A Survey on Graph Neural Network

With Focus on Recurrent & Convolution Operations

Northwestern University, Evanston, IL, USA
Jiaqi Guo, MS*

Abstract

Thanks to CNN's ability to effectively capture the hidden pattern in Euclidean data, the emergence of deep learning technology has brought dramatic progress to many fields, from computer vision to natural language processing. However, With the increase of application scenarios, more and more data that cannot be represented in Euclidean space has imposed great challenges to the existing machine learning frameworks. On the one hand, this is because most of the data, in reality, is non-Euclidean, on the other hand, the existing machine learning algorithms are not very effective at dealing with the topological dependency of information on each object in non-Euclidean data. More recently, to extend the excellent performance of neural networks to the non-Euclidean domain, researchers have proposed an innovative machine learning framework to learn the latent representation of graphs, the Graph Neural Networks (GNNs).

Keywords: Deep learning with graphs; Graph neural network; Message passing; Convolution/Recurrent operation on graph

1 Introduction

Compared to conventional machine learning, which heavily relies on handcrafted feature engineering, various network paradigms, e.g., Convolutional Neural Networks (CNNs) [1], Recurrent Neural Networks (RNNs) [2], and auto-encoders [3], [4] in deep learning can effectively extract hidden features from data with grid-like structure. Based on such a particular property, deep learning techniques have been widely employed in various tasks of pattern recognition and data mining, such as image classification [5], [6], speech recognition [7], and object detection [8]. Typically, the data used in these tasks are represented in the Euclidean domain, which includes image, volume, video that lie on 2D, 3D Euclidean domain, and

sentence, word, sound that lies on 1D Euclidean domain. Take the image as an example, its nature of shift-invariance ensures that for any pixel (node) in the image, the number of neighbor pixels (nodes) is fixed and determined. Therefore, it is straightforward to perform convolution operations on images to extract their hidden patterns.

Recent studies tend to represent non-Euclidean data in the form of graphs, such a generalized data structure can be applied to most forms of data. For example, in the field of biology, we can model the protein as a graph by letting the amino acid residues on the protein surface be nodes [9], [10]. And in social network analysis, users, as an independent node, are connected through their social relationships to form a graph-liked social network [11], [12].

However, unlike data subject to Euclidean distribution, for any node in a graph (non-Euclidean distribution), the number of neighbors may be different, even dynamic, resulting in some basic operations (e.g., pooling and convolution) that are easy to implement in Euclidean space, but difficult to reproduce in the non-Euclidean (graph) domain. Therefore, extending the deep neural networks to the non-Euclidean domain has become an emerging research field that receives enormous attention.

One of the motivations comes from the term of *graph representation learning* [13], [14], which uses a set of low-dimensional basis vectors to represent the complex graph structure like nodes, edges, and subgraphs. In 2014, inspired by the idea of word embedding from *word2vec* [15], [16] and *graph representation learning*, Perozzi and his team proposed a novel node embedding algorithm, *DeepWalk* [17], which uses local information obtained from truncated *random walks* to learn latent representations by treating nodes as equivalents of words and node sequences as equivalents of sentences. The later study *node2vec* [18] improves the neighborhood sampling strategy in the *DeepWalk* algorithm by developing a flexible biased random walk procedure. To be more specific, consider a random walk traversed edge (t, v), and now resides at node v. In *node2vec*, the transition probabilities are no longer random, but follows the equation (1):

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx} \tag{1}$$

where, π_{vx} is the unnormalized transition probabilities on edge (v, x) leading from v, d_{tx} denotes the geodesic distance between node t and x, which is limited to $\{0, 1, 2\}$. And the bias term α_{pq} is defined with two

parameter p and q as equation (2):

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

As the first two works that put forward the concept of "embedding", *DeepWalk* and *node2vec* have laid the theoretical foundation for subsequent studies and point out the potential research direction, but they have two severe drawbacks [19]. These direct embedding methods do not possess the property of permutation invariance and lack the ability of generalization, suggesting the poor capability to be transferred to other new graph-structured data. Another problem arises from the "isolation" of nodes, as no parameters are shared between nodes in the encoder, resulting in inefficient computation.

After the success of *DeepWalk* and *node2vec*, multiple Graph Neural Network structures have been proposed to solve a wide variety of real-world problems with great success in many areas. In this review, we will detail these new GNN models and their applications, with emphasis on the convolution and recurrent operations. Our contribution is summarized as below:

- We summarize the basic structure of the general graph neural network in detail, classify the components of each structure and discuss each variant.
- We divide various GNN networks into three categories according to their corresponding propagation modules. It includes *the Recurrent Operator*, *the Convolution Operator*, and *the Skipping Block*.For the recurrent operation and convolution operation, we have carried out a detailed mathematical analysis of each module and listed representative examples.
- We have extensively collected the various applications of GNN, classified them according to the applied disciplines, and listed the state-of-art achievements of network models in each field.
- We analyze the limitations of existing networks and suggest three prospective research directions.

1.1 Organization of Our Article

The rest of this article is organized as follows: Section 2 provides an overview of the troubleshooting procedures of solving a common GNN problem and defines the commonly used graph concepts. Section 3

conducts comprehensive analysis towards several benchmark GNN models and provide their classification. Section 4 lists the major application of GNNs across various disciplines. Section 5 points out the existing limitations and suggests the future research directions.

2 General Troubleshooting Procedures for GNN Problems

In this section, we will first define the graph-related conceptions and address commonly used notations. Then, we will introduce several training-related factors that determine the choice of the loss function in the network. Finally, we'll give an overview of the general structure of GNN and delve into the composition of a single GNN layer.

2.1 Define the Graph

Define a basic graph: A graph can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, while $\mathcal{E}$ is the set edges. To represent a directed edge between node $v_i, v_j \in \mathcal{V}$, we use $e_{ij} = (v_i, v_j) \in \mathcal{E}$ to denote a directed edge lead from node v_i to v_j. The adjacent matrix is defined as $A \in \mathbb{R}^{n \times n}$, where n is the number of nodes in the graph. Depending on the type of graph, a node v may have feature vector $\mathbf{x}_v \in \mathbb{R}^d$ store in the node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. And a edge e pointing from node v to node u may also have a feature vector $\mathbf{x}^e_{(v,u)}$ which is stored in the edge feature matrix $\mathbf{X}^e \in \mathbb{R}^{m \times c}$. Other commonly used notations are listed below:

Classification of Graph: The complexity of a graph often depends on the types of nodes and edges, and a graph with complex types generally contains more information. We usually classify all graphs into the following types:

- **Directed/Undirected Graphs**: A directed graph is a graph with all edges directed from one node to another, which means information can only propagate in the direction that the edge point to, which provide more information. An undirected graph can be transformed into a directed graph by treating each undirected edge as two opposite directed edges.
- **Homogeneous/Heterogeneous Graphs:** Compared to the homogeneous graph, nodes and edges in heterogeneous graphs may have different types, this makes heterogeneous graphs have powerful description ability and can provide even more information but at the cost of higher computational complexity.

