A Survey of Graph Neural Networks: Methods, Applications, and Challenges

Weijie He

February 2024

Abstract

Graph Neural Networks (GNNs) have emerged as a powerful paradigm for learning representations of graph-structured data, which provides relational information between elements and is a crucial type of data for various scientific domains. Most GNNs follow and extend the idea of traditional deep learning to graph domains, enabling significant advancements in many real-world applications such as social network analysis, chemistry, and recommendation systems. This paper delves into popular GNNs like Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs) and Graph Auto-Encoders (GAEs). It also discusses some of the theoretical analysis of GNNs. Furthermore, the paper provides a review of diverse applications of GNNs and challenges (limitations) and trending future directions.

1 Introduction

While deep learning has achieved significant success in processing Euclidean data with grid-like structures, such as images, text, and sequences, Graph Neural Networks (GNNs) have emerged as a pivotal advancement in machine learning for handling non-Euclidean data. Traditional neural network architectures, including Convolutional Neural Networks (CNNs) [1], Recurrent Neural Networks (RNNs) [2], and Transformers [3], often face challenges in dealing with graph data, which encompasses a set of nodes and their relationships. GNNs, on the other hand, are specifically designed to address these challenges, making them highly effective in applications involving complex relational structures, such as social network analysis and molecular interaction modeling.

At the core of most GNNs is the message-passing mechanism, where each node aggregates information from its neighbors, enabling the network to learn embeddings that reflect the graph's topology. This feature makes GNNs particularly suited for tasks like node classification and link prediction.

GNNs extend the concept of convolution, and attention mechanisms, originally popularized by CNNs and Transformers, to graph domains. This adaptation allows for the extraction of features that respect the graph's geometry,

leading to the development of various GNN architectures, each with its unique approach to capturing graph-structured information.

In the field of unsupervised learning, Graph Autoencoders (GAEs) play a crucial role. These models leverage the power of graph-based data to learn latent representations without the need for labeled data, leading to applications of both network embeddings and graph generation.

The applications of GNNs are diverse, spanning chemistry, recommendation systems, and computer vision. In chemistry, GNNs model molecular structures to predict properties [42], while in recommendation systems [33, 34, 35, 36], they utilize relationships between users and items for more personalized suggestions. In computer vision, GNNs are applied to 3D object recognition [54], leveraging spatial relationships between points and surfaces.

In conclusion, GNNs represent a significant advancement in the ability of machine learning models to handle non-Euclidean data, offering a deeper understanding of complex systems and enabling the development of more intelligent algorithms. As the field of geometric learning continues to evolve, the potential of GNNs to transform various domains of science and technology remains vast and promising.

The rest of the paper is organized as follows

- We provide a review of some of the most representative GNN models. We also introduce research on theoretical analyses of GNN models.
- We introduce the applications of GNNs and datasets for GNN experimentation in real-world scenarios.
- We summarize the challenges and future directions.

2 Background and Definition

A Brief Background The concept of GNNs was first proposed by Gori et al. [84] and was expanded upon in subsequent works by Scarselli et al. [85] and Gallicchio et al. [86]. These initial models, characterized by their recurrent nature, updated node representations through iterative neighbor information propagation until equilibrium was reached. Despite their innovative approach, these models were hindered by high computational demands, which led to the emergence of convolutional graph neural networks (ConvGNNs), which can be broadly classified into spectral-based and spatial-based approaches.

Spectral-based ConvGNNs, such as the one introduced by Bruna et al. [87], employed spectral graph theory to define graph convolutions, with subsequent studies focusing on refining and approximating these techniques [88, 6, 89].

Conversely, spatial-based ConvGNNs have a longer history, with Micheli et al. [90] exploring the use of non-recursive layers to capture graph mutual dependencies, drawing on message-passing concepts. Although initially overlooked, this approach laid the foundation for a variety of spatial-based ConvGNNs that have since gained traction [10, 91].

Beyond ConvGNNs, with the huge advancements made by transformers and unsupervised learning, recent years have seen the development of diverse GNN frameworks, such as graph autoencoders (GAEs) and Graph Attention Networks (GATs). These models can be built upon ConvGNNs or other neural network architectures tailored for graph data representation. We will delve into these models in more detail in later sections, along with other innovative approaches in the field of GNNs.

Definition Basic notations are listed in Table 1. Other use of notations will be indicated individually. Now we define the minimal set of definitions required to understand this paper.

Table 1: Commonly used notations

Notations	**Descriptions**
$\mathbb{R}_m$	m-dimensional real vector space
$a, \mathbf{a}, A$	scalar, vector, and matrix
G	Graph (unless otherwise specified)
V	Set of nodes in a graph
N_v	The neighborhood of node v
A	Adjacency matrix of a graph
$\lvert \cdot \rvert$	Cardinality
$\mathbf{h}_v^{(k)}$	Feature representation of node v of kth hidden layer
$\sigma(\cdot)$	The sigmoid function
$\odot$	Element-wise multiplication
$\lvert\lvert$	Vector concatenation
$\mathcal{F}$	Fourier Transformation
$*_G$	Graph convolution
$\mathbb{E}_{P(X)}$	The expectation under a probabilistic distribution $P(X)$

Definition 2.1 *The adjacency matrix $A \in \mathbb{R}^{n \times n}$ for a simple graph $G = (V, E)$ with n-vertices is a matrix whose ij-th entry is described by*

$$A_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \text{ and } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 2.2 *The degree matrix $D \in \mathbb{R}^{n \times n}$ for a graph $G = (V, E)$ with n-vertices is a diagonal matrix, whose entries are*

$$D_{ij} = d(v_i).$$

where $d : V \to \mathbb{N}$ gives the number of edges that are incident to vertex v_i.

Definition 2.3 *Suppose graph $G = (V, E)$ with n-vertices is undirected, then the Laplacian matrix $L \in \mathbb{R}^{n \times n}$ is defined as*

$$L = D - A$$

or equivalently,

$$L_{ij} = \begin{cases} d(v_i) & \text{if } i = j, \\ -1 & \text{if } \{v_i, v_j\} \in E \text{ and } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

3 Convolutional Graph Neural Networks

3.1 Spectral Method

The spectral method in GCNs was inspired by the theory of spectral graph and was first introduced to graph signal processing by [4], which extends the convolution to the domain of spectral. For the undirected graphs, the normalized Laplacian matrix of the undirected graph is defined as $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, or equivalently, $\mathcal{L} = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where D is a diagonal matrix with $D_{ii} = deg(v_i)$, and L is the Laplacian matrix and A is the adjacency matrix representation of the graph. Since $\mathcal{L}$ is real symmetric and positive semidefinite, we can decompose it into $\mathcal{L} = U \Lambda U^T$, where U is unitary and its columns are ordered eigenvectors of the normalized Laplacian matrix $\mathcal{L}$, and Λ is a diagonal matrix whose entries are the eigenvalues of the ordered eigenvectors respectively.

