN on image processing, spatial GNNs define graph convolutions based on spatial relations of nodes, rather than in the spectral domain. Spatial GNNs are so-called *Message Passing Neural Networks* (MPNN) [15] which treat graph convolutions as a message passing process in which information can be passed from one node to another along edges directly. Such a GNN begins with the node features as the initial representation $h_i^{(0)}$ of a node v_i, and runs L-layer message passing to let each node aggregate information propagated from its neighborhoods. In the l-th layer, every node v_i computes a new representation from its own current representation and the multiset of representations in its neighborhoods; formally, the general framework of Spatial GNN is defined as:

$$\begin{aligned} a_i^{(l)} &= f_{agg}\left(\left\{\left\{h_j^{(l-1)} \mid v_j \in \mathcal{N}(v_i)\right\}\right\}\right), \\ h_i^{(l)} &= f_{upd}\left(h_i^{(l-1)}, a_i^{(l)}\right) \end{aligned} \tag{9}$$

where $\mathcal{N}(v_i)$ is the neighborhood of v_i, f_{agg} is a permutation-invariant function to aggregate messages for v_i from its neighborhood. $a_i^{(l)}$ is the aggregated message at the l-th layer. f_{upd} is a function to integrate the neighborhood message and the central node, $h_i^{(l)}$ is the representation of v_i at the l-th layer, and $h_i^{(0)} = X_i^V$ is the input node feature of v_i. For node representation, the output of the final layer $h_i^{(L)}$ is considered as an informative representation that could be used for node-level downstream tasks. For graph representation, another READOUT function $f_{readout}$ is employed to obtain the a graph-level representation h_G by integrating the final representations of nodes:

$$h_G = f_{readout}\left(\left\{h_i^{(L)} | v_i \in V\right\}\right) \tag{10}$$

Although GCN is a spectral-based method, Eq. (7) can be rewritten as:

$$\begin{aligned} a_i^{(l)} &= \sum_{j \in \mathcal{N}(v_i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{ii}\tilde{\mathbf{D}}_{jj}}} h_j^{(l-1)} \\ h_i^{(l)} &= \sigma\left(\left(\frac{1}{\tilde{\mathbf{D}}_{ii}} h_i^{(l-1)} + a_i^{(l)}\right)\mathbf{W}^{(l)}\right). \end{aligned} \tag{11}$$

As we can see, GCN can be represented as a weighted sum of the transformed center node representation $h_i^{(l-1)}$ and neighbor node representations $h_j^{(l-1)}$, which is in a message passing form. Thus, GCN is also a spatial GNN that establishes a relationship between spectral methods and spatial methods.

GraphSAGE [18] is another early spatial-based GNN. The propagation matrix in GCN is defined as $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, which relies on the adjacency matrix of the input graph structure. Thus GCN requires storing the entire graph in memory which can be computationally expensive for large graphs. GraphSAGE reduces the computational complexity of GCN by sampling a fixed number of neighbors for each node to remove the strong dependency on a complete graph structure. It performs graph convolutions by:

$$h_i^{(l)} = \sigma\left(\mathbf{W}_1^{(l)} h_i^{(l-1)} + \mathbf{W}_2^{(l)} f_{agg}\left(\left\{h_j^{(l-1)} | v_j \in \text{Sample}(\mathcal{N}(v_i), k)\right\}\right)\right), \tag{12}$$

Table 2: A summary of different GNN models.