Table 1: Commonly Used Notation List

Notations	Descriptions
$\mathbf{a}, \mathbf{A}$	Vector and matrix (lowercase\uppercase characters)
$\mathbb{R}^N$	N-dimensional Euclidean space
$\mathbf{I}_N$	Identity matrix of dimension N
n, m	The number of nodes and edges
d, c	The dimension of the node and edge feature vector
$\mathbf{D}$	The degree matrix ($\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$)
$\mathbf{L}$	The Laplacian matrix ($\mathbf{L} = \mathbf{D} - \mathbf{A}$)
$\mathbf{L^{sys}}$	The symmetric normalized Laplacian matrix ($\mathbf{L^{sys}} = \mathbf{I_n} - \mathbf{D^{(-1/2)}LD^{(-1/2)}}$)
$\mathbf{A}^T$	The transpose of the matrix A
$\mathbf{g} * \mathbf{x}$	Convolution of $\mathbf{g}$ and $\mathbf{x}$
$\mathcal{N}_v$	The neighborhood nodes set of node v
$\mathbf{a}_v^t$	Vector a of node v at time step t
$\mathbf{h}_v^{(k)}$	Embedding of node v at k-th GNN layer
$\mathbf{X}^e \in \mathbb{R}^{m \times c}$	The edge feature mat (v, u)
$\mathbf{x}^e_{(v,u)} \in \mathbf{R}^c$	The edge feature vector of the edge (v, u)
$\mathbf{W}, \mathbf{V}, \Theta, w, \epsilon$	Learnable parameters
$\phi(\cdot)$	This can be any nonlinear activation function
$\sigma(\cdot)$	The Sigmoid activation function
$\odot$	Element-wise product
$\|$	Vector concatenation

- **Dynamic/Statistic Graphs:** Graphs are considered as dynamic graphs only if node attributes change over time.

In practice, the actual input to the network is usually a combination of the above types.

2.2 Training Frameworks

Once we specify the graph types, we need to determine the training framework, which is directly related to the rest of the training settings. For example, given a graph with nodes being partially labeled, the research problem will be using the GNN to learn a robust model that can classify the rest of the nodes. This is a typical node-level semi-supervised classification problem, and [20] had proposed an end-to-end framework using a stacking of graph convolutional layers followed by a *softmax* [21] for multiclass classification.

Depending on the scope of tasks and the availability of data labels, we categorize the graph learning tasks into:

- **Node-level:** It generally include the node classification [20], node regression [22], and node clustering, that is, to identify the label of a particular node or assign a continuous value to each existing node.
- **Edge-level:** It generally focus on the edge classification [23] and link prediction [24], [25], that is, to identify the label of a particular set of edges or simply determine the existence of an edge between a pair of nodes.
- **Graph-level:** It generally focus on the graph classification [26], [27], graph regression [22], and graph matching [28]. Among them, graph classification, and graph regression are similar to the tasks at node-level, and graph matching is the problem of finding the similarity between graphs.

From the perspective of availability of data label, we can also divide our learning tasks into three situations:

- **Supervised Learning:** All data are fully labeled.
- **Semi-supervised Learning:** Data are partially labeled.
- **Unsupervised Learning:** All data are unlabeled, a typical unsupervised problem is the node-level regression problem.

After determining the graph type and training setting, we can finalize the choice of the loss function. The next step will be designing the inner structure of a single GNN layer.

2.3 General GNNs Structure

Zhou et al. [29] summarized the general structure of the GNN model as shown in Figure 1. They divided GNN into four parts, from left to right:

- **The graph-structured data input**, which is the combination of graph types mentioned in Sec 2.1.
- **The stacking of K GNN layers**, which may determine the size reception field K in graph convolution.
- **The output graph embedding**, which can also be called state or representation.
- **The task-specified part**, determined by the factors mentioned in Sec 2.2.

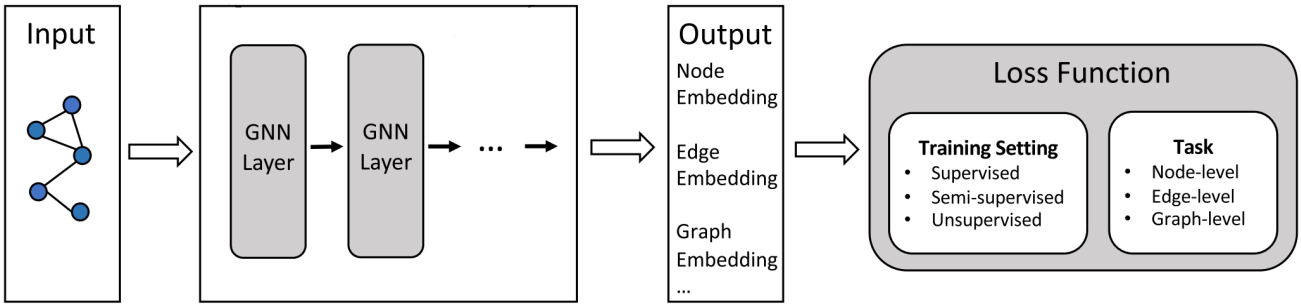

Figure 1: The general structure of the GNN model proposed by Zhou et al.

As the core structure of GNN, the GNN layers play a similar role to the convolutional layer in CNNs. For example, in a simple node-level classification problem, the output node embedding will be fed to the fully connected layers to finish the final classification. In the later sections, we will provide more details about a single GNN layer, which will include its structure and fundamental working principle.

2.4 A Single GNN Layer

The GNN layer is designed to obtain the embedding information of the graph. A typical GNN layer is shown in Figure 2. In this article, we mainly focus on some commonly used modules, which are:

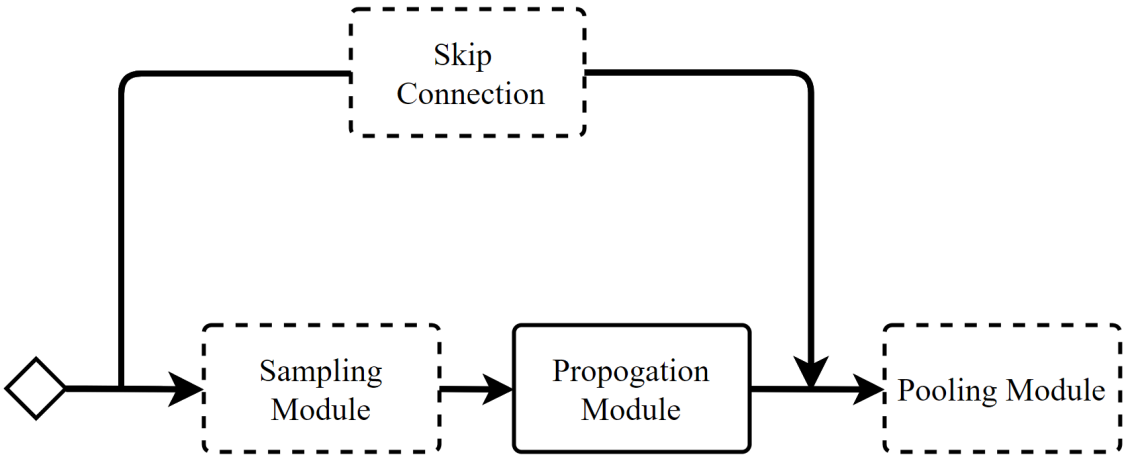

Figure 2: Commonly used modules and their structure in a single GNN layer

Propagation Module: The propagation module is the most important and indispensable component of the entire GNN structure, which is used to propagate information between nodes. In the propagation module, *the convolution* and *recurrent operation* are used to finish the message aggregation and more detail relevant to those two operations will be covered in the following articles.

Pooling Module: Similar to the pooling layer in conventional CNN structure which can capture the general features of input euclidean data. The pooling module is employed to extract the general features from the hierarchy structures inside some complicated and large-scale graphs. Some typical examples are:

Table 2: Popular Graph-based pooling approaches

Pooling Modules	Relevant Publication
Direct Pooling	[30], [31]
Hierarchical Pooling	[32]–[34]

Sampling Module: As a single GNN layer can aggregate information for each node from its neighbors. Therefore, for a K layers GNN, each node in the graph will have a K-hop receptive field. If we increase the number of hops K, the size of the neighborhood will grow exponentially. The sampling module is designed to handle the issue of "neighborhood explosion [29]". More examples can be found in the table below:

Table 3: Popular Graph-based sampling approaches

Sampling Modules	Relevant Publication
Node Sampling	[35]–[37]
Sub-graph Sampling	[38]–[40]
Layer Sampling	[14], [41]

Skip Connection: As we mentioned above, as the depth K of GNN layers increases, the number of nodes in each node's receptive field will also increase exponentially. This not only causes the problem of "neighborhood explosion", but also leads to the overlapping of receptive fields between different nodes, resulting in the "over-smoothing" problem [42]. To be more specific, as the number K increase to infinity, all nodes embedding will eventually converge to a fixed point, which makes the nodes indistinguishable in classification problem.

