A Comprehensive Investigation on Graph Neural Networks: Models, Applications, and Future Challenges

Xinyang Chen

Faculty of Engineering, University of New South Wales, Sydney, Australia

z5499873@ad.unsw.edu.au

Abstract. Graphs, as a kind of complex data structure, are getting wide attention these days, requiring more research on relative models to extract information and make predictions based on graphs. Especially, Graph Neural Network (GNN) shows high performance in many areas. This paper surveys both traditional and advanced GNN models, with their core principles and creative features. It also introduces real-life applications of GNN models, including social recommender systems, drug discovery and traffic prediction. Additionally, this paper discusses current bottlenecks of GNN and prospects of possible improvements. By reviewing GNN from different aspects, this paper tries to provide an entire perspective on this direction for readers. People now encounter many more types of graphs than before, ranging from urban road networks to social platforms, and even molecular structures in biology. By connecting these scenarios, this paper emphasizes the growing importance of GNN research and its potential to transform both scientific discovery and industrial applications.

Keywords: Graph Neural Network, machine learning, deep learning.

1. Introduction

People now have seen much more types of graphs than ever before. For example, road networks in cities are extending, while social networks are connecting individuals. On a minuscule scale, molecules as networks of atoms are dynamically changing in a three-dimensional space. Abstractly, entities and relationships can be represented by knowledge graphs. Graphs can always display complex structures in a concise way. However, it is also difficult to extract information from graphs or find regular patterns. Traditional deep learning framework, such as the Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN), mainly pays attention to Euclidean images or sequential images, being tricky to manipulate irregular, high-dimensional and large-scale graphs [1]. Recently, research on Graph Neural Networks (GNN) is becoming popular, showing great advantages of GNN in graph learning and processing [2].

GNN has become a popular research topic of graph learning since the presentation of Graph Convolutional Network (GCN) [3], Graph Sample and Aggregate (GraphSAGE) [4] and Graph Attention Networks (GAT) [5], with wide applications in drug discovery, knowledge graph and recommendation system. After that, research on GNN evolves surprisingly fast, and more powerful models and frameworks appear. Graph Isomorphism Network (GIN) [6] contributes to deeper understanding of graph representational properties and limitations. After that, a series of research on Graph Transformer introduce transformer architecture into GNN, providing another way for graph learning [7]. However, GNN still faces theoretical bottlenecks of over-smoothing and over-squashing. Analysis on these challenges provides possible solutions for improvements in the future [8]. Training in GNN is also a hot topic. Graph Contrastive Learning (GraphCL) [9] explores ways to learn unsupervised graph data. GraphMAE [10] is presented as a masked graph autoencoder for generative self-supervised graph learning, and GraphMAE [11] improves robustness to disturbance in features. These training methods significantly raise the performance of GNN.

GNN now has numerous applications. In physics, MeshGraphNet [12] is a framework for mesh-based simulation based on GNN. For protein structure prediction, AlphaFold [13] incorporates physical and biological knowledge for prediction and extends to atomic accuracy. Message passing neural network can also be used in building PDE solvers [14].

This paper explains traditional GNN models with basic concepts and propagation rules. After that, cutting-edged research on GNN is introduced with analysis of performance and problems. Then, various application of GNN is listed to help construct better models. Finally current challenges and possible improvement of current GNN models are discussed.

2. Method

2.1. Traditional GNN Models

2.1.1. GCN

GCN is one of the earliest Graph Neural Networks that brings graph convolution to the mainstream of research, proposed by Kipf and Welling in 2017 [3]. They successfully applied the first-order approximation to the graph convolution model, that can be used for semi-supervised classification of nodes in a graph. The forward propagation rule they proposed is:

$$H^{[l+1]} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{[l]}W^{[l]}\right) \tag{1}$$

In which, H is the activation in each layer, A is the adjacent matrix added with self-connected matrix $diag(1)$, D is the degree matrix of A, W is the trainable weight matrix in each layer, and σ is the activation function. This concise formula is motivated by approximation of Chebyshev Polynomial, that is used to calculate convolution of normalized graph Laplacian in Fourier domain, powering GCN as a fast and scalable neural network based on fixed graph information.

