Graph Neural Networks and Their Variants: A Survey

Authors: Chenlong Liu, Liu Chenlong

Date: 2024-01-04T00:00:00+00:00

Abstract

Through detailed reading and comprehension of several papers, I have completed this survey paper, conducting an in-depth analysis and comprehensive investigation into the fundamental concepts, core architectures, and application domains of Graph Neural Networks (GNN) and several of its variants, including Graph Convolutional Networks (GCN), Graph Sampling Neural Networks GraphSAGE, Attention Graph Neural Networks GAT, Graph Recurrent Networks GGNN, and Graph Recurrent Neural Networks HGNN models. I have summarized the research methodologies employed by the authors of these papers and the functionalities and applications achieved by the models they developed, and provided my own insights regarding the future development and research directions of GNN.

Full Text

Preamble

This review paper provides a comprehensive analysis of Graph Neural Networks (GNN) and several of its variants, including Graph Convolutional Networks (GCN), Graph Sample and Aggregation (GraphSAGE), Graph Attention Networks (GAT), Gated Graph Neural Networks (GGNN), and Hypergraph Neural Networks (HGNN). Through detailed examination of key literature, this work investigates the fundamental concepts, core architectures, and application domains of these models. The paper synthesizes the research methodologies employed by authors and the functionalities achieved by their models, while offering perspectives on future development directions and research trajectories for GNNs.

Keywords: machine learning, deep learning, graph neural networks

1. Introduction

In the information age, we increasingly confront complex data networks where entities and relationships exhibit intricate graph structures. Traditional neural network models often struggle to capture and comprehend critical information within such graph data. Consequently, Graph Neural Networks have emerged as a compelling research area. By examining the developmental background and fundamental definitions of GNNs, we can understand why they have garnered significant attention for processing complex graph-structured data. Across diverse application domains, GNNs provide a novel paradigm for understanding and uncovering latent patterns and regularities in graph data. For instance, in e-commerce, graph-based learning systems leverage user-product interactions to deliver highly accurate recommendations; in chemistry, molecules are modeled as graphs to determine their biological activity for drug discovery; and in citation networks, papers linked through citation relationships require classification into distinct groups [7]. The advent and evolution of GNNs have not only advanced frontier research in computer science, artificial intelligence, and data science, but have also provided unprecedented solutions for graph data analysis in practical applications. [Figure 1: see original paper] illustrates the recent development trajectory of GNNs.

This paper focuses on the theoretical foundations, model architectures, and various derived variants of GNNs, each demonstrating unique advantages and application value in specific graph data processing scenarios. For example, GCN propagates information through convolutional operations on graphs; GraphSAGE achieves node feature aggregation through graph sampling strategies; GAT flexibly captures relationships between nodes using attention mechanisms; GGNN specializes in handling dynamic graph data; and HGNN excels in hypergraph structures. These diverse variants provide a range of options for addressing graph-related problems, enabling GNNs to leverage their strengths across different application contexts. Detailed discussion of the original GNN formulation appears primarily in Section 2, while variants are examined in Sections 3–7. Section 8 discusses future research directions in GNN-related fields, and Section 9 concludes this review.

2.1 Background

The history of Graph Neural Networks can be traced back to graph theory and early neural network research in the 1970s and 1980s, when these fields developed independently. Early neural networks focused primarily on processing vector data, while graph theory addressed network and relational data. Although the concept of GNNs was not yet distinct, researchers began attempting to combine graph theory with neural network methodologies. In 1997, Sperduti et al. [8] first applied neural networks to directed acyclic graphs, forming an early prototype of GNNs. The concept was formally proposed by Gori et al. [9] in 2005. However, due to various factors, substantial development of GNNs did not commence until around 2010. Prior to this, computational resources

were relatively limited, particularly for deep learning models requiring large-scale training. GNNs typically need to process extensive graph-structured data, demanding more powerful computational capabilities. Additionally, large-scale, rich graph datasets were scarce in the early days, and deep learning methods heavily rely on extensive training data. Moreover, awareness of the potential of graph-structured data in practical applications was relatively low, as mainstream deep learning applications concentrated on images, speech, and natural language processing, resulting in insufficient motivation for GNN research. Over time, these challenges were gradually resolved. Following the formal proposal of the GNN model by Scarselli et al. in 2009 [1], researchers revisited this technology and recognized the powerful potential of graph-structured data in deep learning, establishing GNNs as a prominent research direction in contemporary deep learning.

2.2 Definitions

We first briefly elaborate on the definition of the graph data structure, then explain the fundamental learning and training definitions of GNN models.