The skip connection, inspired by the ResNet [6], [43], can increase the overall impact of earlier layers on the final node embedding (Figure 3) and therefore can effectively alleviate the problem of over-smoothing.

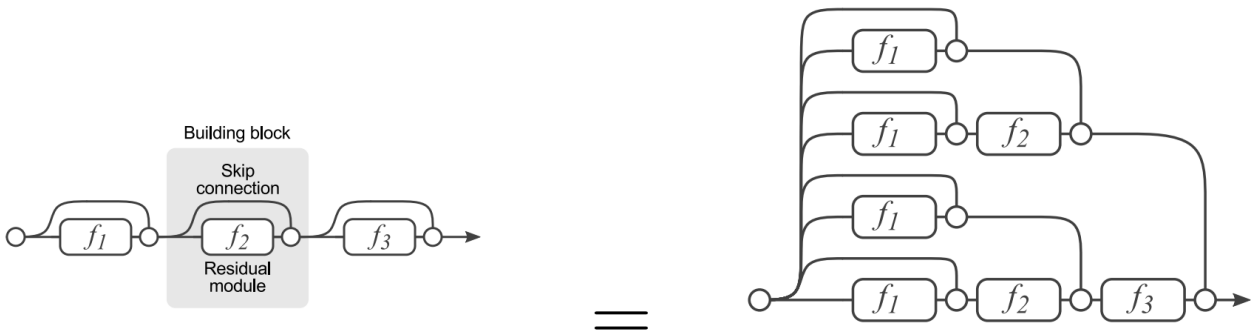

Figure 3: With y_{i-1} as its input, the output y of the i-th Residual block is defined as $y_i \equiv f_i(y_{i-1}) + y_{i-1}$, where f_i is some sequence of operations (e.g., convolution, batch normalization and nonlinear activation function). The left figure shows a conventional 3-block residual network while the right figure shows its *unraveled view* [44]

3 Propagation Modules in GNN

In this section, we will introduce the propagation module in the GNN layer by providing a detailed analysis and comparison of some typical instantiations. We will start our introduction from the recurrent operation in section 3.1, and then introduce the convolution operator in section 3.2 following a order that those methods proposed.

3.1 Recurrent Method

As the pioneer works using to tackled the graph-structured data, the basic idea of Recurrent Graph Neural Networks (RecGNNs) is to get the high-level graph representation by applying the same set of parameters recurrently over nodes in the graph. In this article, we will divided the RecGNNs into two categories: convergence-based approaches and gated approaches.

3.1.1 Convergence-based Approaches

This kind of approach is based on the information diffusion mechanism. The graph will be fed to an encoding network containing N units, where the encoding network is a recurrent network and N correspond to the number of nodes in the graph. Those units in the encoding network will continuously update their representation by information exchanging until they are reaching a convergence state.

Graph Neural Network (GNN*) [45], [46] proposed by Scarselli et al. extend *Recursive neural networks* [47]–[49] and *Markov chains* [50], [51] to process a wider range of graphs (e.g., cyclic, directed, and undirected graphs). Note here we replace the name GNN with GNN* to avoid the ambiguity. A simple

implementation of GNN* is shown in figure 4. To unify the representation of parameters, in this article, we will replace $\mathbf{l}_v$ with $\mathbf{x}_v$, $\mathbf{l}_{(n_v,n_u)}$ with $\mathbf{x}^e_{(v,u)}$ and $\mathbf{x}_v$ with $\mathbf{h}_v$. Then, the computation step of $\mathbf{h}_v$ and $\mathbf{o}_v$ will be:

$$\begin{aligned}\mathbf{h}_v^{(t)} &= f_w\left(\mathbf{x}_v, \mathbf{x}^e_{(v,u)}, \mathbf{x}_u, \mathbf{h}_u^{(t-1)}\right)\\ \mathbf{o}_v^{(t)} &= g_w\left(\mathbf{h}_v^{(t)}, \mathbf{x}_v\right)\end{aligned} \tag{3}$$

Where $\mathbf{h}_v^{(0)}$ is initialized randomly. Experiments have shown that the GNN model can quickly converge and achieve good performance in modeling structural data. However, there are still two drawbacks to this approach. As GNN* requires f_w to be *a contraction map* to ensure the convergence, it is inefficient to update h_v to a fixed point. The second problem is that when dealing with some node-level tasks because all nodes converge to their corresponding fixed points, the problem of over-smoothing may occur.

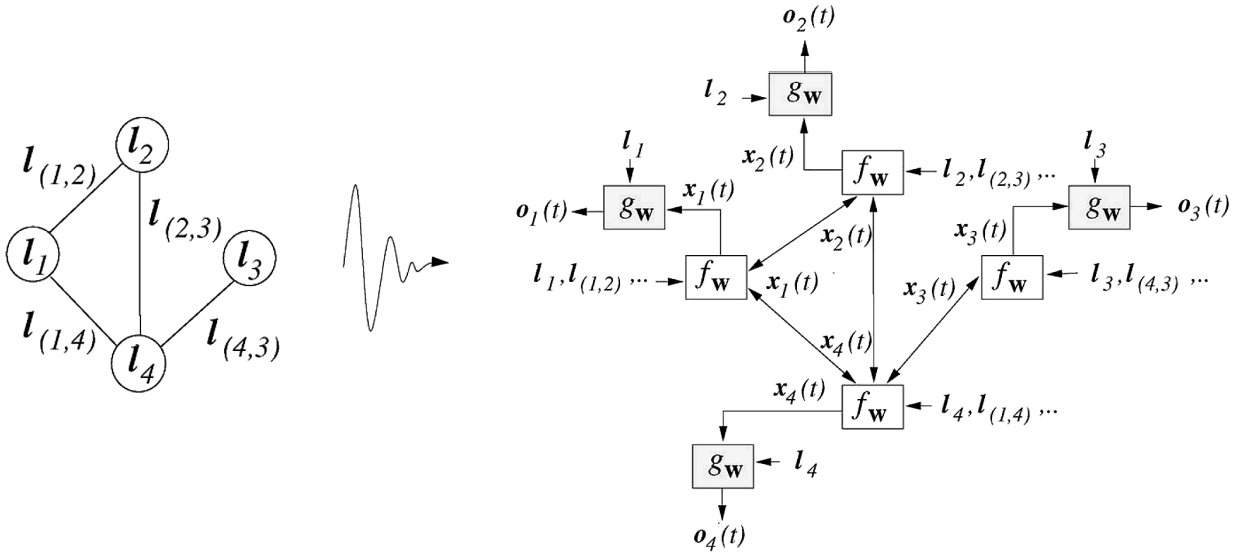

Figure 4: In [45], the nodes in graph (left figure) are replaced by units f_w (*local transition function*) and g_w (*local output function*) in the right figure, which are both the feed-forward neural networks. Here, l_v denotes the node feature vector of node v whereas $l_{(n_v,n_u)}$ is the edge feature vector of edge (n_v, n_u), $x_v(t)$ and $o_v(t)$ are respectively the output and the state of node v at iteration t.