2.1.2. GAT

In GCN, it can be noticed that adjacent matrix is symmetric, which will give the same weights to every neighbour of nodes. What GAT tries to do is to provide different weights to first-order neighbours, that can also be learned by the models [5]. Self-attention, as the shared attentional mechanism in GAT, can project pairs of feature vectors to attention coefficients. Final output features can be calculated by normalizing attention coefficients as weights by a softmax function. The forward propagation rule of GAT is written below:

$$\alpha_{ij} = \frac{\exp\left(LeakyReLU\left(a\left(Wh_i, Wh_j\right)\right)\right)}{\sum_k \exp\left(LeakyReLU\left(a(Wh_i, Wh_k)\right)\right)} \tag{2}$$

$$h'_i = \sigma\left(\sum_j \alpha_{ij} W h_j\right) \tag{3}$$

In which, h is input feature vector, h' is output feature vector, α is self-attention mechanism coefficient, and W is weighted matrix. In practice a can be a learnable vector-like parameter, calculating attention coefficients by multiplying with the concatenation of input vectors. GAT also allows multi-head attention to stabilize the learning process with K independent attention mechanisms. Extended to calculate embeddings for new nodes, attention coefficients can be used for inductive node embeddings.

2.1.3. GraphSAGE

While GCN is only applied in transductive setting with fixed graphs, GraphSAGE provides an inductive unsupervised learning method focusing on inductive node embedding problem [4]. To generate node embeddings on unseen nodes, GraphSAGE tries to randomly sample nodes' neighbourhood and learn an aggregation function, that can be used to represent local relationships between features and embeddings and then calculate for newly observed subgraph. Calculation formula of embeddings is provided below:

$$h_v^{[l]} = \sigma(Wh_v^{[l-1]}, W \cdot Aggregate(\{h_u^{[l-1]} \mid h_u^{[l-1]} \in Sample(Neighbour(h_v^{[l-1]}))\})) \tag{4}$$

In which, h is the feature vector for each layer, W is the weighted matrix, $Aggregate$ is a pre-trained function to calculate embeddings for new nodes, $Neighbour$ represents the set of nodes connected, and $Sample$ represents the sampling function. Note that calculation process relies on the sampling of neighbourhood and aggregating these features. Despite of random sampling, GraphSAGE can learn local graph structure, including mapping nodes to indicators and identifying neighbourhoods. However, some information is possibly lost during sampling process, causing theoretical limitation for GraphSAGE.

2.2. Advanced GNN Models

2.2.1. GIN

One of the main contributions of GIN [6] is that it provides a more robust aggregation function, making GNN as expressive as Weisfeiler-Lehman graph isomorphism test, a powerful test known for distinguishing graphs. It also explains the limitation of GCN and GraphSAGE and their failures on some simple graphs. The updating rules of GIN is listed below:

$$h_v^{[k]} = MLP\left((1+\epsilon)h_v^{[k-1]} + \sum_{h_u^{[k-1]}\ is\ neighbour\ of\ h_v^{[k-1]}} h_u^{[k-1]}\right) \tag{5}$$

In which, h is the feature vector for each layer, ϵ is a weight parameter, and MLP is called Multi-Layer Perceptron, a non-linear function used to fit complex relationships and make the model more expressive. As parameter of MLP, the sum of features is better than mean or maximum value because it keeps more information to be more expressive. A learnable parameter is used to adjust proportion between a node and its neighbourhood. GIN is equivalently expressive with Weisfeiler-Lehman graph isomorphism test, making it a powerful model.

2.2.2. Graph Transformer

Typical GNN relies on local information, making it difficult to build relationships with nodes in further distance. Deeper GNN will also cause over-smoothing problem. Therefore, combination of GNN with Transformer becomes an optimal solution to incorporate global information.

Graph Transformer [7] calculates attentional coefficients considering positional information in the whole graph, including shortest path distance, centrality encoding and Laplacian eigenvectors. Structural-aware attentional coefficients can be calculated as follows:

$$e_{ij} = softmax\left(\frac{Q_i K_j^T}{\sqrt{d}} + b_{ij}\right) \tag{6}$$

In which Q and K are learnable linear combination of input features, representing correlation between input features, and b contains global graph information. By capturing global graph structure, Graph Transformer shows its advantage in tasks like molecule classification and knowledge graph modelling.

2.2.3. GraphCL

Most traditional GNN methods are based on supervised learning and labelled data. However, sometimes it is difficult to acquire labels for graph, such as large-scale social network graph. GraphCL [9] finds a way to learn the representation of nodes by constructing contrastive tasks, making it possible for GNN on large-scale graph.