Model	Type	Filter	Time Complexity
Spectral Graph CNN [2]	Spectral	Interpolation kernel	$O(N^3)$
ChebNet [10]	Spectral	Polynomial	$O(M)$
GWNN [53]	Spectral	Wavelet transform	$O(M)$
GDC [22]	Spectral	PPR/Heat Kernel	$O(M)$
CayleyNet [24]	Spectral	Polynomial	$O(M)$
AGCN [28]	Spectral	Residual Adjacency Matrix	$O(N^2)$
DualGCN [66]	Spectral	PPMI	$O(M)$
S^2GC [65]	Spectral	Markov Diffusion Kernel	$O(M)$
GCN [21]	Spectral/Spatial	Normalized Adjacency Matrix	$O(M)$
GraphSAGE [18]	Spatial	Sampled Neighborhoods	$O(NkL)$
GAT [43]	Spatial	Attention Weighted Neighbors	$O(M)$
GIN [54]	Spatial	Neighborhoods	$O(M)$
PPNP [14]	Spatial	PPR	$O(M)$
DGI [44]	Spatial	Neighborhoods	$O(M)$
FastGCN [4]	Spatial	Sampled Neighborhoods	$O(NkL)$
JK-Net [55]	Spatial	multi-hop neighborhoods	$O(Nk^L)$

where Sample$(\mathcal{N}(v_i), k)$ denotes randomly sampling k neighbors with replacement for v_i.

GAT [43]. The convolution filters in CNN are learnable matrices to adaptively learn the local pattern for each pixel. But for GCN, the convolution filters are deterministic, i.e., GCN directly uses the normalized adjacency matrix $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ as neighborhood aggregation weights for each node. GAT adopts an attention mechanism to learn the contribution of each neighboring node to the central node. The graph convolution operation according to GAT is defined as:

$$h_i^{(l)} = \sigma\left(\sum_{v_j \in \mathcal{N}(v_i) \cup v_i} \alpha_{ij}^{(l)} \mathbf{W}^{(l)} h_j^{(l-1)}\right), \tag{13}$$

where the attention weight α_{ij} is given by:

$$\alpha_{ij}^{(l)} = \text{softmax}\left(\Gamma\left(\mathbf{a}^T\left[\mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} \| \mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)}\right]\right)\right), \tag{14}$$

where $\Gamma(\cdot)$ is a LeakyReLU activation function and $\mathbf{a}$ is a vector of parameters that transforms concatenated node hidden representations into a scalar weight. GAT further performs the multi-head attention to stabilize the learning process of self-attention, i.e., the operations of the convolution layer defined as Eq. (13) are independently replicated K times, and outputs are feature-wise aggregated by concatenation:

$$h_i^{(l)} = \|_{k=1}^{K} \sigma\left(\sum_{v_j \in \mathcal{N}(v_i) \cup v_i} \alpha_{ij,k}^{(l)} \mathbf{W}_k^{(l)} h_j^{(l-1)}\right). \tag{15}$$

GIN [54] theoretically studies the expressive power of spatial GNNs and finds that their expressive power is upper bounded by the 1-dimensional form of Weisfeiler-Lehman (1-WL) test [48] of graph isomorphism. Guided by this insight, the proposed method GIN has expressive power equal to the 1-WL test. The key is to design injective functions for $f_{agg}(\cdot)$, $f_{upd}(\cdot)$ and $f_{readout}(\cdot)$. It uses multilayer perceptron (MLP) to approximate injective functions, then the graph convolution form of GIN as follows:

$$h_i^{(l)} = \text{MLP}_1\left(\left(1 + \epsilon^{(l)}\right) \cdot h_i^{(l-1)} + \sum_{v_j \in \mathcal{N}(v_i)} \text{MLP}_2^{(l-1)}\left(h_j^{(l-1)}\right)\right), \tag{16}$$

where $\epsilon^{(l)}$ is an irrational number to distinguish the central node from its neighbors when they are combined together, thus is able to injectively encode the rooted depth-1 subtree of the central node, the same as how the Weisfeiler-Lehman test color nodes. And $f_{readout}$ is a sum pooling function with MLP to ensure injectiveness.

2.4 Comparison between spectral and spatial GNNs

Spectral GNNs have a complete theoretical foundation based on the convolution theorem and spectral graph theory. They convert graph data to spectral domain data by using the eigenvectors of the Laplacian operator on the graph data and perform learning on the transformed data. These methods rely on a fixed graph and can not learn node representations for a dynamic graph where new nodes are constantly being added to the graph. Spatial GNNs directly adopt neighborhood aggregation to imitate the standard CNN, and the representation of each node only depends on its neighborhoods rather than the whole graph. Thus, these methods are preferred over spectral models due to generality and flexibility issues. We summarize the main characteristics of GCNs surveyed in this tutorial in Table 2.