Stochastic steady-state embedding(SSE) [52] proposed a two-step asynchronous method to update the node embedding. In each iteration, it will sample nodes twice, which will be used for node state update and gradient computation respectively. For updating node state, node v are sampled from the entire dataset $\mathcal{V}$, whereas for gradient computation, node v is sampled from the labeled nodes set $\mathcal{V}^{(y)}$. To ensure stability,

SSE uses moving averages to weight historical and new states, which can be expressed as:

$$\mathbf{h}_v^{(t)} = (1-\alpha)\mathbf{h}_v^{(t-1)} + \alpha \mathbf{W_1} \sigma\left(\mathbf{W_2}\left[\mathbf{x}_v, \sum_{u \in \mathcal{N}v}\left[\mathbf{h}_u^{(t-1)}, \mathbf{x}_u\right]\right]\right) \tag{4}$$

Here, $\mathbf{W}_2$ and $\mathbf{W}_1$ is the parameter of the recurrent operation, $\mathbf{h}_v^{(0)}$ is initialized randomly. As for the local output function g_w, ESS adopted a two-layer neural network, which can be expressed as:

$$\mathbf{o}_v^{(t)} = \sigma\left(V_2^\top \operatorname{ReLU}\left(V_1^\top \mathbf{h}_v^{(t)}\right)\right) \tag{5}$$

In this equation, $\mathbf{V}_1$ and $\mathbf{V}_2$ are two learnable parameters of $g(\cdot)$. Since these two steps are stochastic and only require the 1-hop neighborhood for the state update, which reduces the computational complexity of parameter updating in each iteration and enhances the efficiency of training compared with GNN* and its variants.

3.1.2 Gated Approaches

Recent studies attempt to remove the need to constrain parameters by introducing gate mechanisms such as *Gated Recurrent Units (GRU)* [53] and *Long short-term memory (LSTM)* in the propagation step. They run a fixed number of training steps T without ensuring the convergence of the node state.

Gated graph sequence neural networks (GGNN) [54] introduced *GRU* as its recurrent function, which makes the local transition function f_w no longer have to be a contraction map. The aggregated message $\mathbf{m}_v^t$ from v's neighbor nodes can be written as:

$$\mathbf{m}_v^t = \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{t-1} + \mathbf{b} \tag{6}$$

The next step is to update the hidden state of each node using the aggregated information from the neighbor nodes and the information from the previous time step:

$$
\begin{aligned}
\textbf{Update Gate: } \mathbf{z}_v^t &= \sigma\left(\mathbf{W}^z \mathbf{m}_v^t + \mathbf{U}^z \mathbf{h}_v^{t-1}\right) \\
\textbf{Reset Gate: } \mathbf{r}_v^t &= \sigma\left(\mathbf{W}^r \mathbf{m}_v^t + \mathbf{U}^r \mathbf{h}_v^{t-1}\right) \\
\textbf{Current Memory: } \tilde{\mathbf{h}}_v^t &= \tanh\left(\mathbf{W}\mathbf{m}_v^t + \mathbf{U}\left(\mathbf{r}_v^t \odot \mathbf{h}_v^{t-1}\right)\right) \\
\textbf{Final Memory: } \mathbf{h}_v^t &= \left(1 - \mathbf{z}_v^t\right) \odot \mathbf{h}_v^{t-1} + \mathbf{z}_v^t \odot \tilde{\mathbf{h}}_v^t
\end{aligned}
\tag{7}
$$

Where $\mathbf{h}_v^{(1)} = \mathbf{x}_v$ and $\mathbf{W}$ and $\mathbf{U}$ are the learnable parameter. However, there is still a problem with this approach. *The backpropagation through time* algorithm requires a lot of memory resources during gradients computation, so GGNN cannot be implemented on some large graphs.

Similar to *GRU*, *LSTMs* are also introduced into the propagation process. Some common LSTM-based gated models are **Tree LSTM** [55] and **Graph LSTM** [56], [57].

3.2 Convolutional Method

Convolutional graph neural networks (ConvGNNs) has gained great success in recent years because they are more efficient and convenient to composite with other neural network [58]. The main difference between RecGNNs and ConvGNNs is that RecGNNs adopt a fixed recurrent operation to update the node's representation to a convergent state, whereas, in ConvGNNs, each convolution layer contains a different weight $\mathbf{W}$ and does not require convergence (Figure 5).

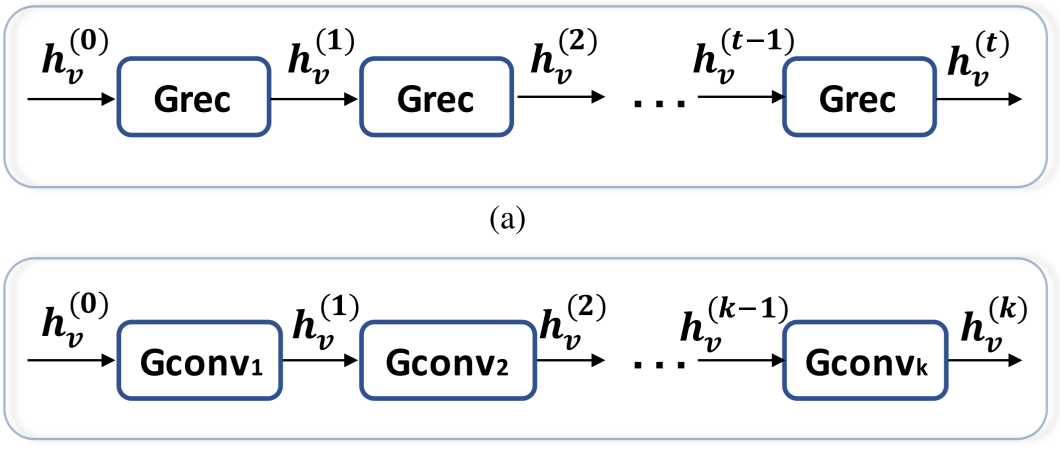

Figure 5: The major difference between RecGNNs and ConvGNNs (a) RecGNNs use the same recurrent layers (**Grec**) to update the node state, while ConvGNNs (b) update the node embedding using different a certain number of convolutional layers ($\mathbf{Gconv_k}$) with different weight [58]

Generally, ConvGNNs can be divided into two categories: spectral-based approaches and spatial-based approaches.

3.2.1 Spectral Approaches

Spectral-based methods define graph convolution from the perspective of graph signal processing [59]–[61]. The normalized graph Laplacian matrix L^{sys} is *real symmetric positive semidefinite*, so it can be decomposed as $L = U\Lambda U^T$ using eigenvalue decomposition. Here, $U = [u_0, u_1, ...u_{n-1}] \in \mathbb{R}^{n\times n}$ is the eigenvectors matrix. Since those eigenvectors are orthogonal to each other, they can form a new **spectral space** as a set of basis vectors.