In graph signal settings, a graph signal $\mathbf{x} \in \mathbb{R}^n$ is a feature vector whose ith entry represents ith node, and the Fourier Transformation of the signal to the spectral domain is defined as $\mathscr{F}(\mathbf{x}) = U^T \mathbf{x}$, since U is unitary, we can immediately infer that the inverse Fourier Transformation of $\mathbf{x}$ is: $\mathscr{F}^{-1}(\mathbf{x}) = U\mathbf{x}$. Thus based on the convolution theorem [5], the convolution of graph signal by a filter $\mathbf{g} \in \mathbb{R}^n$ is defined as follows:

$$\mathbf{g} *_G \mathbf{x} = \mathscr{F}^{-1}(\mathscr{F}(\mathbf{x}) \odot \mathscr{F}(\mathbf{g})), \tag{1}$$

or by definition,

$$\mathbf{g} *_G \mathbf{x} = U(U^T \mathbf{x} \odot U^T \mathbf{g}), \tag{2}$$

For implications, we choose $\mathbf{g}$ to be a learnable matrix $G_\theta = diag(U^T \mathbf{g})$, then the Equation 2 can be simplified as

$$G_\theta *_G \mathbf{x} = U G_\theta U^T \mathbf{x} \tag{3}$$

All spectral-based GCNs use this definition, with only the choices of the learnable filter G_θ varying from one to another. Now we introduce two important filter designs that reduce the computational complexity.

ChebNet [6] approximates the filter G_θ by a truncated expansion in terms of Chebyshev polynomials $T_i(\mathbf{x})$ up to Kth order, i.e,

$$G_\theta = \sum_{i=0}^{K} \theta_i T_i(\tilde{\Lambda}) \tag{4}$$

where $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_n$, and the values of $\tilde{\Lambda}$ lies in [-1, 1]. The Chebyshev polynomials are defined as $T_i(\mathbf{x}) = 2\mathbf{x}T_{i-1}(\mathbf{x}) - T_{i-2}(\mathbf{x})$ with $T_0(\mathbf{x}) = 1$ and $T_1(\mathbf{x}) = \mathbf{x}$. Put it in Equation 3, we get:

$$G_\theta *_G \mathbf{x} = U(\sum_{i=0}^{K} \theta_i T_i(\tilde{\Lambda}))U^T\mathbf{x} \tag{5}$$

Choose $\tilde{\mathcal{L}} = 2\mathcal{L}/\lambda_{max} - I_n$, and since $\mathcal{L} = U\Lambda U^T$, we obtain $T_i(\tilde{\mathcal{L}})) = UT_i(\tilde{\Lambda})U^T$. By using induction on I, we can prove ChebNet takes the form,

$$G_\theta *_G \mathbf{x} = \sum_{i=0}^{K} \theta_i T_i(\tilde{\mathcal{L}})\mathbf{x} \tag{6}$$

It can be observed that the operation is K-localized since it is a Kth-order polynomial in the Laplacian. Defferrard et al. [6] use this K-localized convolution to define a convolutional neural network which could remove the need to compute the eigenvectors of the Laplacian.

Graph Convolutional Network (GCN) [7] further explores this method and introduces a first-order approximation of ChebNet, by assuming $K = 1$ and $\lambda_{max} = 2$, Equation 6 is simplified as

$$G_\theta *_G \mathbf{x} = \theta_0\mathbf{x} - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\mathbf{x} \tag{7}$$

To further simplify and avoid over-fitting, GCN assumes $\theta_0 = \theta_1 = \theta$, thus we get the following equation.

$$G_\theta *_G \mathbf{x} = \theta(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})\mathbf{x} \tag{8}$$

To allow multi-channels of inputs and outputs, GCN modifies Equation 8 into a compositional layer, defined as

$$Z = X *_G G_\Theta = f(\bar{A}X\Theta) \tag{9}$$

where $\bar{A} = I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ and $f(\cdot)$ is an activation function. Stacking this operator could lead to numerical instabilities and exploding/vanishing gradients, Kipf and Welling [7] introduce a renormalization trick: replace $\bar{A} = I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ by $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ where $\tilde{A} = A + I_n$ and $\tilde{D}_{ii} = \sum j\tilde{A}_{ij}$. They eventually generalize the definition to multiple input/output channels as follows:

$$Z = f(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta), \tag{10}$$

where $X \in \mathbb{R}^{N \times C}$ is the signal input and $\Theta \in \mathbb{R}^{C \times F}$ is the matrix of filter parameters and $Z \in \mathbb{R}^{N \times F}$ is the convolved singal matrix. Being a spectral-based method, GCN can be also viewed as a spatial method that aggregates feature information from a node's neighborhood. Equation 9 can be expressed as:

$$\mathbf{h}_v = f(\Theta^T(\sum_{u \in \{N(u) \cup v\}} \bar{A}_{v,u}\mathbf{x}_u)), \forall v \in V \tag{11}$$

3.2 Spatial Method

Analogous to CNNs, Spatial methods define convolutions directly on the graph topology, i.e, neighborhoods in a certain distance. The major challenge of spatial approaches is defining the convolution operation with differently sized neighborhoods and maintaining the local invariance of CNNs. Neural Network for Graphs (NN4G) [8], represents the initial step towards spatial-based Convolutional Graph Neural Networks (ConvGNNs). NN4G adopts a compositional neural architecture with separate parameters for each layer. This design enables the expansion of a node's neighborhood by incrementally building the network's architecture. NN4G performs graph convolutions by directly aggregating information from a node's neighbors. It also incorporates residual and skip connections to preserve information across layers. Consequently, NN4G updates the states of nodes in the next layer by effectively combining neighborhood information and maintaining a memory of the past layer in the following forms.

$$\mathbf{h}_v^{(k)} = f({W^{(k)}}^T\mathbf{x}_v + \sum_{i=1}^{(k)-1} \sum_{u \in \mathcal{N}(v)} {\Theta^{(k)}}^T \mathbf{h}_u^{(k)-1}) \tag{12}$$

or, in matrix form:

$$H^k = f(XW^(k) + \sum_{i=1}^{k-1} AH^{(k-1)}\Theta^{(k)}), \tag{13}$$

Note that NN4G uses the unnormalized adjacency matrix which may potentially cause hidden node states to have extremely different scales. Contextual Graph Markov Model (CGMM) [9] proposes a probabilistic model inspired by NN4G. While maintaining spatial locality, CGMM has the benefit of probabilistic interpretability.