First GraphCL generates graphs on two views by graph augmentation, in ways of node dropping or feature masking. Then GraphCL encodes these views graph to vectors and compare their contrastive loss. The basic logic is that same graph on different views should have similar embeddings. Therefore, the loss function used in GraphCL is listed below:

$$L_i = -log \frac{\exp\left(\frac{sim(z_i, z_i^+)}{\tau}\right)}{\exp\left(\frac{sim(z_i, z_i^+)}{\tau}\right) + \sum_{j \neq i} \exp\left(\frac{sim(z_i, z_j)}{\tau}\right)} \quad (7)$$

In which positive pair means input features should share high similarity. By minimizing contrastive loss, GraphCL performs self-supervised learning to get representation of graphs and nodes, so it can be applied to a wide range of tasks, such as node classification and link prediction.

3. Applications and Discussions

3.1. Applications

3.1.1. Social Recommender System

In social network, users tend to make similar decisions with those who share same attributes with them, according to the effect of social homophily [15]. This correlation between users naturally forms a social graph that can be well analysed by GNN and be widely applied to social recommender systems. Wenqi et al. proposed GraphRec [16], a graph neural network framework, effectively capturing opinions from users in the user-item graph for social recommendations. Qitian et al. used dual graph attention networks to deal with dynamic social effects from friend users [17], as well as a new strategy to weigh social effects. These models improve the effectiveness and accuracy of social recommender systems.

3.1.2. Drug Discovery

Discovery of new drugs demands huge cost in filtering from large amount of candidate molecules. Since molecules can be represented graphs, applications of graph neural networks in drugs have become hot spots. Pietro et al. proposed MG^2N^2, a sequential molecular graph generator model, which could generalize molecular patterns from training sets [18]. With the help of GNN, MGN can utilize more information from molecular graphs and outperforms competitive baselines. John et al. provided AlphaFold, an accurate computational method to predict protein structures. Evoformer, as the key component in AlphaFold, encodes information of the pairwise relations between residues, passing messages among 3-D molecular graphs. These results show great potentials of graph neural networks.

3.1.3. Traffic Prediction

Road networks in cities are naturally graphs, regarding stops and crossing as nodes and road connections as edges. To apply GNN in traffic predictions, what should be noticed is that many attributes of graphs rely on time and space. In the aspect of space, neighbouring roads always share similar traffic states. Dynamically, traffic states can be regarded as a sequential model. Therefore, combination of GNN and sequential models can help predict traffic states in both aspects. Xiaoyang et al. proposed a spatial temporal graph neural network for traffic flow prediction [19], combining GNN layer, Recurrent network layer and Transformer layer. Fuxian et al. proposed Dynamic Graph Convolutional Recurrent Network (DGCRN) [20], providing better models by generating dynamic graph in each time step. Features of road networks make traffic prediction problem a crossing field for both GNN and sequential models.

3.2. Discussion

3.2.1. Challenges

Despite of various models and applications of GNN, some challenges still need to be deeply analysed and solved. The first challenge is over-smoothing problem [21]. The core principle of GNN is message passing, that nodes in graphs could pass their information to neighbours and then aggregate them for predictions. However, repeated message passing will cause individual nodes to make decisions based on messages of similar node sets. Therefore, those nodes tend to make similar decisions and result smooth prediction curves.

Another drawback of message passing is over squashing problem [22]. On the opposite of over smoothing problem, messages passed from nodes in a long distance are difficult to make contributions. During the propagating process, size of neighbourhood increases exponentially, while node embeddings try to compress information in low-dimensional vectors, resulting the bottleneck of taking advantages of long-range dependencies.

Large-scale graphs are always difficult to be handled by GNN in many aspects. Large data sets of nodes and edges will increase computational and memory cost, making message passing impossible to be implemented on the entire large graphs. Potential solutions including dividing graphs into subgraphs or sampling graphs, risking in high bias and variance. Besides, large-scale graphs are usually combined with dynamic changes in structures, while current research on GNN focus on static graphs processing.

3.2.2. Futural Prospects

As more applications of GNN appear in social recommender systems, drug discovery and physical computations, models and frameworks based on GNN are promised to be more expressive and more extendable. First, mathematical analysis of GNN bottlenecks, including over smoothing and over squashing problems, will systematically optimize these problems and improve the principle of message passing. Second, development of distributed systems will reduce the computational cost of GNN, enhancing GNN's performance on large-scale graphs. Suitable hardwares like GPU designed for GNN computation will also increase the computational speed. Third, benefited from the bloom of Large Language Model, GNN's combination with multi-agent models and self-supervised learning will lead GNN to a more stable and well-performed stage.