The graph signal $\mathbf{x} = [x_0, x_1, ..., x_i] \in \mathbb{R}^n$, where x_i is the value of the *i-th* node, will be transformed to the spectral domain using the graph Fourier transform $\mathscr{F}(\mathbf{x}) = \mathbf{U}^T\mathbf{x}$. The essence of graph Fourier transform is to represent the graph as a linear combination of those basis vectors in U, which can be written as equation (8), and it is the expression of inverse graph Fourier transform.

$$\begin{aligned} \mathbf{x} &= \sum_i \hat{x}_i \mathbf{u}_i \\ \mathscr{F}^{-1}(\hat{\mathbf{x}}) &= \mathbf{U}\hat{\mathbf{x}} \end{aligned} \tag{8}$$

Finally the graph convolution between the filter $\mathbf{g} \in \mathbb{R}^n$ and the graph signal x can be defined as:

$$\begin{aligned} \mathbf{g} * \mathbf{x} &= \mathscr{F}^{-1}(\mathscr{F}(\mathbf{g}) \odot \mathscr{F}(\mathbf{x})) \\ &= \mathbf{U}\left(\mathbf{U}^T\mathbf{g} \odot \mathbf{U}^T\mathbf{x}\right) \end{aligned} \tag{9}$$

Here, the $\mathbf{U}^T\mathbf{g} \in \mathbb{R}^n$ is the spectral domain expression of the filter. By replacing it with $\mathbf{g}_w = \mathbf{diag}(\mathbf{U}^T\mathbf{g}) \in \mathbb{R}^{n\times n}$. We can derive the general expression for all Spectral-based ConvGNNs. The only difference lies in the choice of $\mathbf{g}_w$.

$$\mathbf{g}_w * \mathbf{x} = \mathbf{U}\mathbf{g}_w\mathbf{U}^T\mathbf{x} \tag{10}$$

Spectral Network [62] regards the filter matrix as a learnable parameter, that is $\mathbf{g}_w = \mathbf{diag}(\mathbf{w})$. However, due to the eigenvector decomposition operation of the Laplacian matrix, Spectral Network suffers from two several drawbacks. The first problem is the eigenvector decomposition operation is computation inefficient which requires $O(n^3)$ computational complexity. And the second problem is the learned filter is

non-spatially localized.

ChebNet [63] approximate the filter $\mathbf{g}_w$ using the truncated version of Chebyshev polynomials $T_k(x)$ up to K^{th} order. So, the convolution of a graph signal x with the Cheby filter g_w can be written as:

$$\mathbf{g}_w * \mathbf{x} \approx \sum_{k=0}^{K} w_k \mathbf{T}_k(\tilde{\mathbf{\Lambda}})\mathbf{x} \tag{11}$$

Where $\tilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_{\max} - \mathbf{I_n}$ is to map the value of eigenvalue Λ into $[-1, 1]$. $\mathbf{w} \in \mathbb{R}^K$ is the coefficient vector for the Chebyshev polynomial. And the Chebyshev polynomial is generated recursively using $T_i(x) = 2\mathbf{x}T_{k-1}(x) - T_{k-2}(\mathbf{x})$ with $T_0(x) = 1$ and $T_1(x) = x$. As a improved version of Spectral Network, the Chebyshev kernel has good spatial localization (*K-localized* property), which means it can extract local feature regardless of the size of the graph. Besides, such a K-localized convolution method can avoid the eigenvalue decomposition of the Laplacian matrix, and therefore reduce the computation complexity.

Graph Convolution Network (GCN) [20], is actually a special case of ChebyNet that sets K in equation (11) to 1 (a first-order approximation) and assumes $\lambda_{max} = 2$. The equation therefore, can be simplified as:

$$\mathbf{g}_w * \mathbf{x} \approx w_0\mathbf{x} - w_1\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{X} \tag{12}$$

For the purpose of avoiding over-fitting caused by too many parameters, GCN further combines the parameter w_0 and w_1 into w and obtain the following expression:

$$\mathbf{g}_w * \mathbf{x} \approx w\left(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{x} \tag{13}$$

GCN implements a normalization trick to handle with the problem of *exploding/vanishing gradient* [29]. That is, replace $\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ by $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_n}$ and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. Below is the final GCN function:

$$\mathbf{h}_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \mathbf{W}^{(k)}\tilde{\mathbf{A}}_{v,u}\mathbf{h}_u^{(k-1)}\right) \tag{14}$$

Recent works are the improved version of the GCN model, such as **Adaptive Graph Convolution Network (AGCN)** [64], which learned a "residual" graph Laplacian and merged it with the original Laplacian

matrix. As a result, it can learn the hidden structural relations between nodes. **Dual Graph Convolutional Network (DGCN)** [65], as the name suggests, has implemented a two-graph convolutional network architecture to capture local and global consistency on graphs respectively.

A key shortcoming of spectral methods is their dependence on Fourier basis. This results in each model being associated with its corresponding graph domain and cannot be migrated to other new graph domains, indicating poor model migration capabilities.

3.2.2 General Spatial Approaches

Considering the special properties of the graph, the basic idea of a spatial-based GNN layer is to compress a set of vectors (messages) into a single vector, which generally involves two steps: **(1) Message Computation; (2) Message Aggregation**. And an example of two layers GNNs is shown in figure 6:

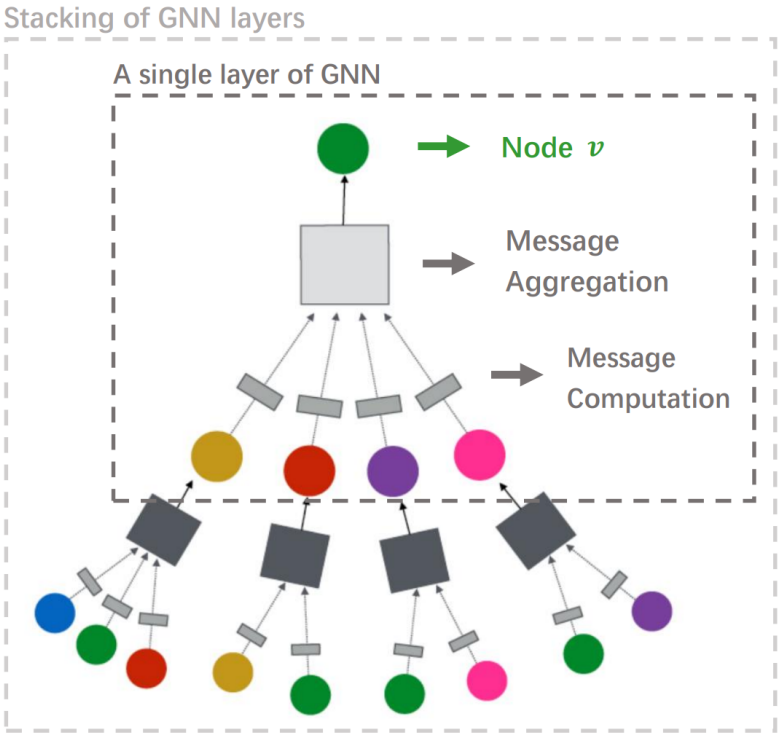

Figure 6: Stacking of two GNN layers

As can be observed from the inner dotted box of figure 2, **the Message Computation** operation takes the embedding of the node at the previous layer and create a new message $\mathbf{m}_u^{(k)}$ using a certain transformation $\text{MSG}^{(k)}$. Therefore, the message computation function of the k-th layer can be written as:

$$\mathbf{m}_u^{(k)} = \mathrm{MSG}^{(k)}\left(\mathbf{h}_u^{(k-1)}\right), u \in \{\mathcal{N}_v \cup v\} \tag{15}$$

Here, to avoid losing information of node v itself, we also add its embedding $\mathbf{h}_v^{(k-1)}$ to the message computation function.

The Message Aggregation operation will aggregate all the transformed messages from node v's neighbor nodes u, which can be expressed as:

$$\mathbf{h}_v^{(k)} = \mathrm{AGG}^{(k)}\left(\left\{\mathbf{m}_u^{(k)}, u \in \mathcal{N}_v\right\}, \mathbf{m}_u^{(k)}\right) \tag{16}$$

Note that the AGG operator in equation (16) can be any possible symmetric aggregation operation (e.g., Σ, Max, Mean).