Diffusional Convolutional Neural Network (DCNN) [10] treats graph convolutions as a diffusion process. It assumes information is transferred from one node to one of its neighboring nodes with a certain transition probability so that information distribution can reach equilibrium after several rounds. DCNN defines the diffusion graph convolution as:

$$H^{(k)} = f(W^{(k)} \odot P^k X) \tag{14}$$

where the kth hidden representation matrix $H^{(k)}$ maintains the same dimension as the input feature matrix X and is computed independently of the previous

hidden representation matrix $H^{(k-1)}$. The transition matrix $P \in \mathbb{R}^{n \times n}$, used for updating the hidden representations, is calculated as $P = D^{-1}A$, where A is the adjacency matrix of the graph and D is the diagonal degree matrix. The final output of the DCNN is obtained through a concatenation operation, denoted as $\|_{k=1}^{K} H^{(k)}$, which combines the hidden representations from all layers. The stationary distribution of a diffusion process can be represented as a summation of a power series of probability transition matrices. In this context, Diffusion Graph Convolution (DGC) [11] deviates from the traditional approach of concatenating outputs at each diffusion step. Instead, it aggregates the outputs by summing them up. DGC defines the diffusion graph convolution operation by

$$H = \sum_{k=0}^{K} f(P^k X W^{(k)}) \tag{15}$$

where $W^{(k)} \in \mathbb{R}^{D \times F}$ and $f(\cdot)$ is an activation function. Using the power of a transition probability matrix implies that distant neighbors contribute very little information to a central node.

Message Passing Neural Network (MPNN) [12] gives the general framework of spatial-based ConvGNNs in which information can be passed from one vertex to another directly via the edges. MPNN runs K-step iterations to let information propagate. It is further generalized by Xu et al. [13] as the following formulation:

Let $G = (V, E)$ denote a graph with node feature vectors $\mathbf{X}_v$, for $v \in V$. Then for k-iteration (layer):

$$\mathbf{a}_v^{(k)} = AGGREGATE^{(k)}(\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)), \mathbf{h}_v^{(k)} = COMBINE^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)}) \tag{16}$$

where $\mathbf{h}_v^{(k)}$ is the feature vector of node v at k-th layer, and $\mathbf{h}_v^0 = \mathbf{X}_v$.

For node classification, the node representation of the final iteration $\mathbf{h}_v^K$ is used for prediction. In addition, for graph classification, there is a $READOUT$ function to aggregate node features from the final iteration to obtain the entire graph representation $\mathbf{h}_G$:

$$\mathbf{h}_G = READOUT(\mathbf{h}_v | v \in G) \tag{17}$$

$READOUT$ can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function [34] [36].

In MPNN, The message-passing function is defined as follows:

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

where $\mathbf{h}_v^{(0)} = \mathbf{x}_v$, $U_k(\cdot)$ and $M_k(\cdot)$ are functions with learnable parameters. The readout function is the same as Equation 13. Xu et al. [13] also find that the representational power is bounded by the WL test, i.e, MPNN-based GNNs are unable to distinguish non-isomorphic graph structures. This interesting theoretical result will be discussed in detail in the next section.

To mitigate this problem, they propose the Graph Isomorphism Network (GIN) by adjusting the central node by a learnable parameter $\epsilon^{(k)}$, and the message-passing function becomes as follows:

$$\mathbf{h}_v^{(k)} = MLP((1+\epsilon^{(k)})\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k-1)}) \tag{19}$$

where $MLP(\cdot)$ is a multi-layer perception.

3.3 Theoretical Aspects

In this section, we discuss some theoretical results on GNNs.

Graph Isomorphism Two graphs are isomorphic if they are topologically identical. Given two non-isomorphic graphs G_1 and G_2, Xu et al. [13] prove that if a GNN $\mathcal{A}$: $\mathcal{G} \to \mathbb{R}^d$ such that $\mathcal{A}(G_1) \neq \mathcal{A}(G_2)$, then these two graphs can also be identified as non-isomorphic by the Weisfeiler-Lehman (WL) test of isomorphism [16]. Furthermore, They provide thorough conditions for a GNN that could be as powerful as the WL test in the following formulation.

Definition 3.1 *A multiset is a 2-tuple $X = (S; m)$ where S is the underlying set of X that is formed from its distinct elements, and $m : S \to \mathbb{N}_{\geq}1$ gives the multiplicity of the elements.*

Theorem 3.2 *Let $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ be a GNN. With a sufficient number of layers, $\mathcal{A}$ maps any graphs G_1 and G_2 that the WL test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:*

a)$\mathcal{A}$ aggregates and update features iteratively with

$$\mathbf{h}_v^{(k)} = \phi(\mathbf{h}_v^{(k-1)}, f(\{\mathbf{h}_u^{(k-1)} | u \in N_v\})),$$

where f operates on multiset and ϕ is injective.

b)$\mathcal{A}$'s graph-level readout, which operates on the multiset of node features $\{\mathbf{h}_\mathbf{v}^{(\mathbf{k})}\}$ is injective.

Note that the node features in WL test are one-hot embeddings that can not capture the similarity between subtrees. However, GNNs that satisfy Theorem 4.2 generalize the WL test by embedding the subtrees to low-dimensional space, where the similarity could be measured. This enables GNNs to not only discriminate different structures but also to learn to map similar graph structures to similar embeddings and capture dependencies between graph structures. Capturing structural similarity of the node labels is shown to be helpful for generalization particularly when the co-occurrence of subtrees is sparse across different graphs or there are noisy edges and node features ([17]). They also show that common GNNs such as GCN [7] and GraphSage [18] are incapable of distinguishing different graph structures.

