

## Review Article

# Survey of Graph Neural Networks and Applications

Fan Liang,<sup>1</sup> Cheng Qian,<sup>2</sup> Wei Yu ,<sup>2</sup> David Griffith,<sup>3</sup> and Nada Golmie<sup>3</sup>

<sup>1</sup>Sam Huston State University, USA

<sup>2</sup>Towson University, USA

<sup>3</sup>National Institute of Standards and Technology (NIST), USA

Correspondence should be addressed to Wei Yu; [wyu@towson.edu](mailto:wyu@towson.edu)

Received 13 April 2022; Accepted 27 June 2022; Published 28 July 2022

Academic Editor: Wei Li

Copyright © 2022 Fan Liang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The advance of deep learning has shown great potential in applications (speech, image, and video classification). In these applications, deep learning models are trained by datasets in Euclidean space with fixed dimensions and sequences. Nonetheless, the rapidly increasing demands on analyzing datasets in non-Euclidean space require additional research. Generally speaking, finding the relationships of elements in datasets and representing such relationships as weighted graphs consisting of vertices and edges is a viable way of analyzing datasets in non-Euclidean space. However, analyzing the weighted graph-based dataset is a challenging problem in existing deep learning models. To address this issue, graph neural networks (GNNs) leverage spectral and spatial strategies to extend and implement convolution operations in non-Euclidean space. Based on graph theory, a number of enhanced GNNs are proposed to deal with non-Euclidean datasets. In this study, we first review the artificial neural networks and GNNs. We then present ways to extend deep learning models to deal with datasets in non-Euclidean space and introduce the GNN-based approaches based on spectral and spatial strategies. Furthermore, we discuss some typical Internet of Things (IoT) applications that employ spectral and spatial convolution strategies, followed by the limitations of GNNs in the current stage.

## 1. Introduction

The artificial neural network is a viable technique to carry out the data-driven modeling and analysis of complex systems, which can be further used to improve their efficient and intelligent monitoring, control, and management. The existing neural networks (Convolutional Neural Networks (CNNs) [1], Recurrent Neural Networks (RNNs) [2], etc.) have been devoted to different problem domains (IoT, etc.), supporting a variety of tasks (prediction, classification, identification, tracking, codesign, and data generation, among others [3–9]). Neural networks can improve the efficiency of data analytics and find the hidden relationships in datasets collected from complex and dynamic systems by carrying out feature extraction from the investigated datasets. Because of the rapid development of computing capacity and optimized architectures and models, neural networks have shown great ability to process large complex data [10–12]. For example, CNNs deal with image classification,

video, text recognition, and audio identification by treating all the training data as data in Euclidean space. For instance, in image classification, CNNs represent an image as a matrix with fixed dimensions, a typical way of processing datasets in Euclidean space. Then, CNNs leverage a fixed filter to extract the features in the image so that meaningful local features can be identified and aggregated, which can be further used to conduct image recognition and classification.

Although CNNs and RNNs have been successful on data in Euclidean space, how to deal with data in non-Euclidean space remains an unsolved problem. For instance, social media and e-commerce applications are typical scenarios that generate graph data as typical examples of data in non-Euclidean space. Social media and e-commerce applications enable individuals to provide recommendation and rating information. Those applications analyze multiple data sources (e.g., user data, market data) to provide precise information to meet individual needs. As the data from different users and markets is dynamic and associated with

different dimensions, it is hard to formalize such data in Euclidean space. Instead, such data can easily be represented by a graph structure, but existing CNNs and RNNs cannot handle data with graph structure. Thus, the complexity, high diversity, and irregularity of graph data pose significant challenges to existing neural networks. Note that the data in non-Euclidean space has no fixed structure, and the relationship between unordered nodes changes rapidly over time. Thus, the convolution operation of existing neural networks cannot extract features from data in non-Euclidean space. For example, a CNN uses fixed convolutional kernel “filters” to extract features from 2D or 3D datasets with static structures. Nonetheless, it is impossible to use fixed filters to extract features from non-Euclidean space datasets since the data in non-Euclidean space has dynamic structures. Thus, it is necessary to design a new type of neural network to process the non-Euclidean space data.

The graph neural network (GNN), as a new type of neural network, has been proposed to extract features from non-Euclidean space data. Motivated by CNN, a GNN enables the use of a scalable kernel to perform convolutions on non-Euclidean space data. To achieve the convolution operation in non-Euclidean space, Sener and Savarese [13] proposed an active learning scheme, which enables the convolutional kernel to select the appropriate shape by itself so that the dynamic shape of non-Euclidean space data can be fit. Based on the active learning scheme, Bronstein et al. [14] leveraged a variable convolutional kernel to extract features from non-Euclidean space data, which is data-driven convolution. Some existing studies survey different variants of GNN [15–17]. There are a number of new GNN models and approaches that have been investigated in the past few years, such as spatial convolution approaches [18, 19], which were not covered by the existing surveys on GNN. The study [20] investigated the existing efforts on optimizing hardware and software to improve the GNN performance. In contrast, our study focuses on introducing the development of GNN that extends the convolutional approaches from Euclidean space to Non-Euclidean space. We detail the existing GNN-based approaches using spectral and spatial strategies. Furthermore, we review and discuss the efforts on applying GNN to some typical IoT systems in energy, transportation, and industrial domains.

To summarize, our contributions are as follows:

- (i) We introduce the background of artificial neural networks in detail and review the development history. Based on the limitation of traditional artificial neural network models (RNN, CNN, etc.) in non-Euclidean space data, we discuss the issues of using RNNs and CNNs to analyze non-Euclidean space data in detail. We also clarify the motivations for expanding the neural networks from Euclidean space to non-Euclidean space data
- (ii) We survey the existing strategies to analyze non-Euclidean space data. We present a comprehensive review of GNN models that can deal with non-Euclidean space data. We introduce the strategies

from the spectral and spatial domains and discuss their pros and cons. We also summarize the existing studies and research efforts on GNN and discuss representative techniques

- (iii) As IoT is one potential application for GNN, we review some existing studies and present some examples of leveraging GNN in different IoT systems such as carrying out traffic prediction in the smart transportation system, electricity prediction in the smart grid (e.g., power outage, solar irradiance), and resource management in industrial IoT (IIoT) systems. We also discuss some limitations of GNNs from the universality and learning complexity aspects

Note that there are some existing GNN survey papers [16, 17, 20–22]. For example, Zhang et al. [21] focused on the generalization ability of graph-based models. Wu et al. [22] conducted a comprehensive review of existing GNN models and proposed a model structure-based taxonomy. Likewise, Abadal et al. [20] reviewed the existing efforts on optimizing hardware and software from the computing aspect to improve the performance of GNNs. In contrast, our study focuses on introducing the development of GNN that extends the convolutional approaches from Euclidean space to non-Euclidean space. We detail the existing GNN-based approaches and categorize the existing GNN model into spectral and spatial strategies that depend on training data transformation. Furthermore, we review and discuss the efforts on applying GNN to some typical IoT systems in energy, transportation, and industrial domains, as well as the limitations of GNNs.

The remainder of this paper is organized as follows: in Section 2, we discuss the background and basic concepts of artificial neural networks and GNNs. In Section 3, we focus on the spectral strategy for convolution operations in graphs. Specifically, we introduce the spectral graph theory and introduce the detail of spectral convolution operations. We then review and summarize the existing efforts on spectral GNN. In Section 4, we focus on the spatial strategy for convolution operations in graphs. Specially, we begin with the introduction of spatial graph convolution and review spatial GNNs. In Section 5, we introduce some typical GNN applications in IoT, including transportation traffic prediction, electrical energy prediction, and resource allocation in IIoT. In Section 6, we discuss the limitations of the GNN model with respect to universality and learning complexity. Finally, we conclude the paper in Section 7.

## 2. Background

In this section, we first review the basic concepts of artificial neural networks and present the fundamentals of GNNs.

*2.1. Artificial Neural Networks.* Artificial neural networks leverage the concept of biological neurons so that complex computing tasks (recognition, classification, etc.) can be carried out. Artificial neural networks are based on weighted and directed graphs, whose vertices can be considered as

neurons and whose edges can be considered as synapses that connect neurons. In each neuron, transcendental functions are applied to compute and aggregate the weighted sum of outputs from the previous neurons, which pass the results (or experience) to nearby connected neurons. By doing this, the results from different neurons can be aggregated and synthesized to produce complex outputs such as image classifications.

Since the 1940s, artificial neural networks have experienced several cycles of failures and revivals. For instance, in 1961, Rosenblatt [23] proposed the fundamental concept of perceptrons, which are the foundation of artificial neural networks. Because of limited computing ability, it was hard to implement complex neural networks at that time. Then, in the 1980s, Hopfield [24] first implemented a neural network on emergent collective physical computing nodes. Motivated by Hopfield's work, Sejnowski [25] proposed the Boltzmann machine, which leverages stochastic binary processing units to solve interactions of different neurons in nonlinear networks. This effort significantly reduces the time complexity for training a neural network and introduces the slow-incremental learning to overcome forgetting. Furthermore, Werbos [26] proposed the backpropagation algorithm, which iteratively optimizes the weights and biases for recurrent systems and improves the performance of neural networks.

Increases in computing power resulted in new efforts to implement neural networks in the 1990s. As an example, Pineda [27] generalized the backpropagation algorithm to RNNs and leveraged the optimized backpropagation algorithm to improve the computational ability on nonlinear functions. McEliece developed [28] the theory of information and coding, which provides the mathematical framework to support simulating neural network by computer. After that, more studies focused on creating complex regression and classification functions. Specifically in the statistics area, Long [29] designed a functional nonlinear model that involves hundreds of variables. Also, the Deep Neural Network (DNN) and CNN extend the data from the time domain to the spatial domain, which leverages the convolutional kernel to extract features from two-dimension and three-dimension datasets [30, 31].

**2.2. Graph Neural Networks (GNNs).** In the following, we review GNNs. Specifically, we first introduce the graph dataset, which is typical non-Euclidean space data. We then introduce the motivations of processing the data in non-Euclidean space. Finally, we overview the road map of GNN.