3 GNNS FOR DATA MANAGEMENT RESEARCH

In recent years, various artificial intelligence techniques have been proposed to optimize different tasks in databases. Unlike traditional machine learning and general deep learning models, GNNs show their superiority on multiple tasks due to their powerful representation capabilities. In this section, we will review the state-of-the-art progress of GNNs for databases. Table 3 summarizes the recent works on GNNs for databases.

First, we could encode the database schema to complete some tasks. Semantic parsing (e.g., transforming natural language to SQL queries) is an important problem in databases. To employ the structure information of the database schema, [1] proposes an encoder-decoder semantic parser, where the structure of the database schema is encoded with a graph neural network. The GNN model is a gated graph sequence neural network that could learn the global schema structure. The encoder and decoder then use the graph representation of the database schema to generate the SQL queries. Similar to [1], the authors of [41] also utilize GNNs to encode the database schema but they use the graph representation for a number of different tasks, including cardinality estimation, cost estimation, query clustering, and SQL-to-Text Generation. Instead of feeding the whole huge schema embedding into the learning models, [41] exploits the query-aware method to generate a sub-graph that only contains SQL-related schema information. Compared with other encoding techniques (e.g., one-hot-encoding), this GNN-based schema embedding could improve the model generalization by encoding the query semantics into the model.

Besides the database schema, the related information of SQL queries (e.g., query graphs, join graphs, join trees, and query plans) could also be encoded into feature vectors as input for learning models to accomplish different tasks, such as materialized view management [20], join order selection [5, 60], and query performance prediction [64]. For materialized view management, estimating the benefit of using an MV to answer a query (i.e., the execution time reduction of answering the query using the view) is extremely challenging, especially for dynamic workloads. [20] leverages GNNs to estimate the benefit of an MV to achieve efficient and effective dynamic MV management. Specifically, it maintains the dynamic workloads as a query graph and encodes key features of queries to model a GNN. To handle large graphs of continuously coming queries, it extracts a small subgraph for benefit estimation. The advantages of using GNN to estimate the benefit are from two aspects: (1) for each node on the query graph, GNNs could well capture the benefit from different types of neighbor nodes through several iterations of feature propagations and thereby provide accurate benefit estimation; (2) the graph structure makes the benefit estimation in parallel for each iteration, which is efficient for query rewrite and MV set maintenance. Join order selection is another essential NP-hard problem in relational databases. Different from the previous DRL-based works [23, 29], [5] uses the graph-based representation to solve this problem. It first constructs a schema graph based on the primary-foreign key relationships, from which table representations are learned to capture the correlations between tables. Then a graph convolutional network (GCN) is utilized for encoding the query graph, and a tailored-tree-based attention module is designed to encode the join plan. Compared with the simple concatenation of column embeddings in previous studies, these learned table representations could capture the correlations between tables and include more semantic information about joins. Moreover, as GCNs have relatively small-scale networks with fewer parameters, the training is more efficient than general deep-learning approaches. Unlike [5] which encodes the join graph, [60] uses a Tree-LSTM to encode the join trees for join order selection. Compared with simple one-hot-encoding for join tress [23, 29], Tree-LSTM has some advantages: 1) the learned vectors can capture the structural information of a join tree, which could lead to better plans, and 2) the learned optimizers could handle database schema changes (e.g., adding columns/tables) and multi-alias table names, which requires to retrain the neural network. Query performance prediction (e.g., execution time and resource demand) prior to execution is useful in many tasks, including admission control, query scheduling, and resource management. To capture the correlations between different concurrent queries, such as lock conflict and buffer sharing, [64] proposes a Graph Embedding Network to encode the graph features of concurrent query plans. The benefits of Graph Embedding Network are two-fold: 1) it effectively reduces computation work by extracting local structure instead of searching the whole graph, and 2) it can concurrently process the local graphs of all the vertices. Compared with previous linear regression [12] and general DNN models [30], graph embedding networks could effectively predict the performance of concurrent queries by learning the potential relationships between those queries.