Equivariance and Invariance A GNN is required to be an equivariant function when performing node-level tasks and be an invariant function when performing graph-level tasks. Maron et al. [19] theoretically characterize the permutation invariant and equivariant linear layers for graph data. Suppose f is a neural network, then for an arbitrary permutation matrix P and an arbitrary adjacency matrix A, we call f order invariant if it satisfies

$$f(P^T AP) = f(A) \tag{20}$$

and equivariant if it satisfies

$$f(P^T AP) = P^T f(A)P. \tag{21}$$

Maron et al. [19] generalize the formulation to hypergraphs and reduce the problem of finding all invariant and equivariant linear operators L to finding solution space to the fixed-point equation:

$$\text{invariant } L : P^{\otimes k} vec(L) = vec(L) \tag{22}$$

$$\text{equivariant } L : P^{\otimes 2k} vec(L) = vec(L) \tag{23}$$

where $\otimes$ is the Kronecker product and $P^{\otimes k} = \overbrace{P \otimes \cdots \otimes P}^{k}$. Given $L \in \mathbb{R}^{a \times b}$, $vec(L) \in \mathbb{R}^{ab \times 1}$ represents the column stack. In Equation 22, $L \in \mathbb{R}^{1 \times n^k}$ is the matrix of an invariant operator; and in Equation 23, $L \in \mathbb{R}^{n^k \times n^k}$ is the matrix of an equivariant operator.

4 Graph Attention Network

The success of the attention mechanism in machine translation [20] [21] [22] and large language models (LLMs) like BERT [23] and GPT [24] demonstrated its effectiveness in capturing long-range dependencies and dynamically weighting the importance of different input features. This inspired researchers to explore how attention mechanisms could be applied to graph-structured data, unlike GCN which treats all neighbors of a node equally, the attention mechanism can learn representations of nodes that effectively capture both their local neighborhood structure and assign different roles (weights) to their neighbors in the broader graph context. Velickovic et al. [25] propose a graph attention network (GAT) that incorporates the attention mechanism into the propagation steps. It follows the self-attention strategy and the hidden state of each node is computed by attending over its neighbors. Velickovic et al. define a single graph attentional layer so that one can construct an arbitrary length of GAT by stacking this layer. The attention coefficient of node j to i is computed as follows:

$$\alpha_{ij} = \frac{exp(LeakyReLU(\mathbf{a}^T[W\mathbf{h}_i || W\mathbf{h}_j]))}{\sum_{l \in N_i} exp(LeakyReLU(\mathbf{a}^T[W\mathbf{h}_i || W\mathbf{h}_l]))} \tag{24}$$

where N_i represents the neighborhoods of node i in the graph, and $\{\mathbf{h}_1, \ldots, \mathbf{h}_N\}, \mathbf{h}_i \in \mathbb{R}^F$ are F-dimensional input feature representations of nodes $i = 1, \ldots, N$, the output features are denoted as $\{\mathbf{h'}_1, \ldots, \mathbf{h'}_N\}, \mathbf{h'}_i \in \mathbb{R}^{F'}$. $W \in \mathbb{R}^{F' \times F}$ is the weight matrix of a linear transformation which can be applied to every node, and $\mathbf{a} \in \mathbb{R}^{2F'}$ is the weight vector. The final output feature of node i is then obtained by adding a nonlinearity σ:

$$\mathbf{h'}_i = \sigma(\sum_{j \in N_i} \alpha_{ij} W \mathbf{h}_j) \tag{25}$$

Velickovic et al. extend the multi-head attention to their model in a similar way to Vaswani et al. [22], in order to stabilize the learning process. It applies K independent attention mechanisms to compute and concatenate (or compute the average of) the hidden features, which has two following representations:

$$\mathbf{h'}_i = ||_{k=1}^{K} \sigma(\sum_{j \in N_i} \alpha_{ij}^k W^k \mathbf{h}_j) \tag{26}$$

where α_{ij}^k is the normalized attention coefficient of kth attention mechanism,$||$ is the concatenation operation. For classification tasks, if we apply multi-head attention to the prediction layer, the concatenation is no longer sensible, instead we apply *averaging*, and delay applying σ until then:

$$\mathbf{h'}_i = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} \alpha_{ij}^k W^k \mathbf{h}_j) \tag{27}$$

The attention framework presented by Velickovic et al. features multiple beneficial attributes: Firstly, it enables parallel computation of node-neighbor pairs, enhancing efficiency. Secondly, it accommodates nodes of varying degrees by allocating appropriate weights to their respective neighbors. Thirdly, it seamlessly adapts to inductive learning scenarios. Consequently, the GAT demonstrates superior performance over GCN in various applications, including semi-supervised node classification and link prediction, among others.

5 Graph Autoencoders

Auto-encoders (AE) and their variants have gained popularity in both unsupervised learning and generative modeling. These methods have been adapted to handle graph-structured data, serving two main purposes: learning network embeddings and generating new graphs. In this section, we will explore some notable models that exemplify these applications in the context of graph data.

Graph Auto-Encoder (GAE) [26] first uses GCNs to encode nodes in the graph. It leverages GCN to encode node structural information and node feature information at the same time. The encoder of GAE consists of two graph convolutional layers, which takes the form:

$$Z = enc(X, A) = conv_G(f(conv_G(A, X; \Theta_1)); \Theta_2) \tag{28}$$

where Z denotes the network embedding matrix of a graph, $f(\cdot)$ is a ReLU activation function and $conv_G(\cdot)$ is a graph convolutional layer defined by Equation 9. The decoder component of GAE is designed to unravel the relational information between nodes by reconstructing the graph's adjacency matrix from their learned embeddings, which is defined as:

$$\hat{A}_{v,u} = dec(\mathbf{z}_v, \mathbf{z}_u) = \sigma(\mathbf{z}_v^T \mathbf{z}_u) \tag{29}$$

where $\mathbf{z}_v$ denotes the embedding of node v and $\sigma(\cdot)$ denotes the sigmoid function. GAE* is trained by minimizing the negative cross entropy given the real adjacency matrix A and the reconstructed adjacency matrix $\hat{A}$. The task of directly reconstructing the graph adjacency matrix can result in overfitting, particularly due to the powerful representational capacity inherent to autoencoders. Graph Autoencoder (VGAE) [26] is a variational version of GAE to learn the distribution of data. VGAE optimizes the variational lower bound L:

$$L = \mathbb{E}_{q(Z|X,A)}[\log p(A|Z)] - KL[q(Z|X,A)||p(Z)] \tag{30}$$

where $KL(\cdot)$ is the Kullback-Leibler divergence function, and where $p(A_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = dec(\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^T \mathbf{z}_j)$, $q(Z|X,A) = \prod_{i=1}^{n} q(\mathbf{z}_i|X,A)$ with $q(\mathbf{z}_i|X,A) = N(\mathbf{z}_i|\mu, diag(\sigma_i^2))$. The last term $p(Z)$ is the prior distribution defined as:

$$p(Z) = \prod_{i=1}^{n} p(\mathbf{z}_i) = \prod_{i=1}^{n} N(\mathbf{z}_i|0, I). \tag{31}$$

The mean vector μ_i is the ith row of an encoder's outputs defined by Equation 24, similarly, $\log \sigma_i$ obtained with another encoder.