**2.2.1. Graph Datasets.** A graph is a data structure in non-Euclidean space that consists of a set of objects (vertices) and relationships between these objects (edges). Since the graph can express complex and dynamic data, especially the logical relationships between sets of time-varying data, the graph structure can be used to represent datasets with dynamic dimensions. Examples include social network data, microscopic molecular structure data, and skeletal motion data [32]. Those datasets cannot be modeled in Euclidean space. Figure 1 illustrates the differences between Euclidean

and non-Euclidean data structures. In general, the Euclidean data has fixed dimensions and input data must be in a specific order that is determined by those dimensions. In contrast, the non-Euclidean data has dynamic dimensions and input data may not in a particular order.

**2.2.2. Motivations.** In some real-world scenarios, the data cannot be mapped to Euclidean space, which is defined by  $\mathbb{R}^n$ , meaning that Euclidean space data can be modeled and represented as a set of points in a  $n$ -dimensional linear space. For example, to present an image, we define  $x$  and  $y$  coordinates to represent the location of each pixel and a  $z$  coordinate to represent the intensity in grayscale images or a set of  $z$  coordinates to represent the intensity of each of the red-green-blue (RGB) or cyan-magenta-yellow-black (CMYK) values in color images. Thus, the image or sets of RGB or CMYK color image components can be considered data in 3-dimensional Euclidean space. However, it is hard to use  $n$ -dimensional linear space to encode some data such as social network data because of the dynamic dimensions. If we map the data to Euclidean space, important information (e.g., the relationship between data entries) will be lost. Furthermore, it is hard to keep the information by adding more dimensions since the relationship between data entries is dynamic. Thus, it is necessary to extend the set of data structures that we use for machine learning from Euclidean spaces to non-Euclidean spaces.

The non-Euclidean space data has dynamic dimension, but the typical neural network (e.g., CNN) can only define a fixed convolution kernel to aggregate the features. Thus, CNNs cannot handle non-Euclidean space data. To deal with non-Euclidean space data, GNNs can extract and combine features of multiscale local spatial data with high representation capabilities, which extends the deep learning models to present non-Euclidean space data. The key benefits of CNNs (e.g., the local connection, shared weights, and multilayer usage) are inherited by GNNs. These characteristics are important for solving problems in graph-based applications. As a unique non-Euclidean data structure, graphs have drawn attention to node classification, link prediction, and cluster analysis [25]. Due to high interpretability, GNN has recently become a widely used graph analysis method.

**2.2.3. Road Map of GNN.** Figure 2 presents the road map of GNNs, which is inspired by CNNs. Similar to CNN, in order to aggregate data features, GNN employs the convolution process. The difference between GNNs and CNNs is that a GNN processes convolution in the graph while a CNN processes discrete convolution in Euclidean space data. The computational complexity of ordinary convolution is defined by the number and the size of convolutional kernels. The non-Euclidean space data is high-dimensional data. As the number of dimensions increases, the number of convolutional kernels grows, which leads to the significant increase of computation complexity. In addition, using a fixed-size convolutional kernel could result in the loss of critical information of non-Euclidean space data. Traditional discrete convolution cannot maintain translation invariance on

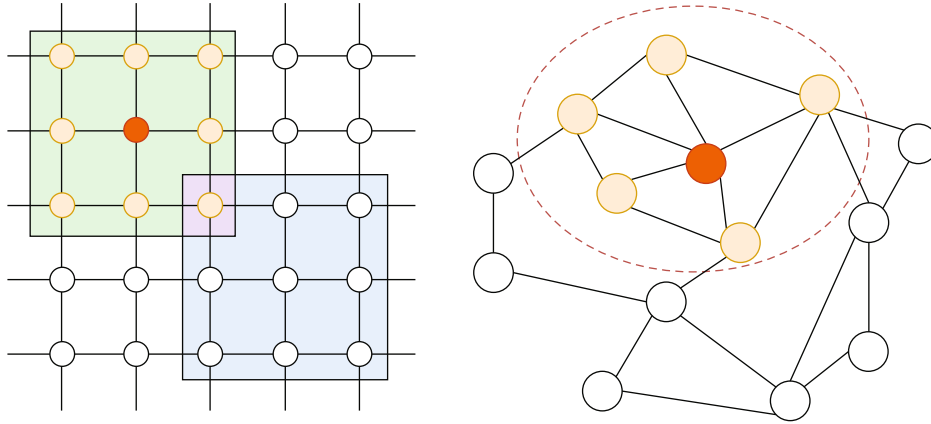


FIGURE 1: Examples of Euclidean (left) and non-Euclidean (right) data structures, with subsets indicated by the colored regions.

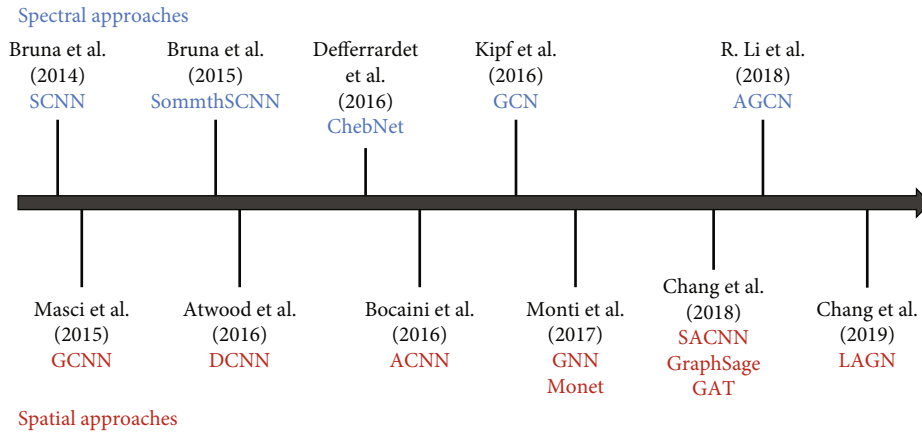


FIGURE 2: Road map of GNN evolution.

non-Euclidean space data. Since the data structure is fixed in Euclidean space data, the number of neighbors is the same for each data element. Thus, discrete convolution can leverage a fixed-size convolution kernel to aggregate the features. However, for non-Euclidean space data, as the number of adjacent vertices of each vertex in the topological graph could be different, it is hard to use a fixed-size convolution kernel.

To deal with non-Euclidean space data, one possible way is to leverage dynamic kernel size to adapt to the data. Chollet [33] proposed a depth separable convolution, which specifically focuses on solving the dynamic dimension problem of non-Euclidean space data. Compared to the traditional convolution, the depth separable convolution employs two different sizes of convolutional kernels. It first leverages a convolutional kernel in larger size to extract the features. Then, it pushes the extracted features to pass a smaller size convolutional kernel so that the features are aggregated. The dynamic convolutional kernel size is capable of handling the dynamic dimension of data.

Another possible way is to increase the size of convolutional kernels, which can increase the receptive fields of con-

volutional kernels. By doing this, the features can be aggregated as much as possible. In order to increase the receptive fields of convolutional kernels, the dilated convolution is proposed in several studies [34–36]. The benefit of dilated convolution is keeping the actual size of convolutional kernels (for the sake of computation complexity) but increasing the receptive fields. By setting different dilated rates, the dilated convolution can significantly increase the ability of high-dimensional data classification.

However, the aforementioned approaches have some limitations. First, it needs to adjust either the kernel size or dilated rate manually based on the characteristics of the data, such as the level of connectivity in the graph. Thus, the user is hard to create a generic network model to fit different datasets. Second, it is still much difficult to handle the complex graph data structure as it is hard to apply the weight information to network models. Thus, GNNs were proposed by combining graph theory and convolution so that graph data can be processed effectively [23]. There are two main strategies to implement a GNN. One strategy relies on spectral graph theory that converts data from spatial domain to spectral domain for further processing [33]. The other

TABLE 1: Notations.

Symbols	Descriptions
$\lambda_l$	The $l^{\text{th}}$ eigenvalue of the Laplacian matrix
$U$	Feature matrix
$G$	Graph with $V$ vertices and $E$ edges
$D$	Degree matrix of vertices (diagonal matrix)
$A$	Adjacency matrix of the graph
$F_1(w)$	Spectral input signal
$F_2(w)$	Spectral convolution kernel
$g_\theta$	Self-learning parameter
$\Lambda$	Diagonal matrix
$X$	$N$ -dimensional vector
$\hat{a}_G$	Graph convolution
$\tilde{\Lambda}$	Adjustable eigenvector matrix
$T_k(x)$	Chebyshev polynomial obeys
$K$	Number of neighbors
$\alpha$	Attention factors
$h$	Aggregation result of GAT

strategy does not rely on graph theory and directly involves the convolution process in the spatial domain [33]. We will discuss both strategies in the following two sections.

### 3. Spectral Convolution Strategy

In this section, we describe the spectral convolution strategy. Specifically, we first briefly describe the fundamentals of spectral convolution. We then survey the existing GNN models that leverage the spectral convolution strategy. Furthermore, we categorize and compare the existing GNN models. Table 1 lists the key notations in the paper.

**3.1. Spectral Graph Theory.** To address the issue that CNN cannot process convolution on the graph, Shuman *et al.* [37] proposed a graph signal processing- (GSP-) based scheme using spectral graph theory. The GSP defines the Fourier transform on the graph. Further, on the spectral domain, GSP defines the convolution on the graph. Based on the GSP, Bruna *et al.* [38] proposed the Spectral Convolution Neural Network (SCNN) to aggregate the features from graphs, which is the initial graph neural network. Thus, the spectral graph theory is fundamental to the GNN.