Finally, we could also directly encode the data in the relational tables. [9] presents a model that uses GNNs to perform supervised learning directly on the data in relational databases. Given a relational database and a prediction target, this method first assembles all related information in the database to form a graph. Then this graph is processed by a GNN to produce a prediction on a specific column, such as time series forecasting or predicting relationships between entities. Compared with other methods, this approach has some strengths, including 1) avoiding data extraction and feature

Table 3: An overview of recent works on Graph Neural Networks for databases

Ref.	Year	Encoding Item	GNN Model	Tasks
[1]	2019	Database Schema	Gated Graph Sequence Neural Networks	Text-to-SQL Parsing
[41]	2022	Database Schema	General GNN	Query Clustering, Cardinality Estimation, Cost Estimation, SQL-to-Text Generation
[20]	2022	Query Graph [1]	General GNN	Dynamic Materialized View Management
[5]	2022	Query Graph [2]	Graph Convolutional Network	Join Order Selection
[60]	2020	Join Trees	Tree-LSTM	Join Order Selection
[64]	2020	Concurrent Query Plans	Graph Embedding Network	Query Performance Prediction
[9]	2020	Relational Tables	Graph Convolutional Network	Prediction in a Particular Column

[1] In this paper, a query graph is built by merging query plan trees of queries in the workload.
[2] In this paper, a query graph is an undirected graph where each node represents a table and each edge represents the join predicate between two tables.

engineering efforts, and 2) preserving the relational structure of the original data.

4 OPEN PROBLEMS AND CHALLENGES

While some progress has already been made, the research on GNNs for databases has just begun, and there are many opportunities and open problems for further exploration.

From the database perspective. GNNs have been proposed to solve some data management tasks in recent years. However, there are still some open problems from the database perspective view. Some possible problems include:

(1) Besides database schema, query graph, query plan, and join tree, what other structures in databases are in the form of trees or graphs, or what structures can be modeled or transformed into graphs?
(2) Considering the current learning models in databases, which component could be replaced or enhanced with GNNs, such as the query encoding part of cardinality estimation or the join tree encoding part of join order selection?
(3) For the current learning models for databases, is it necessary to replace all the pipelines with GNNs or only replace the encoding component with GNNs?

From the GNNs perspective. From the perspective view of GNNs, some basic problems also need to be further researched, such as:

(1) What are the advantages of using GNNs for databases compared to traditional machine learning and general deep learning models? Are the GNN algorithms really suitable for replacing other deep learning approaches for databases?
(2) Considering that there are already many works about AI4DB, could we replace the previous learning models with GNNs to further improve the performance, such as generalization ability, adaptability, and training efficiency?
(3) Since we have so many GNN models and algorithms, how to find the most suitable GNN model for specific database problems?

5 TUTORIAL ORGANIZATION

The tutorial is planned for 1.5 hours (90 minutes) and is divided into the following parts:

Background and Introduction (5'). We introduce the background and remarkable progress of GNNs. We show that GNNs are suitable for solving a lot of database problems.

Graph Neural Networks (45'). We introduce the basics of GNNs, including the theoretical foundations and state-of-the-art GNN methods on various applications.

GNNs for Databases (30'). We present recent research progress of GNNs in various database tasks.

Open Problems and Challenges (10'). We discuss open problems and open challenges to apply GNNs for databases.

6 GOALS OF THE TUTORIAL

Learning Outcomes. The main learning outcomes of this tutorial include: (1) understanding the differences between GNNs and general DNNs. (2) learning the fundamentals and techniques of modern GNNs. (3) knowing the recent research work of GNNs in databases. (4) identifying some open problems and future challenges of applying GNNs in databases.