A graph is denoted as $G = (V, E)$, where V represents the set of nodes and E the set of edges. Let $v_i \in V$ denote a node and $e_{ij} \in (v_i, v_j)$ an edge. The neighborhood of node v can be expressed as $N(v) = \{u \in V | (v, u) \in E\}$. Generally, graphs are categorized as follows:

- **Directed/Undirected Graphs.** Directed graphs have edges pointing from one node to another. Undirected graphs can be considered a special case of directed graphs where connected nodes have bidirectional edges. For undirected graphs, the degree of node v_i represents the number of edges incident to v_i, expressed as $D(v_i) = \{e_{ij} \in E | v_i \in (v_i, v_j)\}$. For directed graphs, degree splits into in-degree and out-degree, representing the number of edges pointing to and originating from v_i, respectively, expressed as $D_{in}(v_i) = \{e_{ij} \in E | v_i = v_j\}$ and $D_{out}(v_i) = \{e_{ij} \in E | v_j = v_i\}$.
- **Homogeneous/Heterogeneous Graphs.** Homogeneous graphs consist of one type of node and edge, while heterogeneous graphs contain multiple types of nodes or edges.
- **Sparse/Dense Graphs.** Graphs with relatively few edges are termed sparse, while those with many edges are dense.
- **Hypergraphs.** Hypergraphs generalize graphs, where a single edge can connect any number of vertices.

Given graph data, the core idea of GNN node representation learning is to iteratively aggregate feature information from neighborhoods during propagation and integrate this aggregated information with the current central node's representation. Architecturally, GNNs stack multiple propagation layers comprising aggregation and update operations [1]. The propagation formula is defined as:

$$= AAAAAAAAAAAAAAAAAAAAll(huu, \forall uu \in NNvv)$$

where $h_u^{(l+1)}$ represents node u's representation at layer l, and Aggregator$_l$ and Updater$_l$ denote the aggregation and update functions at layer l, respectively. The iterative stacking of these processes forms the complete graph neural network. Through successive aggregation and update iterations, each layer's node representation gradually incorporates more neighborhood information, enabling the model to capture high-level features and structural information. This iterative process effectively propagates information, allowing each node's representation to converge within the global context.

GNN training primarily involves two tasks: node classification and link prediction. In node classification, each node is assigned a class label, and the model aims to learn node representations that correctly classify nodes. In link prediction, the model predicts whether connections exist between nodes, enabling exploration of implicit relationships and applications like recommendation. These tasks are trained via supervised learning, driving GNNs to learn node representations that better capture graph structure and information for meaningful predictions and inferences.

GNNs have demonstrated significant advantages in advancing graph data research and applications. Techniques such as local information aggregation, learning node embeddings, and iterative message passing have enabled successful applications in social network analysis, recommendation systems, and bioinformatics. In recommendation systems, for example, complex interactions between users and items can be modeled as graphs where nodes represent users and items and edges represent interactions. GNNs effectively capture high-order patterns in user behavior and item associations, enabling personalized recommendations through learned node embeddings. Compared to traditional methods, GNNs more flexibly handle heterogeneous information, adapting to different node and edge types, which yields more accurate and personalized recommendations [10]. [Figure 2: see original paper] illustrates representative graph structures in recommendation systems.

However, traditional GNN models gradually show limitations when facing increasingly complex graph structures. For instance, GNNs typically assume fixed node neighborhood relationships, making them ill-adapted for dynamic graphs and hypergraphs. They primarily focus on homogeneous graphs, lacking generality for heterogeneous graphs. Furthermore, since each layer's information aggregation is limited to node neighborhoods, GNNs inefficiently capture long-range dependencies, severely impacting performance on certain tasks. These limitations have motivated the development of GNN variants.

3. GCN (Graph Convolutional Network)

GCN can be understood as a GNN that performs convolutional operations on graph structures. Compared to traditional GNNs, GCN incorporates improvements in mathematical formulation and gradient propagation, making it more robust and efficient for learning graph data representations [2]. These improvements include:

- **Local Information Aggregation.** While GNNs aggregate information from neighborhood nodes, GCN employs Laplacian convolution for aggregation, emphasizing contributions from local neighbors when learning node representations.
- **Gradient Propagation.** GCN's symmetric normalization operation facilitates stable gradient propagation and enables training of deeper networks, making it easier to handle gradient flow compared to GNNs.
- **Parameter Sharing.** GCN's convolutional kernels are shared, meaning each node uses the same weight matrix at every layer. This parameter sharing reduces model parameters and improves training efficiency.