The discrete Fourier transform is applied when converting graphs from spatial domain to spectral domain. This process is denoted as the graph Fourier transform. Before applying discrete Fourier transform, a set of orthogonal bases for vertices is required. We first perform a spectral decomposition of the Laplacian matrix, which is the difference of the degree matrix (whose  $n$ th diagonal element is the degree of the  $n$ th vertex) and at adjacency matrix (whose  $(i, j)$ th element is 1 if there is an edge between vertices  $i$  and  $j$ ). This results in  $n$  linear independent eigenvectors that form an orthogonal basis. The graph Fourier

transform projects the orthogonal bases into the orthogonal space, which is equivalent to expressing the arbitrary vector defined on the graph as a linear combination of the eigenvectors of the graph's Laplacian matrix.

The standard graph Fourier transform can be represented by

$$\mathcal{GF}[f](\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^N f(i)u_l(i). \quad (1)$$

Here,  $f$  is a function that maps the vertices of the graph to a number on the real line. Also,  $\lambda_l$  represents the  $l^{\text{th}}$  eigenvalue of the Laplacian matrix, and  $u_l(i)$  represents the  $i^{\text{th}}$  element of the  $l^{\text{th}}$  eigenvector. The graph Fourier transform (discrete Fourier transform) is an inner product operation for  $\lambda_l$  and  $u_l$ . Then, apply matrix multiplication, on Equation (1), as the matrix expression of the graph Fourier transforms. After simplification, the matrix multiplication can be represented by Equation (2). The graph Fourier transform is equivalent to the inner product of transpose matrix of feature matrix and  $N$ -dimensional vector as

$$\hat{f} = U^T f. \quad (2)$$

Here, the feature matrix  $U$  can be computed by the Laplace matrix, which is defined by  $L = D - A$ . Denote graph as  $G = (V, E)$ ,  $L$  as Laplace matrix,  $D$  as the degree matrix of vertices (diagonal matrix), and  $A$  as the adjacency matrix of the graph. Figure 3 illustrates the example of those matrices for a graph. Then, based on the Laplace matrix  $L$ , we can solve the feature matrix  $U$ , which is the Eigen decomposite.

**3.2. Spectral Graph Convolution.** After reviewing the spectral graph theory, we now introduce the spectral graph convolution. The standard convolution mechanism in spectral graph theory can be implemented using the graph Fourier transform, because of the duality between convolution and multiplication that is similar to what we observe with the standard Fourier transform:

$$\mathcal{F}[f_1(t) * f_2(t)] = F_1(w) \cdot F_2(w), \quad (3)$$

where we denote  $f_1(t)$  as the spatial input signal and  $f_2(t)$  as the spatial convolution kernel. It also defines  $F_1(w)$  as the spectral input signal and  $F_2(w)$  as the spectral convolution kernel. Also, denote “ $*$ ” as convolution operation and “ $\cdot$ ” as product operation. Equation (3) shows the implementations of spectral graph convolution. First, it converts the spatial signal to spectral domain and obtains the product of spectral signal and the spectral convolution kernel. Then, we can take the inverse graph Fourier transform of the product of spectral signal and convolution kernel to get the final result in the spatial domain.

As aforementioned, the convolution operation in the spatial domain has the requirements of sequence order and fixed dimensions, which raises difficulties for processing graph data. For example, given the convolution kernel size,  $3 \times 3$ , we define the center of the kernel as the *central*

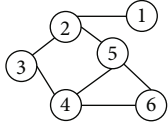
Labeled graph	Degree matrix
	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$
Adjacency matrix	Laplacian matrix
$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}$

FIGURE 3: Example of degree, adjacency, and Laplacian matrices for an undirected graph.

element and the surrounding neighbors as the *receptive field*. In the spatial domain, the kernel size is fixed. Thus, for each central element of input data, a  $3 \times 3$  kernel has eight neighboring elements for convolution operation, indicating that the size of the receptive field must be fixed. However, in graphs, the number of neighbors is uncertain; thus, it is hard to identify the fixed size of the receptive field. In addition, elements are not homogeneously arranged in graphs. Thus, it is difficult to assign a single value to the length of the step of convolution kernel. However, in the spectral domain, we can use the adjustable component of each frequency to deal with the dynamic receptive field. The adjustable component changes along with the receptive field changes so that a suitable receptive field for the spectral domain dynamic situation can be established.

**3.3. Typical Spectral GNN.** In the following, we introduce some typical spectral GNN models and review the existing studies that leverage the spectral convolution strategy.

**3.3.1. Spectral CNN.** Bruna et al. [38] proposed the spectral graph convolution network. It uses the self-learning diagonal matrix instead of the spectral domain convolution kernel. They modified Equation (3) and proposed a self-learning parameter set  $g_\theta$ . The convolution kernel can be represented by

$$X \star_G g_\theta = U \Lambda U^T X, \quad (4)$$

which is the discretization of the anisotropic convolution. Here,  $X \in \mathbb{R}^N$  is a  $N$ -dimensional vector. The  $\hat{a}_G$  represents graph convolution. Also,  $U$  is a matrix that includes eigenvectors and  $\Lambda$  is a diagonal matrix. In this study, the authors leveraged the self-learning diagonal matrix to calculate the eigenvalues of vertices.

There are some limitations of spectral CNN. The first is the high time complexity. In a graph, the feature decomposition of the Laplacian matrix needs to be performed on all the elements. In addition, we have to compute to product of  $U$ ,  $\Lambda$ , and  $U^T$  in each forward propagation. The time complexity is high, especially for some large-scale graphs.

Furthermore, the number of parameters of the convolution kernel depends on the number of vertices. Thus, this approach is not suitable for a graph that consists of a large number of vertices.

**3.3.2. ChebNet.** In order to reduce the computational (or time) complexity, Defferrard et al. [39] proposed a Chebyshev polynomial as a filter so that the time complexity of the convolution kernel can be reduced by leveraging polynomial fitting. The key idea of this scheme is to define the Chebyshev polynomial filter. As we discussed before, the time complexity of Equation (4) is high. The time complexity of the eigenvector matrix  $U$  is  $O(n^2)$ . To reduce the time complexity, Hammond et al. [40] proposed a polynomial approximation for the self-learning parameter set,  $g_\theta$ , which is given by the truncated Chebyshev polynomial expansion:

$$g_\theta = g_\theta(\Lambda) \approx \sum_{i=0}^{K-1} \theta_i T_k(\tilde{\Lambda}). \quad (5)$$

Here,  $\tilde{\Lambda}$  is an adjustable eigenvector matrix, which is adjusted to satisfy the requirement of truncated Chebyshev polynomial expansion. Also,  $\theta_i$  is the Chebyshev coefficient and the Chebyshev polynomial obeys the recursion relation  $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ .

Compared to spectral CNN, the ChebNet expression is  $K$ -localized and has local connectivity as it is a  $K^{\text{th}}$ -order polynomial. In addition, the ChebNet expression identifies the longest step of the convolution process, which is  $K$  distance from the center element. The time complexity of  $T_k(\tilde{\Lambda})$  is  $O(|E|)$ , which has is proportional to the number of edges  $E$ . Thus, the overall time complexity of the expression is  $O(K|E|)$ . When the input data is a sparse graph, it can significantly reduce the time complexity, which is much smaller than  $O(n^2)$ .

**3.3.3. Graph Convolution Network.** Based on the spectral convolution and ChebNet model, Kipf and Welling [41] proposed graph convolutional networks (GCNs), also named as first-order ChebNet. The GCN is a basic GNN

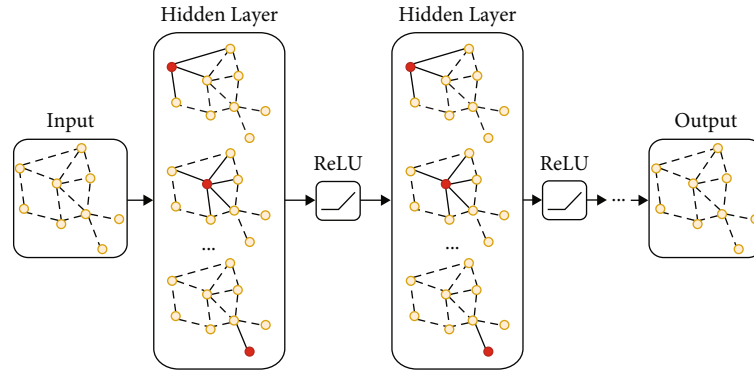


FIGURE 4: GCN model.

model, and a number of studies are based on GCNs to develop new variations of GNN models. There are two key contributions in [41]. First, compared to the schemes that directly operate on graph structure data [38, 39], a simple approximate layer based on the first-order approximation was proposed so that the calculation could be simplified. Second, the graph structure neural network model was validated for conducting quick and scalable processing of the semisupervised classification in graph-related data. The efficiency and accuracy of the proposed scheme were validated on some existing public datasets.

Figure 4 illustrates the structure of GCN. As shown in the figure, the input is an entire graph. In Convolution Layer 1, a convolution operation is performed on the neighbors of each node, and the node is updated with the result of convolution. Then, the GCN applies the activation function (e.g., ReLU) to the convolution results. Following the activation layer, the output is pushed into another convolution layer and activation layer, which is the second loop. After that, the process is repeated until the output approaches the accuracy requirement. Thus, the GCN can increase the depth of convolution layers so that the accuracy requirement for each specific case can be satisfied. The GCN has a local output function, which is used to convert the state of the node (including the hidden state and the node feature) into the task-related tags.

In general, there are two different tasks for a GCN. The first is node level task, such as the classification of social media accounts, which focuses on the classification of different nodes. Each social media account can be a node in the graph. Classifying the account is equivalent to classifying nodes in the graph. Second is graph level task, such as the classification of compounds, which focuses on the classification of different graphs. The compound can be formalized as a graph. Classifying the compound is equivalent to classifying the graph. Convolution operations focus on updating the hidden state of individual nodes, which extract the features of nodes [42–44]. In order to aggregate the features for the entire graph, the GCN employs differentiable pooling after convolution layers [45, 46]. By doing this, the GCN can extract node features and leverage soft clustering and node features to compute the graph representation. Based on the graph representation, the features for the entire graph can be derived.