Unlike VGAE, which reconstructs the adjacency matrix from latent features for each node, Graph Variational Autoencoder (GraphVAE) [27] generates entire graphs in a single process. In GraphVAE, the existence of nodes and edges is modeled using independent random variables. Let $G = (A, E, F)$ be a graph specified with its adjacency matrix A, edge attribute tensor E, and node attribute matrix F. The goal is to learn an encoder and a decoder to map between the space of graphs G and their continuous embedding $\mathbf{z} \in \mathbb{R}^c$. With the posterior distribution $q_\phi(\mathbf{z}|G)$ defined by an encoder and the generative distribution $p_\theta(G|\mathbf{z})$ defined by a decoder, GVAE minimizes the upper bound on negative log-likelihood, i.e, $-\log p_\theta(G)$:

$$\mathcal{L}(\phi, \theta; G) = \mathbb{E}_{q_\phi(\mathbf{z}|G)}[-\log p_\theta(G)] + KL[q_\phi(\mathbf{z}|G)||p(\mathbf{z})] \tag{32}$$

where $p(\mathbf{z}) = N(0, I)$, The first term of $\mathcal{L}$, the reconstruction loss, enforces the high similarity of sampled generated graphs to the input graph G. The second term, KL-divergence, regularizes the code space to allow for sampling of z directly from $p(\mathbf{z})$ instead of from $q_\phi(\mathbf{z}|G)$ later. The dimensionality of z is usually fairly small so the autoencoder is encouraged to learn a high-level compression of the input instead of learning to simply copy any given input. While the regularization is independent of the input space, the reconstruction

loss must be specifically designed for each input modality. With a ConvGNN as the encoder and a simple multi-layer perceptron as the decoder, GraphVAE outputs a generated graph with its adjacency matrix, node attributes and edge attributes.

The Regularized Graph Variational Autoencoder (RGVAE) [28] enhances a graph variational autoencoder by adding validity constraints to the decoder's output distribution. The Molecular Generative Adversarial Network (MolGAN) [31] leverages convolutional GNNs [29], GANs [30], and reinforcement learning to create graphs with specified characteristics. MolGAN is composed of a generator that creates synthetic graphs and a discriminator that tries to differentiate these from real data. A reward network works alongside the discriminator to ensure the synthetic graphs meet certain criteria as determined by an external evaluator.

In summary, sequential methods convert graphs into linear sequences, which may result in the loss of structural details, especially in the presence of cycles. On the other hand, global methods synthesize an entire graph in one go, which can be impractical for large graphs due to the quadratic scaling ($O(n^2)$)of the output space in a graph autoencoder's case [32].

6 Applications

In this section, we categorize the applications of Graph Neural Networks (GNNs) into two main aspects: Structural and Non-structural.

Structural Applications These applications involve data with explicit and well-defined relational structures. They are commonly found in scientific research areas such as the modeling of physical and chemical systems. In the industrial domain, structural applications of GNNs include the use of knowledge graphs, traffic network analysis, and recommendation systems. These applications leverage the inherent graph structure of the data to improve the performance and interpretability of the models.

Non-structural Applications In contrast, non-structural applications deal with data where the relational structure is either implicit or entirely absent. This category primarily encompasses fields such as computer vision and natural language processing. In computer vision, GNNs can be used to capture the relationships between different regions of an image or between objects in a scene. In natural language processing, GNNs can help model the syntactic and semantic relationships between words or sentences in a text. Despite the lack of an explicit graph structure, GNNs can still be effectively applied by constructing graphs based on the data's inherent properties or through learned representations.

6.1 Structural Applications

6.1.1 Recommender Systems

Recommender systems are a prominent application area for Graph Neural Networks (GNNs), where both items and users are represented as nodes in a graph. These systems leverage the intricate relationships among users, items, and their content to provide high-quality recommendations. The essence of a recommender system lies in identifying whether an item is relevant to a user, effectively turning it into a problem of predicting relationships.

Several notable works have contributed to the development of graph-based recommender systems. van den Berg et al. [33] and Ying et al. [34] utilized Convolutional GNNs as encoders to address the issue of missing links between users and items. Monti et al. [35] combined Recurrent Neural Networks (RNNs) with graph convolutions to decode the process behind known ratings. In a more recent approach, Zhang et al. [36] introduced the Dynamic Graph Recommendation Network, a dynamic GNN model designed to extract users' preferences from evolving user-item graphs.

Furthermore, Huang et al. [37] proposed the Knowledge-aware Coupled Graph Neural Network, a knowledge-aware GNN model that integrates knowledge from both users and items to enhance recommendation accuracy. This network is capable of encoding high-order relations between users and items and utilizes mutual information to maintain awareness of the global graph structure. Additionally, it can capture dynamic, multi-typed user-item interaction patterns.

Gu et al. [38] took a different approach by developing a graph collaborative filtering model tailored for multi-behavior recommendations. This model accounts for both the similarities and differences in multiple user behaviors, providing a more nuanced understanding of user preferences. Through these various approaches, GNNs have proven to be highly effective in improving the performance of recommender systems by exploiting the rich relational information inherent in user-item interactions.

6.1.2 Chemistry

In the field of chemistry, Graph Neural Networks (GNNs) are employed to represent and analyze compounds or molecules, where atoms are depicted as nodes and chemical bonds as edges in a graph. Researchers focus on key tasks within these molecular graphs, including node classification, graph generation, and graph classification. Notable contributions include the learning of molecular fingerprints ([39] [40]), inferring protein interfaces [41], predicting molecular properties [42], and synthesizing chemical compounds [43].

Furthermore, recent studies highlight the challenges in manually identifying side effects from drug-drug interactions, which are rare occurrences [44]. The impracticality of testing all possible drug combinations and the difficulty in detecting side effects in small clinical trials exacerbate these challenges. To

address this, Feng et al. [45] employed a two-layer GCN to learn node embeddings, followed by a Deep Neural Network (DNN) to predict drug-drug interactions. Additionally, Rohani et al. [46] introduced a technique called Integrated Similarity-Constrained Matrix Factorization (ISCMF) for drug-drug interaction problems, which employs a matrix factorization constrained by similarities. This method identifies eight types of similarities, such as substructure, targets, and side effects, and integrates them to form an integrated similarity matrix for selection and analysis.