Intended Audience. This tutorial is intended for a broad scope of audience ranging from database systems researchers to industry practitioners, with a focus on GNN and its application in databases. Basic knowledge of graph theory, deep learning, and databases is sufficient to follow this tutorial. Some background in graph machine learning techniques would be fairly helpful.

7 SHORT BIOGRAPHIES

Wei Zhuo is a Ph.D. student at Sun Yat-sen University. He is also a visiting student at the University of Helsinki. His research concentrates on data mining and machine learning on graphs.

Zhengtong Yan is a Ph.D. student at the University of Helsinki. His research topics lie in machine learning and reinforcement learning for databases.

Guang Tan is a Professor at the School of Intelligent Systems Engineering, Sun Yat-sen University (SYSU), China. He works on the design and evaluation of AI-empowered network systems. Before joining SYSU, he was a Professor at the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He is a member of the IEEE and CCF.

Jiaheng Lu is a Professor at the University of Helsinki, Finland. His main research interests lie in database systems specifically in the challenge of efficient data processing from real-life, massive data repositories and the Web. He has written four books on Hadoop and NoSQL databases, and more than 130 journal and conference papers published in SIGMOD, VLDB, TODS, TKDE, etc.

REFERENCES

[1] Ben Bogin, Jonathan Berant, and Matt Gardner. 2019. Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4560–4565.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[3] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record* 1748, 1 (2001), 96–102.

[4] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247* (2018).

[5] Jin Chen, Guanyu Ye, Yan Zhao, Shuncheng Liu, Liwei Deng, Xu Chen, Rui Zhou, and Kai Zheng. 2022. Efficient Join Order Selection Learning with Graph-based Representation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 97–107.

[6] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1082–1090.

[7] Fan RK Chung. 1997. *Spectral graph theory*. Vol. 92. American Mathematical Soc.

[8] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE transactions on knowledge and data engineering* 31, 5 (2018), 833–852.

[9] Milan Cvitkovic. 2020. Supervised learning on relational databases with graph neural networks. *arXiv preprint arXiv:2002.02046* (2020).

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.

[12] Jennie Duggan, Ugur Cetintemel, Olga Papaemmanouil, and Eli Upfal. 2011. Performance prediction for concurrent database workloads. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 337–348.

[13] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. Bine: Bipartite network embedding. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 715–724.

[14] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations (ICLR)*.

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[16] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[19] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[20] Yue Han, Chengliang Chai, Jiabin Liu, Guoliang Li, Chuangxian Wei, and Chaoqun Zhan. 2023. Dynamic Materialized View Management using Graph Neural Network. In *Accepted by ICDE*.

[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[22] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. *arXiv preprint arXiv:1911.05485* (2019).

[23] Sanjay Krishnan, Zongheng Yang, Ken Goldberg, Joseph Hellerstein, and Ion Stoica. 2018. Learning to optimize join queries with deep reinforcement learning. *arXiv preprint arXiv:1808.03196* (2018).

[24] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing* 67, 1 (2018), 97–109.

[25] Guoliang Li, Xuanhe Zhou, and Lei Cao. 2021. AI meets database: AI4DB and DB4AI. In *Proceedings of the 2021 International Conference on Management of Data*. 2859–2866.

[26] Guoliang Li, Xuanhe Zhou, and Lei Cao. 2021. Machine learning for databases. In *The First International Conference on AI-ML-Systems*. 1–2.

[27] Guoliang Li, Xuanhe Zhou, and Lei Cao. 2021. Machine learning for databases. *Proceedings of the VLDB Endowment* 14, 12 (2021), 3190–3193.

[28] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[29] Ryan Marcus and Olga Papaemmanouil. 2018. Deep reinforcement learning for join order enumeration. In *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. 1–4.

[30] Ryan Marcus and Olga Papaemmanouil. 2019. Plan-Structured Deep Neural Network Models for Query Performance Prediction. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1733–1746.