**3.3.4. Summary of Existing Spectral Convolution Schemes.** In Table 2, we summarize some existing studies, which are based upon the spectral convolution strategy. We compare the task, input parameters, input data, number of convolutional layers, performance, and time complexity for existing studies. In the table,  $A$  represents the adjacency matrix,  $X$  represents the eigenvalues of vertices, and  $X^e$  represents the eigenvalues of edges.

## 4. Spatial Convolution Strategy

In this section, we present spatial convolution strategy. In particular, we first present the principle of spatial convolution. Then, we review the existing GNN models that leverage spatial convolution. Finally, we summarize the existing studies that adopt the spatial convolution strategy.

**4.1. Spatial Graph Convolution.** There are some issues when we use the spectral graph convolution strategy. First of all, the spectral graph convolution strategy is not suitable for directed graphs. Although most real-world data can be formalized as undirected graphs, some traffic data or routing data can be formalized in directed graphs, which do not allow use of the graph Fourier transform and cannot convert spatial information to spectral domain. In addition, due to the immutability of Laplacian's feature matrix  $U$ , the structure of the graph cannot be changed during the training process (e.g., the weight of edges, adding or deleting nodes). Nonetheless, graphs are highly dynamic in real applications, including social network data. Finally, the spectral CNN is computationally intensive and time-consuming. Although the ChebNet and GCN reduce the time complexity, they only handle a small number of parameters, which is a limitation of ChebNet and GCN. Thus, there are some studies focused on the spatial convolution strategy, which tends to transform the non-Euclidean space data to Euclidean space data so that the convolution can be carried out directly. We discuss some of these studies in the next subsection.

**4.2. Typical Spatial Graph Neural Networks.** We now introduce four different neural networks that adopt the spatial convolution. Table 3 lists the features of each neural network.

TABLE 2: Summary of existing spectral convolution approaches.

Convolution schemes	Task	Input parameters	Input data	No. of layers	PERF	Time complexity
Spectral CNN [38]	Classification	$A, X$	MNIST	—	98.7%	$O(n^3)$
SyncSpecCNN [47]	Identification	$A, X$	Annotations	10	84.74%	$O(n^3)$
SyncSpecCNN [48]	Prediction	$A, X, X^e$	DBLQ network data	—	—	—
SyncSpecCNN [49]	Classification	$A, X$	Yeast dataset	—	56.0%	—
SSF-CNN [50]	Classification	$A, X, X^e$	HS, CAVE	3	—	$O(n^2)$
Graph CNN [51]	Identification	$A, X$	MNIST	—	94.23%	$O(n^3)$
GCN [52]	Prediction	$A, X, X^e$	MINI	2	77%	$O(n^2)$
Semisupervised GCN [41]	Classification	$A, X, X^e$	Cora, CiteSeer	2	70.3%	$O(K E )$
GCN [53]	Classification	$A, X$	Protein data	—	84.6%	—
Multigraph GCN [54]	Classification	$A, X, X^e$	Bunny mesh	—	93.58%	—
Local SGCN [55]	Classification	$A, X$	MNIST	2	95.74%	$O(n^2)$
S-GCN [56]	Classification	$A, X, X^e$	Reddit, Flickr	—	96.8%	$O(nr)$
S-GCN [57]	Classification	$A, X, X^e$	Cora, CiteSeer	6	83.12%	$O(K E )$
DSGCN [18]	Classification	$A, X$	ENZYMES	7	78.39%	$O(n^2)$
Semisupervised GCN [58]	Classification	$A, X$	Cora, CiteSeer	2	74.5%	—
AGCN [59]	Prediction	$A, X, X^e$	Delaney 2004	7	79.4%	$O(d^2)$
Multiscale GCN [60]	Classification	$A, X, X^e$	Cora, CiteSeer	—	79.2%	$O(n^2)$
EGCN [61]	Classification	$A, X, X^e$	RF, Weave	2	82.0%	$O(n^2)$
Functional brain network [62]	Classification	$A, X, X^e$	ABIDE	10	90.0%	$O(n^2)$
Text level GCN [63]	Classification	$A, X$	R52	—	94.6%	—
Edge-labeling GCN [64]	Classification	$A, X$	miniImageNet	—	76.4%	$O(n^2)$
Graph WNN [65]	Classification	$A, X$	Cora, CiteSeer	16	82.8%	$O(npq)$
$A^2$ GNN [66]	Classification	$A, X$	HDM05, LSC	5	98.6%	—
StemGNN [67]	Identification	$A, X, X^e$	COVID-19	—	91.74%	$O(n^2d)$
PA-GNN [68]	Defense malicious attack	$A, X$	Reddit	2	79.57%	$O(n^2)$
Quantum GNN [69]	Classification	$A, X$	Kolmogorov-Smirnoff	—	95.3%	$O(n^2)$
Recurrent multi-GNN [70]	Classification	$A, X, X^e$	Synthetic dataset	—	99.3%	$O(mn)$
Few-shot GNN [71]	Classification	$A, X$	ILSVRC-12	5	99.2%	$O(n^2)$
Transferability GNN [72]	Classification	$A, X$	Cora	—	76.5%	$O\left(\frac{1}{\sqrt{\min n_1, n_2}}\right)$
Line-GNN [73]	Classification	$A, X$	Stochastic block mode	30	93.7%	$O(n^2)$
Spectrum DNN [74]	Classification	$A, X$	UMD Wikipedia dataset	—	82.61%	$O(n^2)$
Spectral marching [75]	Identification	$A, X$	Image	—	96.1%	$O(n^2K^2)$
TGC-LSTM [76]	Prediction	$A, X, X^e$	INRIX traffic	11	97.43%	$O(n^2)$
Graph-ARMA [77]	Classification	$A, X, X^e$	MNIST, 20news	3	91.5%	$O(n^2)$
GCF [78]	Prediction	$A, X, X^e$	ML-10M, Taobao	3	79.6%	—
Spectral clustering [19]	Classification	$A, X, X^e$	Cora, CiteSeer	—	98.7%	$O(n^3)$
LB spectral filtering [79]	Classification	$A, X, X^e$	ADNI	5	91.1%	—



TABLE 2: Continued.

Convolution schemes	Task	Input parameters	Input data	No. of layers	PERF	Time complexity
GSDN-F, GSDN-EF [80]	Classification	$A, X, X^e$	Cora, CiteSeer	—	95.7%	$O(n^2)$
DAGN [81]	Classification	$A, X, X^e$	Cora, CiteSeer	24	85.4%	$O( E )$
Graph hashing network [82]	Classification	$A, X, X^e$	MNIST	6	84.2%	$O(dn^2)$
KM2A arrays [83]	Classification	$A, X$	Cosmic-ray data	—	95.3%	$O(K \epsilon )$
CayleyNets [84]	Classification	$A, X, X^e$	MNIST	2	99.18%	$O((k+1)rn)$
Median spectral graph [85]	Identification	$A, X$	Attributed graphs	—	83.2%	$O(n^2)$
GfNN [86]	Classification	$A, X$	Cora, CiteSeer	2	80.9%	$O(n^2)$
GIN [87]	Classification	$A, X$	MUTAG	—	91.6%	—
LNPP/SBMF [88]	Classification	$A, X$	Social network	—	96.8%	—
Learning graph [89]	Classification	$A, X, X^e$	Chemical molecular dataset	2	86.4%	$O(dm)$
LNN with graph sparsity [90]	Classification	$A, X$	MNIST	5	96.1%	—
GeniePath [91]	Classification	$A, X, X^e$	MINI	18	96.5%	$O( \epsilon )$
Heterogeneous GAN [92]	Classification	$A, X, X^e$	DBLP, ACM	—	84.76%	$O(V_\phi F_1 F_2 K + E_\phi F_1 K)$

TABLE 3: Example of spatial graph neural networks.

Convolution approach	Neighbor node selection	Order of neighbors	Kernel parameters
GNN	Random walk	Ordered	Not shared
GraphSAGE	Uniform sampling	Disordered	Shared
GAT	First-order neighbor node	Disordered	Shared
PGC	Sampling function	Determined by weight functions	Determined by weight functions

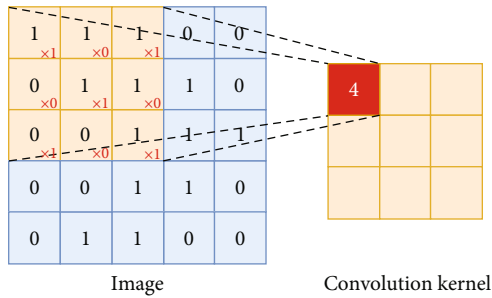


FIGURE 5: Convolution operation on non-Euclidean space data.

**4.2.1. Graph Neural Network.** The GNN transforms the graph to Euclidean space data and then applies the general convolution to the data. There are three steps to successfully complete convolution in GNN. First, it identifies the neighbor field of each node. Since the neighbor nodes are dynamic in non-Euclidean space data, it must identify the neighbor field before defining the convolution kernel. Then, it defines the field of the convolution kernel. Finally, the inner product for corresponding elements in the field and convolution kernel is performed, which is similar to convolution in Euclidean space. Figure 5 illustrates an example of convolution in

Euclidean space which adopts a 3 by 3 filter to abstract features from the original image. In addition, Figure 6 shows an example of convolution by using GNN. It clearly shows that the filter has no fixed structure in the convolutional process of non-Euclidean datasets. The structure of the filter depends on the neighbor field.

As we discussed before, determining the neighbor field is one important step. To this end, Hechtlinger et al. [93] adopted random walk to identify the neighbor field within graphs. In their study, the following parameters are defined:  $P$  matrix as the random walk transition matrix (i.e.,  $P_{ij}$  as the transition probability from node  $i$  to node  $j$ ),  $S$  matrix as the similarity matrix whose elements indicate how similar the graph nodes are, and  $D$  matrix as the degree matrix. Given a graph  $G$ ,  $P$  can be defined as  $P = D^{-1}S$ . In addition,  $P^k$  represents multistep transition matrix whose  $(i, j)$ th element is the probability of the random walk moving from node  $i$  to node  $j$  in  $k$  steps. Increasing the value of  $k$  will grow the size of neighbor field.