6.1.3 Social Networks

Several innovative approaches have been developed to enhance graph neural network (GNN) applications, particularly in the context of social networks and communication networks:

DeepInf [47]: This model integrates user-specific features and network structures through graph convergence and attention processes to predict social influence effectively.

SEAL [48]: As a link prediction framework, SEAL extracts local enclosing subgraphs for each target link and employs a GNN to learn generic graph-structured characteristics.

Multiple Conditional Network Embedding (MCNE) [49]: MCNE introduces a binary mask followed by an attentive network, combined with a message-passing-based GNN, to generate embeddings that capture multiple conditions.

Polysemic Node Embedding [50]: Observing the limitations of single vector representations in integrating networks, this research proposes a polysemic technique for embedding nodes, allowing for multiple node models to represent different aspects of a node's characteristics.

RCNN [51]: This method employs a convolutional neural network (CNN) approach, inspired by graph convolutional networks (GCNs), to rank nodes in complex communication networks, identifying critical nodes (such as super spreaders) based on their influence and connectivity.

These advancements highlight the ongoing development of GNN techniques to address various challenges in network analysis, from predicting social influence to identifying key nodes in communication networks.

6.2 Non-structural Applications

6.2.1 Computer Vision

Graph Neural Networks (GNNs) have a variety of applications in computer vision, such as creating scene graphs, classifying point clouds, and recognizing actions. By identifying the semantic links between objects, GNNs help in interpreting visual scenes with greater depth. Models like those by Yang et al. [52] analyze images as graphs with nodes representing objects and edges representing their semantic relationships. Scene graphs can also be used to generate lifelike images from text descriptions, as shown by Johnson et al. [53]. In the context

of LIDAR sensing, point clouds are analyzed to understand the environment. These point clouds are sets of three-dimensional points generated from LIDAR scans. Studies by Wang et al. [54], Landrieu and Simonovsky [55] have used Convolutional GNNs to explore the structure of point clouds by transforming them into graphs. Additionally, GNNs are increasingly being used in other areas of computer vision, such as human-object interaction [56], few-shot image classification [57], and semantic segmentation [58].

6.2.2 Natral Language Processing

GNNs have gained dramatic attention recently, and becoming more popular in text Categorization has been a focus in much of the existing literature. However, a recent study by Li et al. [61] stands out as it utilizes both meta-learning and Graph Neural Networks (GNNs) for cross-lingual sentiment classification, albeit using GNN primarily as a tool for meta-learning. In contrast, most previous work has been concentrated on monolingual text classification. The primary challenge in this domain is to bridge the semantic and syntactic gap between different languages. To address this, the majority of techniques aim to identify semantic similarities across languages and learn a language-agnostic representation for documents written in multiple languages [62]. This approach is exemplified by state-of-the-art (SOTA) multilingual pre-trained language models [63], which leverage large-scale multilingual corpora to pre-train transformer-based neural networks.

6.3 Datasets

We have organized some popular datasets into five categories: citation networks, web graphs, social networks, communication networks, and biochemical networks. These are detailed in Table 2, located in the Appendix.

7 Challenges and Future Directions

The Model Depth Problem The successful application of LLMS has made it tempting to stack traditional deep learning neural networks with hundreds of layers for enhanced performance. However, experiments conducted by Li et al. [64] reveal that stacking multiple GCN layers leads to over-smoothing, a phenomenon where the feature representation of each node converges exponentially to the same value as the model's depth increases. This issue has confined most GNNs to within three layers. Although some researchers have managed to address this problem [65], it remains the most significant limitation of GNNs.

Higher Structures and Topological Learning In complex networks, higher-order structures such as motifs, graphlets, and topological constructs like simplicial complexes and cell complexes play a crucial role, particularly in characterizing protein-protein interactions in biological applications. While Graph Neural

Networks (GNNs) primarily capture binary relations between vertices, topological data, including interactions of edges, triangles in meshes, or cliques, arise naturally in a variety of novel applications. These applications include complex physical systems, traffic forecasting, social influence, protein interaction, molecular design, visual enhancement, recommendation systems, and epidemiology. To effectively model such data, it is essential to go beyond graphs and consider the qualitative spatial properties that remain unchanged under geometric transformations. In other words, the topology of data needs to be considered to develop neural network architectures capable of extracting semantic meaning from complex data. Recently, there have been efforts to address this challenge, such as the Combinatorial Complexes Neural Network (CCNN) proposed by Hajij et al. [66], which utilizes applied algebraic topology and unifies graphs, simplicial and cell complexes, and hypergraphs as special cases. This innovative approach, along with other advances in this area such as [67, 68, 69], underscores the potential of leveraging topological methods in neural networks and invites researchers and practitioners to further explore and contribute to this burgeoning field.

Dynamic Graphs In numerous real-world applications such as traffic flow forecasting, rumor detection, and link prediction in recommender systems, there is a need for graphs that can handle time-varying topology and/or attributes to effectively model dynamic systems. This necessity has led to the emergence of Dynamic Graph Neural Networks (DGNNs), a relatively recent area of research compared to neural networks designed for learning on sequences and static graphs. The foundational DGNN models were introduced in 2018 [70] and 2019 [71] for discrete and continuous cases, respectively. Over the years, various terms have been used in the literature to refer to graphs with evolving structures and attributes, including dynamic graphs [72, 73], temporal graphs [74, 75, 76], evolving graphs [77, 78], time-varying graphs [79, 80], time-dependent graphs [81, 82]. These terms represent conceptual variants describing the same principles, with the diversity of terminology stemming from the interest of different scientific communities in this kind of model, as well as the field's relative youth. Despite recent advances, developing GNN models that can effectively handle continuous-time graphs represented as streams of node- or edge-wise events remains an open challenge in the field [83].

Scalability Trade-off The scalability of GNNs is gained at the price of corrupting graph completeness. Whether using sampling or clustering, a model will lose part of the graph information. By sampling, a node may miss its influential neighbors. By clustering, a graph may be deprived of a distinct structural pattern. How to trade off algorithm scalability and graph integrity could be a future research direction.