[31] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. 2018. Graph signal processing: Overview, challenges, and applications. *Proc. IEEE* 106, 5 (2018), 808–828.

[32] Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).

[33] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[34] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 459–467.

[35] Howell Reichel, Angela Kassulke, and Nadia Konopelski. 2007. MARCHS ADVANCED ORGANIC CHEMISTRY: REACTIONS MECHANISMS AND STRUCTURE. (2007).

[36] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. 2017. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078* (2017).

[37] Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo. 2011. Exploiting place features in link prediction on location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1046–1054.

[38] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[39] Weijing Shi and Raj Rajkumar. 2020. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1711–1719.

[40] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[41] Xiu Tang, Sai Wu, Mingli Song, Shanshan Ying, Feifei Li, and Gang Chen. 2022. PreQR: Pre-training Representation for SQL Understanding. In *Proceedings of the 2022 International Conference on Management of Data*. 204–216.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[43] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ

[44] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. *ICLR (Poster)* 2, 3 (2019), 4.

[45] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *Thirty-first AAAI conference on artificial intelligence*.

[46] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* 38, 5 (2019), 1–12.

[47] Zixiao Wang, Yuluo Guo, Jin Zhao, Yu Zhang, Hui Yu, Xiaofei Liao, Hai Jin, Biao Wang, and Ting Yu. 2022. GIDN: A Lightweight Graph Inception Diffusion Network for High-efficient Link Prediction. *arXiv preprint arXiv:2210.01301* (2022).

[48] Boris Weisfeiler and Andrei Leman. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series* 2, 9 (1968), 12–16.

[49] Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. 2014. Exploiting social network structure for person-to-person sentiment analysis. *Transactions of the Association for Computational Linguistics* 2 (2014), 297–310.

[50] Xixi Wu, Yun Xiong, Yao Zhang, Yizhu Jiao, Caihua Shan, Yiheng Sun, Yangyong Zhu, and Philip S Yu. 2022. CLARE: A Semi-supervised Community Detection Algorithm. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2059–2069.

[51] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[52] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a

benchmark for molecular machine learning. *Chemical science* 9, 2 (2018), 513–530.

[53] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2019. Graph Wavelet Neural Network. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1ewdiR5tQ

[54] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

[55] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*. PMLR, 5453–5462.

[56] Zhengtong Yan, Jiaheng Lu, Naresh Chainani, and Chunbin Lin. 2021. Workload-Aware Performance Tuning for Autonomous DBMSs. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2365–2368.

[57] Zhengtong Yan, Jiaheng Lu, Qingsong Guo, Gongsheng Yuan, Calvin Sun, and Steven Yang. 2022. Make Wise Decisions for Your DBMSs: Workload Forecasting and Performance Prediction Before Execution. In *27th International Conference on Database Systems for Advanced Applications (DASFAA)*. Springer.

[58] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.

[59] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Badly for Graph Representation?. In *Thirty-Fifth Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=OeWooOxFwDa

[60] Xiang Yu, Guoliang Li, Chengliang Chai, and Nan Tang. 2020. Reinforcement learning with tree-lstm for join order selection. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1297–1308.

[61] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[62] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on Large-scale Text-attributed Graphs via Variational Inference. *arXiv preprint arXiv:2210.14709* (2022).

[63] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

[64] Xuanhe Zhou, Ji Sun, Guoliang Li, and Jianhua Feng. 2020. Query performance prediction for concurrent queries using graph embedding. *Proceedings of the VLDB Endowment* 13, 9 (2020), 1416–1428.

[65] Hao Zhu and Piotr Koniusz. 2020. Simple spectral graph convolution. In *International Conference on Learning Representations*.

[66] Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*. 499–508.

[67] Wei Zhuo and Guang Tan. 2022. Proximity Enhanced Graph Neural Networks with Channel Contrast. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2448–2455. https://doi.org/10.24963/ijcai.2022/340 Main Track.