**4.2.2. Graph Sample and Aggregate.** Another graph convolution neural network model is Graph Sample and Aggregate (GraphSAGE), which was proposed by Hamilton et al. [94]. In this study, the graph convolution can be realized

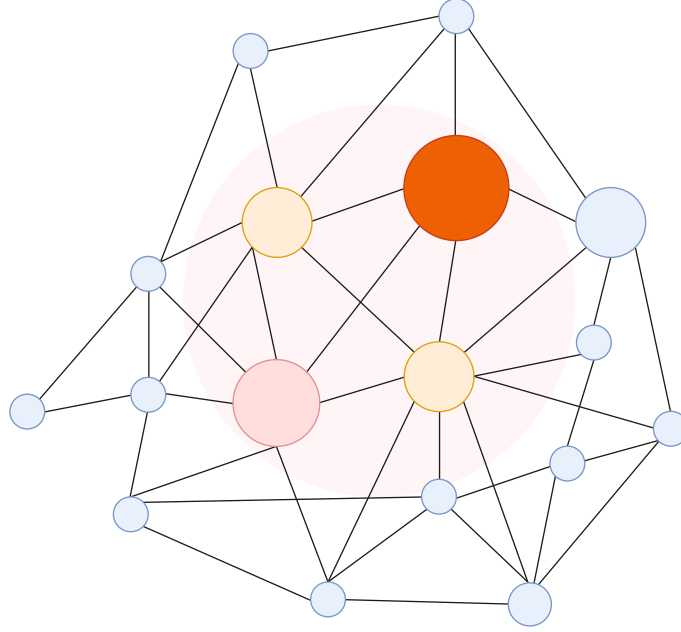


FIGURE 6: Convolution operation on Euclidean space data.

by sampling and aggregation. As a variation from GNN, the input order of aggregation functions has no impact on the result in GraphSAGE, meaning that GraphSAGE can handle disordered neighbor nodes. There are the following three key steps in GraphSAGE: (i) a fixed number of neighboring nodes is obtained through sampling; (ii) the aggregation function is used to obtain the aggregation information of neighboring nodes so that the features of the center element can be obtained; and (iii) the aggregation information of neighboring nodes is used to classify or predict the content or labels of the center element. In Hamilton et al.'s study, the authors leveraged the uniform sampling strategy to select a fixed number of neighbor nodes. Uniform sampling can be repeated on the connected first-order nodes to obtain a neighbor field with a fixed number of nodes. Furthermore, they proposed three aggregators. The first one is the mean aggregator, which is computed by weight matrices and a nonlinear activation function. The second aggregator is called the long short-term memory (LSTM) aggregator, which is an updated version of the mean aggregator. In the LSTM aggregator, the input data needs to be organized in a sequential manner. Because of the extra operation, the LSTM aggregator shows a stronger expressive ability than the mean aggregator. The last aggregator is the pool aggregator. In the pooling approach, the vectors of each neighbor are fully fed with forwarding propagation. After the forward propagation, the max-pooling aggregates features across the neighbor fields.

**4.2.3. Graph Attention Network.** Veličković et al. [95] proposed a Graph Attention Network (GAT), which introduced the attention mechanism into the graph convolution model, and used the attention mechanism to model the correlation among nodes. In this study, they summarized the weakness of GNN and GraphSAGE, which shares the same convolu-

tion kernel parameters for all the nodes. This affects the final results in some cases as the degree of association is different between nodes in the neighbor field. It is necessary to adopt different convolution kernel parameters to treat different nodes. The aggregation process of the GAT is represented by

$$\vec{h}'_i = \sigma \left( \frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j \right). \quad (6)$$

The  $\alpha_{ij}^k W^k \vec{h}_j$  represents the attention mechanism. Here,  $W$  denotes a weight matrix, and a set of attention mechanism coefficients  $\{\alpha_{ij}^k\}_{j,k}$  is included in Equation (6), which is a single forward propagation neural network. In addition, the attention mechanism adopts the ‘‘LeakyReLU’’ nonlinear activation function, which has a small negative linear slope for negative input values [96]. After defining the aggregation process of the GAT, Veličković et al. [95] leveraged Cora, CiteSeer, and PubMed datasets to evaluate the efficacy of GAT, and experimental results demonstrated the ability on classifications.

**4.2.4. Partition Graph Convolution.** In Partition Graph Convolution (PGC), the convolution process is treated as sample and weight function. Yan *et al.* [97] proposed a scheme to obtain the sample function and weight function. In detail, the sampling function selects sample nodes in the neighbor field. The key is to identify the neighbor field, which is the sampling area. For the weight function, it classifies different nodes in the neighbor field to  $K$  groups. It shares the convolution kernel parameters within the same group and does not share parameters across different groups, which reduces the shared field size and increases the final accuracy. In the study, they defined three different classification schemes.

TABLE 4: Summary of existing spatial convolution schemes.

Convolution scheme	Task	Input parameters	Input data	No. of layers	PERF	Time complexity
GNN [98]	Classification	$A, X, X^e$	SNAP social network	30	76.5%	$O(n^2)$
Enhanced GNN [99]	Classification	$A, X, X^e$	Cora, CiteSeer	—	92.5%	—
graph-CNN [100]	Classification	$A, X, X^e$	MCI/AD diagnosis	6	87.5%	—
GGNN [101]	Identification	$A, X, X^e$	Protein data	2	72.6%	$O(n^2)$
SPAGNN [102]	Prediction	$A, X$	Self-driving	4	83.9%	—
STGNN [103]	Prediction	$A, X$	METR-LA	—	93.45%	$O(n^2)$
ST-GCN [97]	Identification	$A, X$	Skeleton actions	3	88.3%	—
SIA-GCN [104]	Identification	$A, X, X^e$	Panoptic dataset	5	81.7%	$O(n^2)$
ST-ResNet [105]	Prediction	$A, X, X^e$	Traffic data	15	93.7%	—
AttConvLSTM [106]	Prediction	$A, X, X^e$	TaxiNYC	12	86.7%	—
DMVSTN [107]	Prediction	$A, X, X^e$	Didi Taxi	5	90.7%	$O(n^2)$
Sequential GNN [108]	Prediction	$A, X, X^e$	Traffic data	13	83.5%	$O(n^2)$
STAGN [109]	Classification	$A, X$	Card transaction data	5	870.0%	—
SDynamicGRCNN [110]	Prediction	$A, X, X^e$	Traffic data	—	89.7%	—
Semisupervised GNN [111]	Prediction	$A, X, X^e$	Parking dataset	12	93.1%	$O(n^2)$

The first classification scheme is called unilabeling, which is the same as GNN since it classifies all the nodes as one group. The second scheme is called distance classification, which classifies nodes according to the order. The central element is “0” order, adjacent elements are the first-order, etc. The last scheme is called spatial configuration, which is specifically for skeleton action identification. The spatial configuration defines a reference distance. Based on the reference distance, it classifies different nodes. After determining two functions, the convolution function can be represented by  $f_{out}(v_i) = \sum_{v_j \in B(v_i)} (1/Z_i(v_j)) f_{in}(v_j) \cdot W(l_i(v_j))$ . Here,  $v_j \in B(v_i)$  denotes the sampling function, which samples from  $B(v_i)$ , the set of distance-1 neighbors of node  $v_i$  in different orders. Also,  $Z_i(v_j)$  is the normalization coefficient, and  $W(\cdot)$  is the weight function. Compared to GraphSAGE that adopts mean sampling, PGC is more generic for different cases since it defines a sampling function.

**4.2.5. Summary of Existing Spatial Convolution Schemes.** In Table 4, we summarize the existing studies based on the spatial convolution strategy. Similar to Subsection 3.3.4, we also compare the task, input parameters, input data, number of convolutional layers, performance, and time complexity for the existing studies.

## 5. GNN Applications in Internet of Things (IoT)

**5.1. Overview.** Since GNN has the ability to analyze non-Euclidean space data and the IoT data is highly dynamic and dimensional, there are some potentials to leverage GNN to assist data analysis in IoT systems. In IoT systems, sensors collect data related to a variety of things, including

weather, location, and temperature. To obtain the accurate results of data analysis, existing studies usually involve multiple data sources in deep learning models and the data source is considered as a key factor in the training process of learning models. Nonetheless, the diverse data source, dynamics data, and complex relationship between different data elements make it difficult to organize and process the data as the one in Euclidean space. Traditional deep learning models (CNN, RNN, etc.) cannot treat data analysis and obtain accurate results in non-Euclidean space data. In addition, IoT data has become more complex since more IoT devices are connected to IoT systems and systems are dynamic. In order to handle the complex IoT data, GNN models can be adopted in some IoT systems for analyzing IoT data, since GNN models show great abilities in analyzing non-Euclidean space data. There are some existing studies on applying GNN to IoT systems. In the following, we review several typical IoT scenarios, which adopt GNN to assist data analysis process.

**5.2. GNN in Transportation Traffic Prediction.** The goal of traffic prediction in transportation systems is leveraging the historical traffic data and road topology to predict the traffic speed, traffic flow, and throughput so that the decisions for mitigating the traffic congestion can be provided [112–114]. Obviously, the precise prediction results can assist drivers in selecting optimal paths and reducing the occupation of transportation networks. For example, Du et al. [115] proposed a GNN-based method to carry out the prediction of traffic flows in the transportation system. In their study, they focused on the temporal traffic data that is time-series data. First, they defined the traffic data as  $V_t \in \mathbb{R}^{n \times c}$ , where  $n$  is the number of nodes and  $c$  is the number

of channels. Note that  $c$  is a  $c$ -dimensional vector for each node  $n$ , which represents the features at a specific time  $t$ . The status can be the values of traffic flow, speed, and congestion in this area. When  $c$  is 1, the model only considers one feature of the traffic. After that, they formalized the traffic prediction process. In their study, they only consider the scenario with one node. Also, the input is 1-dimensional vectors, indicating that only time domain is considered. Thus, no spatial information is involved in the model as one node is considered. Then, they arranged all the nodes to an array (1-dimensional vector) and leveraged full connected layers to analyze the data. Some similar research efforts consider different features of traffic, such as traffic flow [116], traffic speed [117], and weather [118].