Finally, by combining the above two equations, and passing it through a nonlinear activation function ϕ, we can derive the general mathematical expression of a single spatial-based GNN layer $f(\mathbf{h}_v)$. For simplicity of expression, we will replace MSG with ψ, and AGG with $\square$.

$$\mathbf{h}_v^{(k)} = \phi\left(\mathbf{h}_v^{(k-1)}, \underset{u \in \mathcal{N}_v}{\square} \psi\left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}\right)\right) \tag{17}$$

By further introduce the edge feature vector $\mathbf{x}_{(v,u)}^e \in \mathbf{R}^c$ into this equation, and replace ϕ with $U_k(\cdot)$ which is a function with learnable parameters. We can obtain a general spatial-based ConvGNNs framework, namely the **Message-Passing Neural Network (MPNN) [66]:**

$$\mathbf{h}_v^{(k)} = U_k\left(\mathbf{h}_v^{(k-1)}, \sum_{u \in \mathcal{N}_v} \psi\left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{x}_{(v,u)}^e\right)\right) \tag{18}$$

Besides, by comparing equation (14) and (17), we can observe that they have the same format, suggesting that **GCN** can also be regarded as a spatial-based method.

Mixture model Network (MoNet) [67] also proposed a generic spatial-based ConvGNNs framework. Many existing spatial-based GNN model can be unified as special cases by this framework, such as [20], [68]–[71]. In Monet, each node v in a graph is regarded as the origin of a *pseudo-coordinates system* $\mathbf{u}(v, u)$. And

$\mathbf{w_\Theta}(\mathbf{u}) = (w_1(\mathbf{u}), \ldots, w_J(\mathbf{u}))$ is a defined weighting function parametrized by $\mathbf{\Theta}$. The graph convolution between two function f and g can be defined as:

$$\begin{aligned} D_j(v)f &= \sum_{u \in \mathcal{N}_v} w_j(\mathbf{u}(v,u)) f(u) \\ (f * g) &= \sum_{j=1}^{J} g_j D_j(v) f \end{aligned} \tag{19}$$

Here, j is the dimensionality of the extracted patch, and $D_j(v)f$ is the aggregated values of all neighbors' weighting function. For MoNet, the pseudo-coordinate for (v,u) is $\mathbf{u}(x,y) = \left(\frac{1}{\sqrt{\deg(x)}}, \frac{1}{\sqrt{\deg(y)}}\right)^\top$ and $w_j(\mathbf{u}) = \exp\left(-\frac{1}{2}\left(\mathbf{u} - \boldsymbol{\mu}_j\right)^\top \Sigma_j^{-1}\left(\mathbf{u} - \boldsymbol{\mu}_j\right)\right)$.

3.2.3 Other Spatial-based Benchmarks

Neural Network for Graph (MN4G) [72], as the first spatial-based GNNs, performs graph convolution by summing up the neighborhoods' transformed embedding, $\mathbf{\Theta}^{(k)^T}\mathbf{h}_u^{(k-1)}$ and its own transformed embedding, $\mathbf{W}^{(k)^T}\mathbf{x}_b$ together, which can be written as:

$$\mathbf{h}_v^{(k)} = \phi\left(\mathbf{W}^{(k)^T}\mathbf{x}_v + \sum_{i=1}^{k-1}\sum_{u \in \mathcal{N}_v} \mathbf{\Theta}^{(k)^T}\mathbf{h}_u^{(k-1)}\right) \tag{20}$$

To represent the MN4G in matrix form:

$$\mathbf{H}^{(k)} = \phi\left(\mathbf{X}\mathbf{W}^{(k)} + \sum_{i=1}^{k-1}\mathbf{A}\mathbf{H}^{(k-1)}\mathbf{\Theta}^{(k)}\right) \tag{21}$$

Since the adjacency matrix here is not normalized and Sum$(\cdot)$ is chosen as the aggregator, there may be large differences between the node hidden states.

Diffusion Convolutional Neural Network (DCNN) [71] regarded the graph convolution as a diffusion process. They describe the probability of 1-hop information transmission between two nodes by establishing a probability transition matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ and $\mathbf{P} \in \mathbb{R}^{n \times n}$. Thusly, $\mathbf{P}^k$ can represent the transfer probability of k-hop information transmission. The *diffusion graph convolution (DGC)* can be defined as:

$$\mathbf{H}^{(k)} = \phi\left(\mathbf{W}^{(k)} \odot \mathbf{P}^k\mathbf{X}\right) \tag{22}$$

Where $\mathbf{W}^{(k)}$ is the matrix of learnable parameters and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the input feature matrix. Unlike the other spatial-based convolution method, for *DGC*, the hidden representation matrix $\mathbf{H}^{(k)}$ is irrelevant to $\mathbf{H}^{(k-1)}$, which means they all have the same dimension as $\mathbf{X}$. For the final output of the model, DCNN uses the concatenation of the output $\mathbf{H}^{(k)}$ of all diffusion steps together. But in *DGC* [73], the output is equal to the summation of the hidden representation matrix which is defined as equation (23):

$$\mathbf{H} = \sum_{k=0}^{K} f\left(\mathbf{P}^k \mathbf{X} \mathbf{W}^{(k)}\right) \tag{23}$$

Graph Sample and Aggregate (GraphSAGE) [35] proposed a node sampling algorithm to solve the problem of low computational efficiency caused by the excessive number of neighbor nodes. Instead of aggregating information on all neighbor nodes, GraphSAGE samples n neighbor nodes for each vertex. If the number of neighbors of nodes is less than n, the sampling method with replacement is adopted (Otherwise, the method without replacement will be adopted) until n nodes are sampled. In GraphSAGE, the graph convolution can be defined as follow, which can be consider as a special case of *MPNN*:

$$\mathbf{h}_v^{(k)} = \sigma\left(\mathbf{W}^{(k)} \cdot \underset{u \in S_{\mathcal{N}_v}}{\square} \left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}\right)\right) \tag{24}$$

where $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. To assure the permutation invariance property of aggregation, $S_{\mathcal{N}_v}$ is the random sampling on the node v's neighbors, and $\square$ is a kind of symmetric aggregator.

Graph Attention Networks (GAT) [74], has adopted an attention mechanism α to measure the degree of relationship between two connected nodes. Therefore, each node in the graph can be assigned different weights based on the feature vector of its neighbors. Besides, GAT can be also considered as a special case of *MPNN*, which can be written as:

$$\mathbf{h}_v^{(k)} = \phi\left(\mathbf{h}_v^{(k-1)}, \underset{u \in \mathcal{N}_v}{\square} \alpha\left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}\right) \psi\left(\mathbf{h}_u^{(k-1)}\right)\right) \tag{25}$$

Here, $\psi\left(\mathbf{h}_u^{(k-1)}\right) = \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}$, where $\mathbf{W}^{(k)}$ is the matrix of learnable parameters. And the attention weight $\alpha\left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}\right)$ can be expressed as:

$$\alpha\left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}\right) = \text{softmax}\left(\text{LeakyReLU}\left(\mathbf{\Theta}^T \left[\mathbf{W}^{(k)} \mathbf{h}_v^{(k-1)} \| \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right]\right)\right) \tag{26}$$

Compared to other models, GAT has the following strengths:

- **Higher Computational Efficiency:** GAT does not require complex matrix operations such as eigenvalue decomposition, and its computation complexity is:

  $$O\left(|\mathcal{V}| \times F \times F'\right) + O\left(|\mathcal{E}| \times F'\right)$$

  where $|\mathcal{V}|$ is the number of nodes, $|\mathcal{E}|$ is the number of edges, and F and F' respectively represent the input and output's dimensionality.

- **Higher Expressive Capability:** The attention mechanism assigns different importance to every single node, which improves the expressiveness of the model.

- **Higher Information Integrity:** Unlike in *GraphSAGE*, all neighbor nodes are considered in GAT.