GCN's learning process approximates first-order eigendecomposition of the graph Laplacian, iteratively aggregating neighborhood information. Specifically, it performs graph convolution through aggregation and update steps:

$$vv \in NNvvAAvvvvddvvvvddvvvv \ = 1 \quad djj = \sum kajk = \delta\delta(WW(ll+1)$$

where $\delta(\cdot)$ is a nonlinear activation function (e.g., ReLU), d is the diagonal matrix of a, $W^{(l)}$ is the learnable transformation matrix at layer l, and $\tilde{a}_{vj}$ represents neighborhood weights ($\tilde{a}_{vv} = 1$), while $d_{jj} = \sum_k a_{jk}$.

GCN training involves input, parameter initialization, forward propagation, loss computation, backward propagation, and iteration. Forward propagation incorporates the above graph convolution operation, considering the normalized adjacency matrix and node degree influences to update node representations.

GCN is suitable for irregular graph structures where node connections can be arbitrary, unconstrained by regular networks. It supports end-to-end learning, optimizing model parameters through backpropagation to learn node representations directly from raw data without manual feature engineering. Moreover, GCN's graph convolution operation is interpretable, as each layer's node representations can be interpreted as linear combinations of local neighborhood features in the graph structure. These advantages enable GNNs to fully leverage deep learning for graph-structured data, yielding widespread applications in node classification, link prediction, recommendation systems, community detection, bioinformatics, semantic segmentation, knowledge graphs, and traffic forecasting.

4. GraphSAGE (Graph Sample and Aggregation)

Before GraphSAGE, deep learning methods for graph structures required all graph nodes to be present during embedding training. These methods were essentially transductive and could not generalize to unseen nodes. GraphSAGE breaks this limitation. As a general, inductive framework, GraphSAGE efficiently generates node embeddings using node feature information (e.g., text attributes). Instead of training separate embeddings for each node, it learns a function that generates embeddings by sampling and aggregating from local neighborhoods [3].

GraphSAGE introduces node sampling, randomly selecting a fixed number of neighborhood nodes for each node. This flexible sampling mechanism reduces computational and storage complexity when processing large-scale graphs, improving scalability. Sampled neighbor features are integrated through aggregation mechanisms using mean/sum/max pooling aggregation and concatenation, implemented as follows:

$$= AAAAAAAAAAAAAAAAAAAAll(huu(ll+1) \cdot hvv = \delta\delta(WW, \forall uu \in NNvv) \oplus nnvv$$

where Aggregator_l denotes the aggregation function at layer l, $\delta(\cdot)$ is a nonlinear activation function, $W^{(l)}$ is the learnable transformation matrix at layer l, and $\oplus$ represents vector concatenation.

GraphSAGE's flexibility and scalability make it an effective method for large-scale graph data. It preserves graph structural information while reducing computational complexity through neighbor sampling and information aggregation, making learning on large graphs more feasible. It has achieved successful applications in multiple domains:

- **Node Classification.** GraphSAGE has achieved significant success in node classification tasks. By learning node representations, the model effectively characterizes relationships between nodes for accurate classification.
- **Link Prediction.** In link prediction, GraphSAGE learns node representations to accurately predict whether connections exist between nodes, providing effective solutions for social networks and recommendation systems.
- **Recommendation Systems.** GraphSAGE is widely applied in recommendation systems. By learning complex relationships between users and items, the model improves recommendation accuracy and enables personalized suggestions.
- **Community Detection.** GraphSAGE also performs well in community detection. By learning node representations, the model effectively discovers tightly connected node groups in graphs.

5. GAT (Graph Attention Network)

GAT is a neural network architecture operating on graph-structured data that employs masked self-attention layers to address limitations of previous graph convolution or approximation-based methods. By stacking layers where nodes can attend to neighborhood features, GAT achieves the ability to assign different weights to different nodes in a neighborhood without requiring expensive matrix operations (e.g., inversion) or prior knowledge of graph structure. This approach addresses several key challenges of spectral-based GNNs and can be readily applied to both inductive and transductive problems [4].

A masked self-attention layer is an attention mechanism typically used for sequential data, such as text sequences in natural language processing. This layer allows the model to consider only information before the current position when computing attention, preventing information leakage. In self-attention, each position's output is a weighted sum of all positions' inputs, with weights derived from correlations between input positions. However, in many tasks, we prefer to prevent the model from seeing future information to avoid data leakage. The key to masked self-attention is introducing a mask matrix that blocks future information (setting it to negative infinity or zero), ensuring only preceding positions are considered when computing attention weights. Specifically, the masked self-attention process involves: (1) computing attention scores for each position's correlation with others; (2) applying a mask to set attention scores for future positions to negative infinity or zero; (3) applying softmax to convert attention scores to probability distributions; and (4) computing weighted sums of input sequence features using these weights to produce each position's output.