Likewise, Ma et al. [119] proposed a graph CNN for the prediction of traffic congestion in transportation networks. In their study, they proposed a spatial-temporal matrix, in order to treat the time-series data at multiple locations in the transportation network as an image. This scheme applies two convolution operations on temporal and spatial domains. Thus, the model involves multiple nodes and complex situations, and their study converts graph data to images and applies two convolution operations, which do not apply the graph convolution directly on the graph datasets. Thus, the impacts of the relationship between different nodes cannot be manifested in prediction results. In order to overcome the issue, Wang et al. [120] proposed a lattice-based data representation and leveraged convolution operation in lattice-based images directly. However, this approach inherits some limitations. First, traffic data includes geographical information; however, lattice-based images cannot embed the correlation geographical information. Furthermore, as the lattice-based scheme treats different nodes as pixels, only the impact of connected neighbors is considered. In real-world scenarios, the traffic conditions in one location could be affected not only by neighboring traffic conditions but also by traffic conditions that are far away from this location.

In order to overcome the aforementioned problem, Yu et al. [121] proposed a GNN-based scheme, which creates a graph to represent the traffic conditions. They leveraged nodes to represent monitoring stations and weights to represent different features between stations. By doing this, they created a comprehensive graph to represent irregular transportation networks. When creating the graph, they used a Gaussian kernel to construct the weighted adjacency matrix,  $W$ . In detail, they adopted a Euclidean distance metric and defined a threshold  $\lambda$ . If the Euclidean distance is larger than the threshold  $\lambda$ , the relationship features can be represented by "0"; otherwise, leverage the relationship features as parameter and Gaussian kernel function to obtain the result.

Furthermore, there are other studies that focus on improving the performance of [121]. For example, Chai et al. [122] studied a multigraph convolutional network to optimize the computation of distances. Wu et al. [123] investigated a Graph WaveNet that leverages an adaptable adjacency matrix to deal with different traffic situations. Likewise, Guo et al. [124] designed an attention-based spatial-temporal graph convolutional networks, which can adapt to different traffic situations.

**5.3. GNN in Electrical Energy Prediction.** The smart grid is a typical energy-based IoT system, which adopts IoT devices (sensors, actuators, etc.) to collect massive amounts of data in electricity transmission and distribution systems. Based on the data analysis, the smart grid system can be operated efficiently and intelligently. The electric grid can be modeled as graph structure where the collected data associated with various nodes corresponds to different geographic locations. In addition, the data in the grid is highly dynamic. There are some studies on leveraging GNN to assist the data analysis process.

For example, Owerko et al. [125] proposed a GNN-based prediction scheme to prevent power outages. Since GNNs can process highly dynamic data (e.g., graph data), they considered not only the historical data of the electrical grid but also weather conditions, geographical location, and altitude. Then, they formalized weather, geographical location, altitude, and power usage data as an undirected graph. The best evaluation results achieved 1.04% error rate in predictions when using a GNN with no pooling. Likewise, Khodayar and Wang [126] proposed a GNN-based short-term wind level prediction scheme, in order to increase the output of wind power. Wind speed prediction is still a challenging problem due to stochastic and timely varying properties of wind. In their study, the authors proposed a graph deep learning model to capture the spatial-temporal features of the wind near to wind turbines. Each node of the graph corresponds to a wind site, and a localized first-order approximation of spectral graph convolutions is used to obtain the value of features. Their evaluation results show their proposed scheme outperform other prediction schemes that they considered, such as feed-forward neural networks and nonlinear autoregressive neural networks with respect to both Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) performance metrics.

In addition, Khodayar et al. [127] adopted the GNN model to predict solar irradiance, in order to increase the power generation from photovoltaic systems. In their study, they adopted the deep generative model to capture probability densities for continuous neighboring nodes for a graph. In order to reduce the complexity, their approach involves a scalable generative optimization algorithm to assist the capture process for probability densities. Further, the authors leverage probability densities to generate the convolutional graph autoencoder (CGAE), which is used to forecast solar irradiance. By using real data in the northern states of the U.S., the evaluation results confirm that the investigated scheme has the best performance with respect to reliability, sharpness, and continuously ranked probability score.

**5.4. GNN in Industrial IoT.** Industrial IoT (IIoT) which encompasses multiple areas such as manufacturing, transportation and distribution, and mining and refining of raw materials connects massive numbers of IoT devices, which generate massive amounts of data. Based on the data analysis, IIoT will rely on data collection and analysis to achieve automation and intelligence. From the cyberphysical system perspective, a IIoT system consists of a network subsystem, a

control subsystem, and a computing subsystem. As IIoT systems could generate massive amounts of data and computing resources are generally limited, how to optimize the performance of the network, control, and computing subsystems is a critical issue [128].

There are a number of research efforts on leveraging machine learning to optimize the resource allocation problem in IIoT systems [11, 129–132]. Most of the efforts only focused on Euclidean space datasets, which did not consider system topologies and other related features. Some studies have been leveraging new developments in GNNs to assist resource management. For example, Liu et al. [133] proposed a Dyna-Q (DDQ) approach based on the discrete-time Markov decision process (DTMDP). Since the service requirements are highly dynamic in IIoT systems, the network function virtualization technology receives growing attention. Given limited computing resources in the virtual environment, designing efficient scheduling algorithms is necessary. To this end, the proposed DDQ leverages GNN to predict resource requirements for the virtual network function instance (VNFI) so that the service scheduling can be dynamically reconfigured on the virtualized service chain, leading to the improvement of resource utilization.

In addition, Kim et al. [134] proposed a GNN-based autonomous operation control scheme for IIoT networks. They leveraged GNNs to analyze the behaviors of different devices, based on the relationship and security requirements, and provided autonomous control to ensure both access and security. From the system perspective, Zhang et al. [135] proposed a Graph Neural Network Modeling for IoT (GNNM-IoT) scheme that leverages GNNs to simulate IoT network systems. By leveraging the GNN, the proposed simulator has salient ability to analyze the hidden logical relationships between different domains of massive embedded sensors. Zhang *et al.* used the proposed GNNM-IoT to generate complex nonlinear datasets, achieving better results than long short-term memory (LSTM) and Autoregressive Integrated Moving Average (ARIMA) schemes. Likewise, Protogerou *et al.* [136] proposed a multiagent system based on GNNs to detect attacks against the network. To improve the performance, they formalized active network nodes such as IoT devices, software-defined network forwarders, and fog nodes as a graph to generate the high-dimensional data input. The proposed GNN model is capable of accurately detecting anomalies. To evaluate the proposed GNNs' response to malware attacks, they simulated network flows of various normal and abnormal packet distributions.

## 6. Limitations of Graph Neural Networks

We now review the limitations of GNNs with respect to universality and computing overhead.

**6.1. Universality of GNN.** If a machine learning model can adapt to any input and situation, we consider that the model has Turing universality [137]. Nonetheless, there is no machine learning model that has Turing universality. Similar to GNN, the universality of GNN is one of its limitations. Satisfying some sufficient conditions, GNN can operate on

any input function in the form of a Turing machine and is not limited to the network structure [138]. By establishing the Turing equivalence between GNN and the classic distributed computing model, we can summarize the sufficient conditions [139]: the sufficient depth of layers, sufficient breadth of convolution layers, independent nodes, and accurate expressions for each layer. However, in some cases, it is impossible to obtain accurate mathematical expressions for the model. In addition, adding more layers to a model increases the computing complexity as well. The computing ability of physical devices limits the number of layers of a model. Thus, a number of research efforts leverage matrix approximation, instead of obtaining exact expressions [137]. Thus, how to obtain accurate mathematical expressions for the model while reducing the computing complexity is an important and challenging issue. Furthermore, GNN has low flexibility, transductive, and scalability [140].

**6.2. Computing Overhead of GNN.** As we discussed earlier, because of limited depth and breadth, GNN cannot show its Turing universality and has high computing complexity. Thus, it cannot obtain accurate results when it is applied to some specific datasets. In this case, optimizing the GNN remains an unsolved problem. Related to this issue, Li et al. [141] proposed a learning-based approach based on approximation algorithms and heuristic solvers, in order to reduce the computing complexity. In detail, they leveraged a well-trained graph convolutional network to estimate the likelihood of whether the specific vertex has an optimal solution. By doing so, the GNN significantly increases the search speed for traversing all the vertex in the graph. Then, they adopted a tree search to traverse all the vertex in the graph. The proposed approach can increase search speed in large graphs.

## 7. Final Remarks

Since the complexity of datasets is increasing, how to deal with highly dynamic and dimensional data is a critical issue for machine learning models. In this study, we reviewed the principle of GNNs and existing research efforts. We first introduced the motivation for extending the training data from Euclidean space data to non-Euclidean space data. After that, we introduced two different strategies used to handle non-Euclidean space data: spectral and spatial convolution strategy. We surveyed the existing studies and categorized those studies according to their respective convolution strategies. Next, we reviewed the existing studies on the application of GNNs to emergent IoT systems, including vehicular traffic prediction, electrical energy prediction, and resource management in IIoT systems. Finally, we discussed some limitations of GNNs with respect to universality and computing overhead.