8 Conclusion

In reviewing the landscape of Graph Neural Networks (GNNs), this survey has highlighted their impressive adaptability and the breadth of their applications, from social networks and recommender systems to chemistry and computer vision. We have delved into the technical details and theoretical aspects of multiple GNNs, exploring both spectral and spatial approaches, as well as other architectures such as Graph Attention Networks and Graph Autoencoders. Concurrently, we have addressed the challenging issues of model depth, adaptability to dynamic graphs, and the trade-off balance of scalability.

The future of GNNs is vibrant and promising, with ample room for innovation. As research continues to overcome current limitations and leverages the unique strengths of GNNs, these networks are poised to become even more integral in extracting meaningful insights from complex data.

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] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008, 2017.

[4] 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.

[5] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Mathematics of Computation, vol. 19, no. 90, pp. 297–301, 1965.

[6] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in Proc. of NIPS, 2016, pp. 3844–3852.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. of ICLR, 2017.

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

[9] D. Bacciu, F. Errica, and A. Micheli, "Contextual graph markov model: A deep and generative approach to graph processing," in Proc. of ICML, 2018.

[10] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in Proc. of NIPS, 2016, pp. 1993–2001.

[11] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in Proc. of ICLR, 2018.

[12] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in Proc. of ICML, 2017, pp. 1263–1272.

[13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks," in Proc. of ICLR, 2019.

[14] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in Proc. of NeurIPS, 2018, pp. 4801–4811.

[15] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4438–4445, 2018.

[16] B. Weisfeiler and A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," Nauchno- Technicheskaya Informatsia, vol. 2, no. 9, pp. 12–16, 1968.

[17] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1365–1374. ACM, 2015.

[18] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Proc. of NIPS, 2017, pp. 1024–1034.

[19] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, "Invariant and equivariant graph networks," in ICLR, 2019.

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[21] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, pp. 123–135, 2017.

[22] A. Vaswani, N. Shazeer, N. Parmar, L. Jones, J. Uszkoreit, A. N. Gomez, L. Kaiser, "Attention is all you need," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pp. 5998–6008, 2017.

[23] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[24] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI Blog, 2018. [Online].

[25] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in Proc. of ICLR, 2017.

[26] T. N. Kipf and M. Welling, "Variational graph auto-encoders," NIPS Workshop on Bayesian Deep Learning, 2016.

[27] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in ICANN. Springer, 2018, pp. 412–422.

[28] T. Ma, J. Chen, and C. Xiao, "Constrained generation of semantically valid graphs via regularizing variational autoencoders," in Proc. of NeurIPS, 2018, pp. 7110–7121.

[29] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in ESWC. Springer, 2018, pp. 593–607.

[30] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in Proc. of NIPS, 2017, pp. 5767–5777.

[31] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models, 2018.

[32] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[33] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," stat, vol. 1050, p. 7, 2017.

[34] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in Proc. of KDD. ACM, 2018, pp. 974–983.

[35] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in Proc. of NIPS, 2017, pp. 3697–3707.

[36] W. Wang, F. Wei, L. Nie, X. He, X. Wang, T.-S. Chua, and R. Hong, "Dynamic Graph Neural Networks for Sequential Recommendation," in Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 910-918.

[37] C. Gao, X. He, D. Gan, X. Wang, F. Feng, Y. Li, and T.-S. Chua, "Knowledge-aware Coupled Graph Neural Network for Social Recommendation," in Proceedings of The Web Conference 2020, 2020, pp. 2335-2341.

[38] Y. Gu, J. Chang, Z. Cheng, Z. Liu, J. Chen, and M. Kankanhalli, "Learning Intents behind Interactions with Knowledge Graph for Recommendation," in Proceedings of The Web Conference 2020, 2020, pp. 1901-1911.

[39] Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R.P. (2015). Convolutional networks on graphs for learning molecular fingerprints.

[40] Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. Journal of Computer-Aided Molecular Design, 30(8), 595-608.

[41] Fout, A.M. (2017). Protein interface prediction using graph convolutional networks. PhD thesis, Colorado State University.

[42] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., and Dahl, G.E. (2017). Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning (Vol. 70, pp. 1263-1272).

[43] You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation.

[44] Bansal, M., Yang, J., Karan, C., Menden, M.P., Costello, J.C., Tang, H., Xiao, G., Li, Y., Allen, J., Zhong, R., et al. (2014). A community computational challenge to predict the activity of pairs of compounds. Nature Biotechnology, 32(12), 1213-1222.

[45] Feng, Y.-H., Zhang, S.-W., and Shi, J.-Y. (2020). Dpddi: a deep predictor for drug–drug interactions. BMC Bioinformatics, 21(1), 1-15.

[46] Rohani, N., Eslahchi, C., and Katanforoush, A. (2020). Iscmf: integrated similarity-constrained matrix factorization for drug–drug interaction prediction. Network Modeling Analysis in Health Informatics and Bioinformatics, 9(1), 1-8.

[47] Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K., and Tang, J. (2018b). Deepinf: Social influence prediction with deep learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 2110-2119). Association for Computing Machinery.

[48] Zhang, M., and Chen, Y. (2018). Link prediction based on graph neural networks. In Advances in Neural Information Processing Systems (Vol. 31, pp. 5165-5175).

[49] Wang, H., Xu, T., Liu, Q., Lian, D., Chen, E., Du, D., Wu, H., and Su, W. (2019c). MCNE: An end-to-end framework for learning multiple conditional network representations of social network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1064-1072). Association for Computing Machinery, New York, NY, USA.

[50] Liu, N., Tan, Q., Li, Y., Yang, H., Zhou, J., and Hu, X. (2019c). Is a single vector enough? Exploring node polysemy for network embedding. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 932-940). Association for Computing Machinery, New York, NY, USA.

[51] Yu, E.-Y., Wang, Y.-P., Fu, Y., Chen, D.-B., and Xie, M. (2020). Identifying critical nodes in complex networks via graph convolutional networks. Knowledge-Based Systems, 198, 105893.

[52] Yang, J., Lu, J., Lee, S., Batra, D., and Parikh, D. (2018). Graph R-CNN for scene graph generation. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 670-685). Springer, Cham.

[53] Johnson J, Gupta A, Fei-Fei L (2018) Image generation from scene graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1219–1228. IEEE, Salt Lake City, UT, USA

[54] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM (2019a) Dynamic graph CNN for learning on point clouds. Acm Trans Graph 38(5):1–12

[55] Landrieu, L., and Simonovsky, M. (2018). Large-scale point cloud semantic segmentation with superpoint graphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4558-4567). IEEE, Salt Lake City, UT, USA.