GAT assumes that neighbor node influences are neither identical nor predetermined by graph structure. Instead, it leverages attention mechanisms to differentiate neighbor contributions and update each node's vector by attending to its neighborhood:

$$vv \in NNvvexp(AAAAAAhvv\alpha\alpha vvvvhvv, \alpha\alpha vvvv = kk \in NNvvexp(AAAAAAhvv(ll{+}1) = \delta\delta(WW \oplus W(l)hj$$

where $\mathrm{Att}(\cdot)$ is an attention function, typically including LeakyReLU layers ($a^T[W^{(l)}h_v^{(l)} \oplus W^{(l)}h_j^{(l)}]$), with $W^{(l)}$ transforming node representations at the l-th propagation step and a being learnable parameters.

GAT's flexibility and expressiveness on graph data make it an important model in the GNN field. Through attention mechanisms, the model can flexibly focus on different contributions from neighborhood nodes rather than using fixed weights, allowing each node to dynamically compute weights with its neighbors. These advantages enable GAT to adapt to complex graph structures, handle diverse graph types and scales, and improve graph representation capacity. Consequently, GAT is widely applied in node classification, graph classification, link

prediction, community detection, and recommendation systems, demonstrating excellent performance across various applications.

6. GGNN (Gated Graph Neural Network)

GGNN is a graph-structured recurrent neural network based on the GNN model [1] that introduces gating mechanisms, demonstrating superior performance in processing dynamic graphs and temporal data compared to purely sequence-based models like LSTM, with more favorable inductive biases [5]. The gating mechanism operates as follows: (1) **Update gates** control the extent to which each node updates its state, regulating when new information is integrated; (2) **Reset gates** determine which previous state information should be forgotten or reset to better adapt to new inputs; and (3) **Hidden state updates** iteratively modify node hidden states based on gating computations, effectively capturing the evolution of node states.

GGNN's gating mechanisms enable effective handling of dynamic graphs by allowing flexible adjustment of node states to accommodate graph structure changes. The model is highly suitable for temporal data modeling, as gating appropriately updates and forgets information across time steps to capture long-term dependencies in temporal data. For sequence prediction tasks, such as forecasting future node states in graphs, gating mechanisms allow adaptive adjustment of attention to historical information, improving prediction performance.

Specifically, GGNN employs Gated Recurrent Units (GRU) in its update step:

$|NNvv| \sum vv \in NNvvGGNN$ executes recurrent functions across all nodes multiple times, creating scalabilit

7. HGNN (Heterogeneous Graph Neural Network)

HGNN is a hypergraph neural network framework for data representation learning that can encode high-order data correlations in hypergraph structures, demonstrating superiority in processing complex data and heterogeneous graphs. In this approach, HGNN employs a hyperedge convolution operation to process data correlations during representation learning, enabling effective implementation of traditional hypergraph learning processes through hyperedge convolution.

HGNN can learn hidden layer representations that consider high-order data structures, making it a general framework for accounting for complex data correlations [6].

Hyperedge convolution is an HGNN operation for processing high-order data correlations in hypergraphs. While traditional GNNs perform convolution on graph edges, hypergraphs contain hyperedges connecting multiple nodes, necessitating a convolution operation adapted to hypergraph structures. Hyperedge

convolution is designed to effectively capture and utilize high-order relationships in hypergraphs. Specifically, hyperedge convolution involves: (1) aggregating features of all nodes connected by each hyperedge to capture high-order relationships; (2) computing weights for nodes within hyperedges to reflect their contribution to hyperedge representation, typically learned via attention mechanisms or other weight allocation strategies; (3) updating hyperedge representations using computed weights through weighted summation of aggregated results; and (4) updating node representations based on updated hyperedge information through appropriate aggregation operations.

The hyperedge convolution layer is expressed as:

$$= DDvv(ll + 1)DDee$$

where $\delta(\cdot)$ is a nonlinear activation function, $W^{(l)}$ is the learnable transformation matrix at layer l, E is the hypergraph adjacency matrix, and D_e and D_v are diagonal matrices representing edge degrees and vertex degrees, respectively.

$$= \delta\delta(WW$$

The key to hyperedge convolution is enabling the model to effectively capture and propagate high-order data correlations in hypergraphs during representation learning. This is highly useful for processing multimodal data or data containing complex relationships. In hypergraph neural networks, hyperedge convolution is an effective mechanism for more comprehensively considering complex relationships between nodes, enhancing model expressiveness. This has led to applications in recommendation systems, knowledge graphs, bioinformatics, social network analysis, and e-commerce platforms, providing effective solutions for modeling and analyzing complex relational networks.