## Data Availability

This article does not cover data research. No data were used to support this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

- [1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [2] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.
- [3] W. G. Hatcher and W. Yu, "A survey of deep learning: platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24411–24432, 2018.
- [4] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks: a survey toward private and secure applications," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, 2021.
- [5] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial internet of things: a cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78238–78259, 2018.
- [6] F. Liang, W. G. Hatcher, W. Liao, W. Gao, and W. Yu, "Machine learning for security and the internet of things: the good, the bad, and the ugly," *IEEE Access*, vol. 7, pp. 158126–158147, 2019.
- [7] H. Xu, X. Liu, W. Yu, D. Griffith, and N. Golmie, "Reinforcement learning-based control and networking co-design for industrial internet of things," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 885–898, 2020.
- [8] K. Li, G. Luo, Y. Ye, W. Li, S. Ji, and Z. Cai, "Adversarial privacy-preserving graph embedding against inference attack," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6904–6915, 2021.
- [9] C. Qian, W. Yu, C. Lu, D. Griffith, and N. Golmie, "Toward generative adversarial networks for the industrial internet of things," *IEEE Internet of Things Journal*, p. 1, 2022.
- [10] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, "Deep learning based inference of private information using embedded sensors in smart devices," *IEEE Network*, vol. 32, no. 4, pp. 8–14, 2018.
- [11] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward edge-based deep learning in industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4329–4341, 2020.
- [12] W. G. Hatcher, C. Qian, W. Gao, F. Liang, K. Hua, and W. Yu, "Towards efficient and intelligent internet of things search engine," *IEEE Access*, vol. 9, pp. 15778–15795, 2021.
- [13] O. Sener and S. Savarese, "Active learning for convolutional neural networks: a core-set approach," 2017, <https://arxiv.org/abs/1708.00489>.
- [14] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.
- [15] Z. Chen, F. Chen, L. Zhang et al., "Bridging the gap between spatial and spectral domains: a survey on graph neural networks," 2020, <https://arxiv.org/abs/2002.11867>.
- [16] R. Sato, "A survey on the expressive power of graph neural networks," 2020, <https://arxiv.org/abs/2003.04078>.
- [17] L. C. Lamb, A. Garcez, M. Gori, M. Prates, P. Avelar, and M. Vardi, "Graph neural networks meet neural-symbolic computing: A survey and perspective," 2020, <https://arxiv.org/abs/2003.00330>.
- [18] M. Balcilar, G. Renton, P. Héroux, B. Gauzere, S. Adam, and P. Honeine, "Bridging the gap between spectral and spatial domains in graph neural networks," 2020, <https://arxiv.org/abs/2003.11702>.
- [19] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *International Conference on Machine Learning*, pp. 874–883, 2020.
- [20] S. Abadal, A. Jain, R. Guirado, J. López-Alonso, and E. Alarcón, "Computing graph neural networks: a survey from algorithms to accelerators," *Computing surveys*, vol. 54, no. 9, pp. 1–38, 2022.
- [21] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2022.
- [22] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2021.
- [23] F. Rosenblatt, *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*, Cornell Aeronautical Lab Inc, Buffalo NY, 1961.
- [24] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.
- [25] T. J. Sejnowski, "Higher-order Boltzmann machines," in *AIP Conference Proceedings*, vol. 151no. 1, pp. 398–403, 1986.
- [26] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, no. 4, pp. 339–356, 1988.
- [27] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Physical Review Letters*, vol. 59, no. 19, pp. 2229–2232, 1987.
- [28] R. McEliece, *The Theory of Information and Coding*, Cambridge University Press, 2009.
- [29] J. S. Long, *Regression Models for Categorical and Limited Dependent Variables*, 1997.
- [30] J. Bromley, I. Guyon, Y. Le Cun, E. Säckinger, and R. Shah, "Signature verification using a Siamese time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 669–688, 1993.
- [31] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.
- [32] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: a survey," 2021, <https://arxiv.org/abs/2101.11174>.
- [33] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.
- [34] X. Zhang, Y. Zou, and W. Shi, "Dilated convolution neural network with LeakyReLU for environmental sound classification," in *2017 22nd International Conference on Digital Signal Processing (DSP)*, pp. 1–5, London, UK, 2017.
- [35] L. Zhou, C. Zhang, and M. Wu, "D-LinkNet: LinkNet with pretrained encoder and dilated convolution for high

- resolution satellite imagery road extraction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 182–186, 2018.
- [36] Y. Wei, H. Xiao, H. Shi, Z. Jie, J. Feng, and T. S. Huang, “Revisiting dilated convolution: a simple approach for weakly- and semi-supervised semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7268–7277, 2018.
- [37] 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 Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.
- [38] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” 2013, <https://arxiv.org/abs/1312.6203>.
- [39] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” 2016, <https://arxiv.org/abs/1606.09375>.
- [40] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.
- [41] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2016, <https://arxiv.org/abs/1609.02907>.
- [42] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *International Conference on Machine Learning*, pp. 1263–1272, 2017.
- [43] J. Atwood and D. Towsley, “Diffusion-convolutional neural networks,” 2015, <https://arxiv.org/abs/1511.02136>.
- [44] M. Niepert, M. Ahmed, and K. Kutzkov, “Learning convolutional neural networks for graphs,” in *International conference on machine learning*, pp. 2014–2023, 2016.
- [45] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, “Molecular graph convolutions: moving beyond fingerprints,” *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, 2016.
- [46] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” 2018, <https://arxiv.org/abs/1806.08804>.
- [47] L. Yi, H. Su, X. Guo, and L. J. Guibas, “SyncSpecCNN: synchronized spectral CNN for 3D shape segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2282–2290, 2017.
- [48] J. Kunegis and A. Lommatzsch, “Learning spectral graph transformations for link prediction,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 561–568, 2009.
- [49] A. D. Perkins and M. A. Langston, “Threshold selection in gene coexpression networks using spectral graph theory techniques,” in *BMC bioinformatics*, vol. 10, no. 11pp. 1–11, BioMed Central, 2009.
- [50] X.-H. Han, B. Shi, and Y. Zheng, “SSF-CNN: spatial and spectral fusion with CNN for hyperspectral image super-resolution,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 2506–2510, Athens, Greece, 2018.
- [51] M. Edwards and X. Xie, “Graph based convolutional neural network,” 2016, <https://arxiv.org/abs/1609.08965>.
- [52] S. Parisot, S. I. Ktena, E. Ferrante et al., “Spectral graph convolutions for population-based disease prediction,” in *International conference on medical image computing and computer-assisted intervention*, pp. 177–185, Cham, 2017.
- [53] P. Tripathi and P. N. Pandey, “A novel alignment-free method to classify protein folding types by combining spectral graph clustering with Chou’s pseudo amino acid composition,” *Journal of Theoretical Biology*, vol. 424, pp. 49–54, 2017.
- [54] R. Levie, W. Huang, L. Bucci, M. M. Bronstein, and G. Kutyniok, “Transferability of spectral graph convolutional neural networks,” 2019, <https://arxiv.org/abs/1907.12972>.
- [55] C. Wang, B. Samari, and K. Siddiqi, “Local spectral graph convolution for point set feature learning,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 52–66, 2018.
- [56] E. Rossi, F. Frasca, B. Chamberlain, D. Eynard, M. Bronstein, and F. Monti, “Sign: scalable inception graph neural networks,” 2020, <https://arxiv.org/abs/2004.11198>.
- [57] A. Salim, “Framework for designing filters of spectral graph convolutional neural networks in the context of regularization theory,” 2020, <https://arxiv.org/abs/2009.13801>.
- [58] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, “Pitfalls of graph neural network evaluation,” 2018, <https://arxiv.org/abs/1811.05868>.
- [59] R. Li, S. Wang, F. Zhu, and J. Huang, “Adaptive graph convolutional neural networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018no. 1.
- [60] S. Luan, M. Zhao, X.-W. Chang, and D. Precup, “Break the ceiling: stronger multi-scale deep graph convolutional networks,” 2019, <https://arxiv.org/abs/1906.02174>.
- [61] L. Gong and Q. Cheng, “Exploiting edge features for graph neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9211–9219, 2019.
- [62] S. I. Ktena, S. Parisot, E. Ferrante et al., “Distance metric learning using graph convolutional networks: application to functional brain networks,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 469–477, 2017.
- [63] L. Huang, D. Ma, S. Li, X. Zhang, and H. Wang, “Text level graph neural network for text classification,” 2019, <https://arxiv.org/abs/1910.02356>.
- [64] J. Kim, T. Kim, S. Kim, and C. D. Yoo, “Edge-labeling graph neural network for few-shot learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11–20, 2019.
- [65] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, “Graph wavelet neural network,” 2019, <https://arxiv.org/abs/1904.07785>.
- [66] C. Li, Z. Cui, W. Zheng, C. Xu, R. Ji, and J. Yang, “Action-attending graphic neural network,” *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3657–3670, 2018.
- [67] D. Cao, Y. Wang, J. Duan et al., “Spectral temporal graph neural network for multivariate time-series forecasting,” 2021, <https://arxiv.org/abs/2103.07719>.
- [68] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang, “Transferring robustness for graph neural network against poisoning attacks,” in *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 600–608, 2020.