- **Higher Generalization Ability:** The attention mechanism is shared for all edges in the graph. Therefore, GAT is also a localized model. This characteristic gives GAT a high degree of generalization Ability

Graph Isomorphism Network (GIN) [75] proves that in graph isomorphism problems, GNN can at most have the same expressive capability as Weisfeiler-Lehman(WL) [76] test. Moreover, GIN improves the expressivity of GNNs by introducing a learnable parameter $\epsilon^{(k)}$ on the weight of the center node:

$$\mathbf{h}_v^{(k)} = \text{MLP}\left(\left(1 + \epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}v} \mathbf{h}_u^{(k-1)}\right) \tag{27}$$

where $\text{MLP}(\cdot)$ represents a multilayer perceptron.

3.3 Summary of Models

Here, we are going summary all of the mentioned methods in terms of their categories and Inputs.

4 Application

GNN has been applied to many interdisciplinary fields with great success due to the powerful expressivity of graphs. This includes not only classical AI research fields such as Computer Vision (CV) and Natural Language Processing (NLP), but also many disciplines such as biology, chemistry, and physics. In the

Table 4: Summary of RecGNNs and ConvGNNs

Models	Category	Inputs
GNN* (2008) [45], [46]	RecGNN	A, X, X^e
SSE (2018) [52]	RecGNN	A, X
GGNN (2015) [54]	Gated-RecGNN	A, X
Spectral CNN (2013) [62]	Spectral-based ConvGNN	A, X
ChebyNet (2016) [63]	Spectral-based ConvGNN	A, X
AGCN (2018) [64]	Spectral-based ConvGNN	A, X
DGCN (2018) [65]	Spectral-based ConvGNN	A, X
NN4G (2009) [72]	Spatial-based ConvGNN	A, X
DCNN (2016) [71]	Spatial-based ConvGNN	A, X
MPNN (2017) [66]	Spatial-based ConvGNN	A, X, X^e
MoNet (2017) [67]	Spatial-based ConvGNN	A, X
GraphSAGE (2017) [35]	Spatial-based ConvGNN	A, X
GAT (2017) [74]	Spatial-based ConvGNN	A, X
GIN (2018) [75]	Spatial-based ConvGNN	A, X

following section, we will present the most representative applications of GNNs in different disciplines.

Biology: There are numerous amino acid residues on the surface of proteins, which are directly related to the properties and functions of proteins. Therefore, one typical instance of implementing GNN techniques in the biology field is in protein interface prediction. By letting the residues be nodes, the surface of a protein can be modeled as a graph. Recent studies [77], [78] were established based on the assumption that the biological properties of the central residue depend in part on the properties of its neighbors' residuals. For example, Fout et al. [10] has proposed a GCN based method to learn features across pairs of proteins and determine the existence of residues

Medical: GNN is widely used in medical image analysis and has achieved great success because there are a large number of medical data with complex structures (e.g, CT, MRI), and 2d images cannot well represent some structural information of these data. For example, Hao et al. [79] considered every Computed Tomography (CT) slice as the nodes and proposed the *UG-GAT* by introducing uncertainty into *GAT* to distinguish the UPPE and CPPE image in a CT scanning. Azcona et al. employed the idea of *SpiralNet++* [80] into the structures of *Conditional Variational Auto-Encoder (CVAE)* [81] and proposed a *generation model Discriminative and Generative Spiral Networks* to help doctors diagnose Alzheimer's

syndrome and predict its progression.

Chemistry: Complex molecular structures in chemistry can be naturally represented as graphs and such an encoding process of chemical molecular structures is called a molecular fingerprint. Compared to the classical static fingerprint methods (e.g., one-hot encoding), the inertia of the graph enables it to embed the dynamic structural features more effectively. Scientists often use these encoded molecular structures to conduct molecular structural studies, such as drug molecular structural design [82]–[84] and chemical reaction prediction [85], [86].

Recommendation System: User-item collaborative filtering, sequential recommendation, social recommendation, and knowledge graph-based Recommendation [87] are four typical recommendation systems, and the information used in those systems are all graph-structured. GNN was widely used in those recommendation system because recent studies have revealed the ground-breaking graph representation learning capability of GNNs [29].

For example, **social recommendation systems** assume that every user in a social community is inevitably influenced by their social network (e.g., friends, friends of friends). From a graph perspective, this suggests that the representation of nodes in social networks should be largely determined by their neighbors' representation. *Diffusion Neural Network (DiffNet)* [88] stimulate the recursive social diffusion process by assuming that the central node will be always influenced equally by its neighbor nodes. But this assumption ignores that in reality, people tend to be influenced by friends with stronger social ties. Therefore, the attention scheme was widely implemented in the later GNN-based social recommendation models, such as [89] which consider the strength of heterogeneity of social relationships and allocate different attention to different edges accordingly.

Traffic System: Graph Neural Network can also be employed in traffic forecasting tasks since traffic networks are also graph-liked. An additional consideration for transportation network is that transportation system is always dynamic. Yu et al. [90] use *Spatial-temporal Graph Neural Networks (STGNNs)* to capture the spatial and temporal dependencies of a graph simultaneously. In this paper, the nodes are the sensors installed on roads (with the average traffic speed within a window as dynamic input features), while the

edges are the distance between two neighbor sensors.

5 Future Directions

Although GNN has achieved remarkable success in many fields, it is still unable to deal with some specific graph-based problems due to the complexity of graph structure. In this section, we list some possible future research directions.

Graph Structure: With the further study of GNN, the graph structures are tend to become more complex and flexible. In many studies, researchers attempt to build a more complex graph model to solve those existing problems, such as heterogeneous graphs and dynamic graphs. As more application scenarios emerge, there will certainly be more demands on the structure of graphs.

Scalability Issue: Network such as social networks are becoming larger and more complex. The solution of the existing model is to randomly sample a fixed number of nodes in information Propagation to reduce the computational complexity of the model. The price is the loss of graph integrity, because a node may lose some influential neighbor during sampling. Therefore, how to reduce computational complexity on the premise of preserving information integrity as much as possible can become one of the possible research directions in the future.

Locality: Many models are designed to focus on local behavior in graph-structure data, but these methods cannot take into account some long-range spatial dependencies between nodes. Therefore, how to make the model capture both local and long-range spatial dependencies will be a research problem in the future

Depth of the Model: As the increasing of GNN layers, the reception field of nodes will greatly overlap, resulting in the decrease in network performance. Therefore, how to choose the number of convolutional layers reasonably in different application scenarios will be a possible research topic in the future.

6 Conclusion

In this study, we conduct a comprehensive review of existing graph neural networks, with a special focus on convolutional graph neural networks and recurrent graph neural networks. We first propose a set of general troubleshooting pipelines to categorize and solve GNN-based tasks. In terms of GNN models, we classify several GNN benchmark models, conduct in-depth theoretical analysis, and provide unified mathematical representations. Finally, we introduce several GNN-related applications and suggest the future research direction.

References

[1] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series", *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words", *Neurocomputing*, vol. 139, pp. 84–96, 2014.

[4] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.", *Journal of machine learning research*, vol. 11, no. 12, 2010.

[5] C. Affonso, A. L. D. Rossi, F. H. A. Vieira, A. C. P. de Leon Ferreira, *et al.*, "Deep learning for biological image classification", *Expert Systems with Applications*, vol. 85, pp. 114–122, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] G. Hinton, L. Deng, D. Yu, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", *Advances in neural information processing systems*, vol. 28, 2015.

[9] N. Xu, P. Wang, L. Chen, J. Tao, and J. Zhao, "Mr-gnn: Multi-resolution and dual graph neural network for predicting structured entity interactions", *arXiv preprint arXiv:1905.09558*, 2019.

[10] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks", *Advances in neural information processing systems*, vol. 30, 2017.

[11] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 346–353.

[12] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion", *arXiv preprint arXiv:1706.02263*, 2017.