8. Future Research Directions

GNNs have demonstrated powerful performance in processing graph-structured data. However, as application scenarios expand and practical problems become more complex, GNNs continue to face challenges. Based on our analysis, future research directions will likely concentrate on the following key issues:

- **Interpretability.** Many GNN models exhibit excellent performance on graph data, but their black-box nature limits understanding of model decision-making processes. Future research will focus on designing more interpretable GNN models to reveal the intrinsic mechanisms of graph data learning, improving decision transparency and trustworthiness.
- **Scalability for Large-Scale Graph Data.** As graph data scales grow, existing GNN models may face computational and storage challenges. Future research will aim to design more scalable GNN algorithms and models

to accommodate large-scale graph processing requirements while maintaining efficient performance.

- **Heterogeneous Graph Data Processing.** Real-world data typically contains multiple node and edge types, yet most current GNN models focus on homogeneous graphs. Future research will explore more effective methods to adapt GNNs to heterogeneous graph structures and better integrate multi-source information, enhancing applicability in real-world scenarios.
- **Generalization Performance.** Despite strong performance on training sets, GNN generalization to unseen graph data remains challenging. Future research will explore methods to improve GNN generalization, such as introducing stronger regularization strategies and transfer learning, to better adapt to different graph structures.
- **Comparative Studies Across Variants.** Numerous GNN variants have emerged in recent years, but which models are more suitable for different applications remains an open question. Future research will deeply investigate similarities and differences among variants regarding interpretability, scalability, and generalization performance, providing more targeted guidelines for researchers in different fields.

9. Summary

The emergence of Graph Neural Networks marks a significant advancement in combining graph-structured data modeling with deep learning, providing powerful tools for processing complex relationships and network structures. Through detailed exploration of GNNs and their major variants—including GCN, GraphSAGE, GAT, GGNN, and HGNN—we have gained deep understanding of their respective strengths, limitations, and applications across different scenarios. GNN models have achieved remarkable success in social networks, recommendation systems, bioinformatics, knowledge graphs, and other domains, offering new perspectives for data modeling and analysis. However, we have also identified challenges in current research, including interpretability, scalability, heterogeneous graph processing, and generalization performance, which require further investigation and innovation.

Despite many successes, the potential research space for GNNs remains to be fully explored. We believe future work will enable deeper investigation of these models through innovative methods and techniques to address current challenges. For improving interpretability, researchers could introduce explainable attention mechanisms or conduct more in-depth explanatory studies. For enhancing scalability, more efficient GNN architectures or distributed computing strategies could be explored. For heterogeneous graph processing, more flexible model structures could be investigated to better integrate information from different node and edge types. Meanwhile, improving generalization perfor-

mance could involve introducing stronger transfer learning strategies or designing smarter regularization methods.

Intelligence and Machine Learning, Introduction [1] Franco S, Marco G, Chung A T, et al. The graph neural network model. [J]. IEEE transactions on neural networks, 2009, 20(1): 61-80. [2] KIPF T, WELLING M. Semi-Supervised Classification with Graph Convolutional Networks[J]. arXiv: Learning, arXiv: Learning, 2016. [3] HAMILTON WilliamL, YING Z, LESKOVEC J. Inductive Representation Learning on Large Graphs[J]. Neural Information Processing Systems, Neural Information Processing Systems, 2017. [4] LIU Z, ZHOU J. Graph Attention Networks[M/OL]//Synthesis Lectures on to Graph Neural Artificial Networks, 2020: 39-41. [5] LI Y, ZEMEL R, BROCKSCHMIDT M, et al. GATED GRAPH SEQUENCE NEURAL NETWORKS[J]. [6] FENG Y, YOU H, ZHANG Z, et al. Hypergraph Neural Networks[J/OL]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019: 3558-3565. [7] WU Z, PAN S, CHEN F, et al. A Comprehensive Survey on Graph Neural Networks[J/OL]. IEEE Transactions on Neural Networks and Learning Systems, 2021: [8] SPERDUTI A, STARITA A. Supervised neural networks for the classification of structures[J/OL]. IEEE Transactions on Neural Networks, 1997: 714-735. [9] GORI M, MONFARDINI G, SCARSELLI F. A new model for learning in graph domains[C/OL]//Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. 2006. [10] WU S, SUN F, ZHANG W, et al. Graph Neural Networks in Recommender Systems: A Survey[J/OL]. ACM Computing Surveys, 2023: 1-37.

Note: Figure translations are in progress. See original paper for figures.