- [69] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary, "Quantum graph neural networks," 2019, <https://arxiv.org/abs/1909.12264>.
- [70] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," 2017, <https://arxiv.org/abs/1704.06803>.
- [71] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," 2017, <https://arxiv.org/abs/1711.04043>.
- [72] L. Ruiz, L. F. Chamon, and A. Ribeiro, "Graphon neural networks and the transferability of graph neural networks," 2020, <https://arxiv.org/abs/2006.03548>.
- [73] Z. Chen, X. Li, and J. Bruna, "Supervised community detection with line graph neural networks," 2017, <https://arxiv.org/abs/1705.08415>.
- [74] S. Yuan, X. Wu, J. Li, and A. Lu, "Spectrum-based deep neural networks for fraud detection," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 2419–2422, 2017.
- [75] A. Egozi, Y. Keller, and H. Guterman, "A probabilistic approach to spectral graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 18–27, 2013.
- [76] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: a deep learning framework for network-scale traffic learning and forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4883–4894, 2019.
- [77] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional ARMA filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 1–3507, 2021.
- [78] R. Yin, K. Li, G. Zhang, and J. Lu, "A deeper graph neural network for recommender systems," *Knowledge-Based Systems*, vol. 185, article 105020, 2019.
- [79] S.-G. Huang, M. K. Chung, A. Qiu, and A. D. N. Initiative, "Revisiting convolutional neural network on graphs with polynomial approximations of Laplace-Beltrami spectral filtering," 2020, <https://arxiv.org/abs/2010.13269>.
- [80] G. Fu, Y. Hou, J. Zhang, K. Ma, B. F. Kamhoua, and J. Cheng, "Understanding graph neural networks from graph signal denoising perspectives," 2020, <https://arxiv.org/abs/2006.04386>.
- [81] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Direct multi-hop attention based graph neural network," 2020, <https://arxiv.org/abs/2009.14332>.
- [82] X. Li, D. Hu, and F. Nie, "Large graph hashing with spectral rotation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017no. 1.
- [83] C. Jin, S. Z. Chen, and H. H. He, "Classifying cosmic-ray proton and light groups in LHAASO-KM2A experiment with graph neural network," *Chinese Physics C*, vol. 44, no. 6, article 065002, 2020.
- [84] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayley-nets: graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2019.
- [85] M. Ferrer, F. Serratos, and A. Sanfeliu, "Synthesis of median spectral graph," in *Iberian Conference on Pattern Recognition and Image Analysis*, pp. 139–146, Berlin, Heidelberg, 2005.
- [86] H. Nt and T. Maehara, "Revisiting graph neural networks: all we have is low-pass filters," 2019, <https://arxiv.org/abs/1905.09550>.
- [87] T. Cai, S. Luo, K. Xu, D. He, T.-y. Liu, and L. Wang, "Graph-norm: a principled approach to accelerating graph neural network training," 2020, <https://arxiv.org/abs/2009.03294>.
- [88] Y. Wang, X. Wu, and L. Wu, "Differential privacy preserving spectral graph analysis," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 329–340, Berlin, Heidelberg, 2013.
- [89] R. Li and J. Huang, "Learning graph while training: an evolving graph convolutional neural network," 2017, <https://arxiv.org/abs/1708.04675>.
- [90] E. Tam and D. Dunson, "Fiedler regularization: learning neural networks with graph sparsity," in *International Conference on Machine Learning*, pp. 9346–9355, 2020.
- [91] Z. Liu, C. Chen, L. Li et al., "GeniePath: graph neural networks with adaptive receptive paths," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 4424–4431, 2019.
- [92] X. Wang, H. Ji, C. Shi et al., "Heterogeneous graph attention network," in *The World Wide Web Conference*, pp. 2022–2032, 2019.
- [93] Y. Hechtlinger, P. Chakravarti, and J. Qin, "A generalization of convolutional neural networks to graph-structured data," 2017, <https://arxiv.org/abs/1704.08165>.
- [94] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2017, <https://arxiv.org/abs/1706.02216>.
- [95] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, <https://arxiv.org/abs/1710.10903>.
- [96] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, vol. 30no. 1, p. 3, 2013.
- [97] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [98] J. Bruna and X. Li, "Community detection with graph neural networks," *Stat*, vol. 1050, p. 27, 2017.
- [99] Q. Zhao, Z. Ye, C. Chen, and Y. Wang, "Persistence enhanced graph neural network," in *International Conference on Artificial Intelligence and Statistics*, pp. 2896–2906, 2020.
- [100] R. Wang, D. D. Nguyen, and G.-W. Wei, "Persistent spectral graph," *International journal for numerical methods in biomedical engineering*, vol. 36, no. 9, article e3376, 2020.
- [101] J. Zhou, G. Cui, Z. Zhang et al., "Graph neural networks: a review of methods and applications," 2018, <https://arxiv.org/abs/1812.08434>.
- [102] S. Casas, C. Gulino, R. Liao, and R. Urtasun, "Spatially-aware graph neural networks for relational behavior forecasting from sensor data," 2019, <https://arxiv.org/abs/1910.08233>.
- [103] X. Wang, Y. Ma, Y. Wang et al., "Traffic flow prediction via spatial temporal graph neural network," in *Proceedings of The Web Conference 2020*, pp. 1082–1092, 2020.
- [104] D. Kong, H. Ma, and X. Xie, *SIA-GCN: a spatial information aware graph neural network with 2d convolutions for hand pose estimation*, 2020.



- [105] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Thirty-first AAAI conference on artificial intelligence*, vol. 31, 2017, <https://ojs.aaai.org/index.php/AAAI/article/view/10735>.
- [106] X. Zhou, Y. Shen, Y. Zhu, and L. Huang, "Predicting multi-step citywide passenger demands using attention-based neural networks," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 736–744, 2018.
- [107] H. Yao, F. Wu, J. Ke et al., "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018, <https://ojs.aaai.org/index.php/AAAI/article/view/11836>.
- [108] Z. Xie, W. Lv, S. Huang, Z. Lu, B. Du, and R. Huang, "Sequential graph neural network for urban road traffic speed prediction," *IEEE Access*, vol. 8, pp. 63349–63358, 2020.
- [109] D. Cheng, X. Wang, Y. Zhang, and L. Zhang, "Graph neural network for fraud detection via spatial-temporal attention," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3800–3813, 2022.
- [110] H. Peng, H. Wang, B. Du et al., "Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting," *Information Sciences*, vol. 521, pp. 277–290, 2020.
- [111] W. Zhang, H. Liu, Y. Liu, J. Zhou, and H. Xiong, "Semi-supervised hierarchical recurrent graph neural network for city-wide parking availability prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34no. 1, pp. 1186–1193, 2020, <https://ojs.aaai.org/index.php/AAAI/article/view/5471>.
- [112] F. Diehl, T. Brunner, M. T. Le, and A. Knoll, "Graph neural networks for modelling traffic participant interaction," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 695–701, Paris, France, 2019.
- [113] S.-Y. Chen, Q. Gao-Feng, X.-H. Wang, and H.-Z. Zhang, "Traffic flow forecasting based on grey neural network model," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, vol. 2, pp. 1275–1278, Xi'an, China, 2003.
- [114] X. Hu, C. Zhao, and G. Wang, "A traffic light dynamic control algorithm with deep reinforcement learning based on GNN prediction," 2020, <https://arxiv.org/abs/2009.14627>.
- [115] S. Du, T. Li, X. Gong, and S.-J. Horng, "A hybrid method for traffic flow forecasting using multimodal deep learning," 2019, <https://arxiv.org/abs/1803.02099>.
- [116] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
- [117] Y. Jia, W. Jianping, and D. Yiman, "Traffic speed prediction using deep learning method," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1217–1222, Rio de Janeiro, Brazil, 2016.
- [118] A. Koesdwiady, R. Souza, and F. Karray, "Improving traffic flow prediction with weather information in connected cars: a deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9508–9517, 2016.
- [119] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.
- [120] B. Wang, X. Luo, F. Zhang, B. Yuan, A. L. Bertozzi, and P. J. Brantingham, "Graph-based deep modeling and real time forecasting of sparse spatio-temporal data," 2018, <https://arxiv.org/abs/1804.00684>.
- [121] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018.
- [122] D. Chai, L. Wang, and Q. Yang, "Bike flow prediction with multi-graph convolutional networks," in *Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems*, pp. 397–400, 2018.
- [123] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," 2019, <https://arxiv.org/abs/1906.00121>.
- [124] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 922–929, 2019.
- [125] D. Owerko, F. Gama, and A. Ribeiro, "Predicting power outages using graph neural networks," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 743–747, Anaheim, CA, USA, 2018.
- [126] M. Khodayar and J. Wang, "Spatio-temporal graph deep neural network for short-term wind speed forecasting," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 2, pp. 670–681, 2019.
- [127] M. Khodayar, S. Mohammadi, M. E. Khodayar, J. Wang, and G. Liu, "Convolutional graph autoencoder: a generative deep neural network for probabilistic spatio-temporal solar irradiance forecasting," *IEEE Transactions on Sustainable Energy*, vol. 11, no. 2, pp. 571–583, 2020.
- [128] W. Yu, F. Liang, X. He et al., "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.
- [129] F. Liang, C. Qian, W. G. Hatcher, and W. Yu, "Search engine for the internet of things: lessons from web search, vision, and opportunities," *IEEE Access*, vol. 7, pp. 104673–104691, 2019.
- [130] F. Liang, W. G. Hatcher, G. Xu, J. Nguyen, W. Liao, and W. Yu, "Towards online deep learning-based energy forecasting," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, Valencia, Spain, 2019.
- [131] X. Liu, W. Yu, F. Liang, D. Griffith, and N. Golmie, "Deep learning in security of Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12163–12175, 2021.
- [132] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Towards computing resource reservation scheduling in industrial internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8210–8222, 2020.
- [133] Y. Liu, Y. Lu, X. Li, Z. Yao, and D. Zhao, "On dynamic service function chain reconfiguration in IoT networks," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10969–10984, 2020.
- [134] J.-H. Kim, S. Lee, and S. Hong, "Autonomous operation control of IoT blockchain networks," *Electronics*, vol. 10, no. 2, p. 204, 2021.
- [135] W. Zhang, Y. Zhang, L. Xu et al., "Modeling IoT equipment with graph neural networks," *IEEE Access*, vol. 7, pp. 32754–32764, 2019.
- [136] A. Protogerou, S. Papadopoulos, A. Drosou, D. Tzovaras, and I. Refanidis, "A graph neural network method for distributed

- anomaly detection in IoT,” *Evolving Systems*, vol. 12, pp. 19–36, 2021.
- [137] P. Rendell, *Turing Machine Universality of the Game of Life*, Springer, 2016.
- [138] C. Petzold, *The Annotated Turing: a Guided Tour through Alan Turing’s Historic Paper on Computability and the Turing Machine*, Wiley Publishing, 2008.
- [139] T. Landelius, *Reinforcement Learning and Distributed Local Model Synthesis*, Ph.D. dissertation, Linkoping University Electronic Press, 1997.
- [140] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, “Session-based social recommendation via dynamic graph attention networks,” in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 555–563, 2019.
- [141] Z. Li, Q. Chen, and V. Koltun, “Combinatorial optimization with graph convolutional networks and guided tree search,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.