[13] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding", *IEEE transactions on knowledge and data engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[14] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method", *arXiv preprint arXiv:1907.04931*, 2019.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", *arXiv preprint arXiv:1301.3781*, 2013.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", *Advances in neural information processing systems*, vol. 26, 2013.

[17] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations", in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[18] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks", in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[19] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications", *arXiv preprint arXiv:1709.05584*, 2017.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks", *arXiv preprint arXiv:1609.02907*, 2016.

[21] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, 4. Springer, 2006, vol. 4.

[22] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. N. Metaxas, "Semantic graph convolutional networks for 3d human pose regression", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3425–3435.

[23] B. Cai, Y. Wang, L. Zeng, Y. Hu, and H. Li, "Edge classification based on convolutional neural networks for community detection in complex network", *Physica A: Statistical Mechanics and its Applications*, vol. 556, p. 124 826, 2020.

[24] M. Zhang and Y. Chen, "Link prediction based on graph neural networks", *Advances in neural information processing systems*, vol. 31, 2018.

[25] J. Kunegis and A. Lommatzsch, "Learning spectral graph transformations for link prediction", in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 561–568.

[26] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification", in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[27] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention", in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1666–1674.

[28] S. Wang, Z. Chen, X. Yu, *et al.*, "Heterogeneous graph matching networks", *arXiv preprint arXiv:1910.08074*, 2019.

[29] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications", *AI Open*, vol. 1, pp. 57–81, 2020.

[30] I. Henrion, J. Brehmer, J. Bruna, *et al.*, "Neural message passing for jet physics", 2017.

[31] Y. Zhang, J. Hare, and A. Prügel-Bennett, "Fspool: Learning set representations with featurewise sort pooling", *arXiv preprint arXiv:1906.02795*, 2019.

[32] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling", *Advances in neural information processing systems*, vol. 31, 2018.

[33] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3693–3702.

[34] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach", *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[35] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs", *Advances in neural information processing systems*, vol. 30, 2017.

[36] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction", *arXiv preprint arXiv:1710.10568*, 2017.

[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems", in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[38] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling", *arXiv preprint arXiv:1801.10247*, 2018.

[39] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning", *Advances in neural information processing systems*, vol. 31, 2018.

[40] D. Zou, Z. Hu, Y. Wang, S. Jiang, Y. Sun, and Q. Gu, "Layer-dependent importance sampling for training deep and large graph convolutional networks", *Advances in neural information processing systems*, vol. 32, 2019.

[41] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks", in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.

[42] C. Yang, R. Wang, S. Yao, S. Liu, and T. Abdelzaher, "Revisiting over-smoothing in deep gcns", *arXiv preprint arXiv:2003.13663*, 2020.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks", in *European conference on computer vision*, Springer, 2016, pp. 630–645.

[44] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks", *Advances in neural information processing systems*, vol. 29, 2016.

[45] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model", *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[46] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains", in *Proceedings. 2005 IEEE international joint conference on neural networks*, vol. 2, 2005, pp. 729–734.

[47] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures", *IEEE transactions on Neural Networks*, vol. 9, no. 5, pp. 768–786, 1998.

[48] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures", *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714–735, 1997.

[49] M. Hagenbuchner, A. Sperduti, and A. C. Tsoi, "A self-organizing map for adaptive processing of structured data", *IEEE transactions on Neural Networks*, vol. 14, no. 3, pp. 491–505, 2003.

[50] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine", *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[51] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment", *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[52] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steady-states of iterative algorithms over graphs", in *International conference on machine learning*, PMLR, 2018, pp. 1106–1114.

[53] K. Cho, B. Van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation", *arXiv preprint arXiv:1406.1078*, 2014.

[54] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks", *arXiv preprint arXiv:1511.05493*, 2015.

[55] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks", in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1556–1566. DOI: 10.3115/v1/P15-1150. [Online]. Available: https://aclanthology.org/P15-1150.

[56] V. Zayats and M. Ostendorf, "Conversation modeling on reddit using a graph-structured lstm", *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 121–132, 2018.

[57] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms", *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 101–115, 2017.

[58] 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, 2020.

[59] 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.

[60] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs", *IEEE transactions on signal processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[61] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory<? pub _newline=""?", *IEEE transactions on signal processing*, vol. 63, no. 24, pp. 6510–6523, 2015.

[62] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs", *arXiv preprint arXiv:1312.6203*, 2013.

[63] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering", *Advances in neural information processing systems*, vol. 29, 2016.

[64] 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, 2018.

[65] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification", in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 499–508.

[66] 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*, PMLR, 2017, pp. 1263–1272.

[67] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5115–5124.

[68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[69] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data", *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[70] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks", *Advances in neural information processing systems*, vol. 29, 2016.

[71] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks", *Advances in neural information processing systems*, vol. 29, 2016.

[72] A. Micheli, "Neural network for graphs: A contextual constructive approach", *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[73] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting", *arXiv preprint arXiv:1707.01926*, 2017.

[74] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks", *stat*, vol. 1050, p. 20, 2017.

[75] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?", *arXiv preprint arXiv:1810.00826*, 2018.

[76] B. L. Douglas, "The weisfeiler-lehman method and graph isomorphism testing", *arXiv preprint arXiv:1101.5211*, 2011.

[77] N. Pancino, A. Rossi, G. Ciano, *et al.*, "Graph neural networks for the prediction of protein-protein interfaces.", in *ESANN*, 2020, pp. 127–132.

[78] Y. Xia, C.-Q. Xia, X. Pan, and H.-B. Shen, "Graphbind: Protein structural context embedded rules learned by hierarchical graph neural networks for recognizing nucleic-acid-binding residues", *Nucleic acids research*, vol. 49, no. 9, e51–e51, 2021.

[79] J. Hao, J. Liu, E. Pereira, *et al.*, "Uncertainty-guided graph attention network for parapneumonic effusion diagnosis", *Medical Image Analysis*, vol. 75, p. 102 217, 2022.

[80] S. Gong, L. Chen, M. Bronstein, and S. Zafeiriou, "Spiralnet++: A fast and highly efficient mesh convolution operator", in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[81] T. Zhao, R. Zhao, and M. Eskenazi, "Learning discourse-level diversity for neural dialog models using conditional variational autoencoders", *arXiv preprint arXiv:1703.10960*, 2017.

[82] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, *et al.*, "Convolutional networks on graphs for learning molecular fingerprints", *Advances in neural information processing systems*, vol. 28, 2015.

[83] J. Xiong, Z. Xiong, K. Chen, H. Jiang, and M. Zheng, "Graph neural networks for automated de novo drug design", *Drug Discovery Today*, vol. 26, no. 6, pp. 1382–1393, 2021.

[84] P. Bongini, M. Bianchini, and F. Scarselli, "Molecular generative graph neural networks for drug discovery", *Neurocomputing*, vol. 450, pp. 242–252, 2021.

[85] Y. Wang, J. Wang, Z. Cao, and A. Barati Farimani, "Molecular contrastive learning of representations via graph neural networks", *Nature Machine Intelligence*, pp. 1–9, 2022.

[86] P. Bove, A. Micheli, P. Milazzo, and M. Podda, "Prediction of dynamical properties of biochemical pathways with graph neural networks.", in *Bioinformatics*, 2020, pp. 32–43.

[87] S. Wu, F. Sun, W. Zhang, and B. Cui, "Graph neural networks in recommender systems: A survey", *arXiv preprint arXiv:2011.02260*, 2020.

[88] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation", in *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, pp. 235–244.

[89] W. Fan, Y. Ma, Q. Li, *et al.*, "Graph neural networks for social recommendation", in *The world wide web conference*, 2019, pp. 417–426.

[90] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting", *arXiv preprint arXiv:1709.04875*, 2017.