[56] Qi S, Wang W, Jia B, Shen J, Zhu S-C (2018) Learning human-object interactions by graph parsing neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 401–417. Springer Cham

[57] Garcia, V., and Bruna, J. (2017). Few-shot learning with graph neural networks. arXiv preprint arXiv:1711.04043.

[58] Qi, X., Liao, R., Jia, J., Fidler, S., and Urtasun, R. (2017). 3D graph neural networks for RGBD semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 5199-5208).

[59] Yao, L., Mao, C., and Luo, Y. (2019). Graph convolutional networks for text classification. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (Vol. 33, pp. 7370-7377).

[60] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 7370-7377.

[61] Z. Li, M. Kumar, W. Headden, B. Yin, Y. Wei, Y. Zhang, and Q. Yang, "Learn to cross-lingual transfer with meta graph learning across heterogeneous languages," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 2020, pp. 2290-2301.

[62] X. Chen, Y. Sun, B. Athiwaratkun, C. Cardie, and K. Weinberger, "Adversarial deep averaging networks for cross-lingual sentiment classification," Transactions of the Association for Computational Linguistics, vol. 6, pp. 557-570, 2018.

[63] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota, 2019, pp. 4171-4186.

[64] Q. Li, Z. Han, and X.-M.Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in Proc. of AAAI, 2018.

[65] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A Survey on Oversmoothing in Graph Neural Networks," arXiv:2303.10993, 2023.

[66] M. Hajij, G. Zamzmi, T. Papamarkou, N. Miolane, A. Guzmán-Sáenz, K. Natesan Ramamurthy, T. Birdal, T. Dey, S. Mukherjee, S. Samaga, N. Livesay, R. Walters, P. Rosen, and M. Schaub, "Topological Deep Learning: Going Beyond Graph Data," arXiv preprint arXiv:2206.00606v3, 2023.

[67] F. Battiston et al., "Networks beyond pairwise interactions: structure and dynamics," Physics Reports, vol. 874, pp. 1-92, 2020.

[68] L. Torres et al., "The why, how, and when of representations for complex systems," SIAM Review, vol. 63, no. 3, pp. 435-485, 2021.

[69] C. Bick et al., "What are higher-order networks?" arXiv preprint arXiv:2104.11329, 2021.

[70] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in Proceedings of the International Conference on Neural Information Processing, 2018, pp. 362-373.

[71] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in Proceedings of the International Conference on Learning Representations, 2019.

[72] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," arXiv preprint arXiv:2006.10637, 2020.

[73] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 5363-5370.

[74] U. Singer, I. Guy, and K. Radinsky, "Node embedding over temporal graphs," arXiv preprint arXiv:1903.08889, 2019.

[75] A. Rehman, M. Ahmad, and O. Khan, "Exploring accelerator and parallel graph algorithmic choices for temporal graphs," in Proceedings of the Eleventh International Workshop on Programming Models and Applications for Multicores and Manycores, 2020, pp. 1-10.

[76] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," arXiv preprint arXiv:2002.07962, 2020.

[77] B. Bahmani, R. Kumar, M. Mahdian, and E. Upfal, "Pagerank on an evolving graph," in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 24-32.

[78] C. Song, K. Shu, and B. Wu, "Temporally evolving graph neural network for fake news detection," Information Processing and Management, vol. 58, 102712, 2021.

[79] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2826-2830.

[80] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro, "Time-varying graphs and dynamic networks," International Journal of Parallel, Emergent and Distributed Systems, vol. 27, pp. 387-408, 2012.

[81] Y. Wang, Y. Yuan, Y. Ma, and G. Wang, "Time-dependent graphs: Definitions, applications, and algorithms," Data Science and Engineering, vol. 4, pp. 352-366, 2019.

[82] H. Huang, A. Savkin, and C. Huang, "Reliable path planning for drone delivery using a stochastic time-dependent public transportation network," IEEE Transactions on Intelligent Transportation Systems, vol. 22, pp. 4941-4950, 2020.

[83] L. Yang, C. Chatelain, and S. Adam, "Dynamic Graph Representation Learning with Neural Networks: A Survey," IEEE Access, vol. PP, pp. 1-1, 2024, doi: 10.1109/ACCESS.2024.3378111.

[84] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in Proc. of IJCNN, vol. 2. IEEE, 2005, pp. 729–734.

[85] 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, 2009.

[86] C. Gallicchio and A. Micheli, "Graph echo state networks," in IJCNN. IEEE, 2010, pp. 1–8.

[87] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in Proc. of ICLR, 2014.

[88] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," arXiv preprint arXiv:1506.05163, 2015.

[89] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. of ICLR, 2017.

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

[91] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in Proc. of ICML, 2016, pp. 2014–2023.

APPENDIX

Dataset

Table 2: Summary of available benchmark datasets for GNN experimentation

Category	Dataset	Edge Type	Heterogeneity	Classes	Weights
Citation Networks	Cora	$\rightarrow$	N	7	Unweighted
	Citeseer	$\rightarrow$	N	6	Unweighted
	PubMed	$\rightarrow$	N	3	Weighted
	DBLP	$\rightarrow$	N	4	Weighted
	Patents	$\rightarrow$	N	-	Unweighted
	HepPh	$\rightarrow$	N	-	Unweighted
Web graphs	Cornell	$\rightarrow$	Y	5	Unweighted
	Texas	$\rightarrow$	Y	5	Unweighted
	Washington	$\rightarrow$	Y	5	Unweighted
	Google	$\rightarrow$	Y	-	Unweighted
	Stanford	$\rightarrow$	Y	-	Unweighted
Social Networks	Karate club	$\leftrightarrow$	Y	2	Unweighted
	Reddit	$\rightarrow$	Y	41	Weighted
	BlogCatalog	$\rightarrow$	Y	39	Unweighted
	Facebook	$\leftrightarrow$	Y	-	Unweighted
	Youtube	$\leftrightarrow$	Y	8385	Unweighted
Communication Networks	email-EuAll	$\rightarrow$	Y	-	Unweighted
	Enron	$\leftrightarrow$	Y	-	Unweighted
	wiki-Talk	$\rightarrow$	Y	-	Unweighted
	f2f-Resistance	$\rightarrow$	Y	-	Weighted
Bio-chemical	MUTAG	$\leftrightarrow$	N	2	Unweighted
	PROTEINS	$\leftrightarrow$	N	2	Unweighted
	PPI	$\rightarrow$	Y	121	Weighted
	NCI-1	$\leftrightarrow$	N	2	Unweighted