4. Conclusion

In conclusion, this paper explains three classical GNN models: GCN, GAT and GraphSAGE, by introducing models' core principles, advantages and applications. Then, this paper introduces three advanced GNN models: GIN, Graph Transformer and GraphCL, with models' propagation rules and improvements over previous solutions. Moreover, various application fields of GNN are discussed, including social recommender systems, drug discovery and traffic prediction. Finally challenges faced by GNN are introduced and analysed. Over-smoothing is the main problem requiring further optimizations. Over-squashing problem should also be noticed. Large-scale graphs still demand better GNN frameworks. In the future, GNN is expected to have more support from systems architecture, more strategies to improve performance more applications in various fields.

References

[1] Wu Z., Pan S., Chen F., Long G., Zhang C., Yu P. S. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2021, 32 (1): 4-24.

[2] Ma L., et al. Acceleration algorithms in GNNs: A survey. IEEE Transactions on Knowledge and Data Engineering, 2025, 37 (6): 3173-3192.

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

[4] Hamilton W., Ying Z., Leskovec J. Inductive representation learning on large graphs. Advances in Neural Information Processing Systems, 2017, 30: 1024-1034.

[5] Veličković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.

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

[7] Li Y., Li Z., Wang P., Li J., Sun X., Cheng H., Yu J. X. A survey of graph meets large language model: Progress and future directions. arXiv preprint arXiv:2311.12399, 2023.

[8] Topping J., Di Giovanni F., Chamberlain B. P., Dong X., Bronstein M. M. Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522, 2021.

[9] You Y., Chen T., Sui Y., Chen T., Wang Z., Shen Y. Graph contrastive learning with augmentations. Advances in Neural Information Processing Systems, 2020, 33: 5812-5823.

[10] Hou Z., Liu X., Cen Y., Dong Y., Yang H., Wang C., Tang J. GraphMAE: Self-supervised masked graph autoencoders. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022: 594-604.

[11] Hou Z., He Y., Cen Y., Liu X., Dong Y., Kharlamov E., Tang J. GraphMAE2: A decoding-enhanced masked self-supervised graph learner. Proceedings of the ACM Web Conference, 2023: 737-746.

[12] Pfaff T., Fortunato M., Sanchez-Gonzalez A., Battaglia P. Learning mesh-based simulation with graph networks. International Conference on Learning Representations, 2020.

[13] Jumper J., Evans R., Pritzel A., Green T., Figurnov M., Ronneberger O., et al. Highly accurate protein structure prediction with AlphaFold. Nature, 2021, 596 (7873): 583-589.

[14] Brandstetter J., Worrall D., Welling M. Message passing neural PDE solvers. arXiv preprint arXiv:2202.03376, 2022.

[15] Sharma K., Lee Y. C., Nambi S., Salian A., Shah S., Kim S. W., Kumar S. A survey of graph neural networks for social recommender systems. ACM Computing Surveys, 2024, 56 (10): 1-34.

[16] Fan W., Ma Y., Li Q., He Y., Zhao E., Tang J., Yin D. Graph neural networks for social recommendation. Proceedings of the World Wide Web Conference, 2019: 417-426.

[17] Wu Q., Zhang H., Gao X., He P., Weng P., Gao H., Chen G. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. Proceedings of the World Wide Web Conference, 2019: 2091-2102.

[18] Bongini P., Bianchini M., Scarselli F. Molecular generative graph neural networks for drug discovery. Neurocomputing, 2021, 450: 242-252.

[19] Wang X., Ma Y., Wang Y., Jin W., Wang X., Tang J., et al. Traffic flow prediction via spatial temporal graph neural network. Proceedings of the Web Conference, 2020: 1082-1092.

[20] Li F., Feng J., Yan H., Jin G., Yang F., Sun F., et al. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. ACM Transactions on Knowledge Discovery from Data, 2023, 17 (1): 1-21.

[21] Rusch T. K., Bronstein M. M., Mishra S. A survey on oversmoothing in graph neural networks. arXiv preprint arXiv:2303.10993, 2023.

[22] Topping J., Di Giovanni F., Chamberlain B. P., Dong X., Bronstein M. M. Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522, 2021.