A Survey of Applications of Graph Neural Networks in Computer Vision

Jian Xu
Nanjing University
Nanjing, China
161200063@smail.nju.edu.cn

Abstract

Despite convolutional neural networks being the most dominate method in the field of computer vision, graph neural networks also have a lot of promising applications. Due to its strong ability to learn from connections, graph neural networks are especially good at high level computer vision tasks. This survey mainly introduces the very basic principle of common graph neural networks and their applications in computer vision.

1. Introduction

In the past decades, deep learning has been applied in many fields, due to the improvement of GPU performances, the development of data sets and the optimization of algorithms. Traditional machine learning tasks that require manual feature extraction (such as pattern recognition, digital image processing, object detection, etc.) have now been replaced by more powerful and efficient end-to-end deep learning models.

With the rapid development of the Internet and mobile devices, an enormous amount of data is generated in our everyday life. These data can be divided into two categories: Euclidean data and non-Euclidean data.

The data we often see in convolutional neural networks (CNNs) is basically Euclidean data. Euclidean data tends to have a regular spatial geometry, which remains unchanged during translation. A picture (image) is a typical Euclidean data, which can be represented as a two-dimensional matrix, suitable for the input of a convolutional neural network.

Conversely, the other type is non-Euclidean data, which is generated from a non-Euclidean space and often does not have a regular structure. Road network maps, paper citations, images, human skeleton actions, molecular structures of compounds , knowledge graphs, social/e-commerce networks, recommend systems, are all non-Euclidean data. The nodes in these graphs do not have an fixed number of neighbors, so the graphs may vary in shape, making them difficult to handle with a simple convolutional neural network. Here comes graph neural networks, to process this kind of data.

In summary, different from CNN, GNN is mainly used in scenarios with complex relationships, such as recommendation systems, social networks, molecular structures, etc. Although CNN is more often used in computer vision, GNN has its own advantages in some specific computer vision tasks. This survey mainly covers the basic principle of GNN and its application in the field of computer vision.

2. How do GNNs work?

2.1. overview

The data in a graph neural network contains the following information:

(1) Node features: Each node has its own features.

(2) Topological features: the geometric topology information in the graph, that is, the connections between nodes.

(3) Edge features: In some graphs, the edges between nodes also have features besides topological relationships, such as different edge weights in weighted graphs or different types of connection in heterogeneous graphs.

In general, non-Euclidean graph data includes not only the information of nodes themselves, but also the topological connections between nodes. The core idea of graph convolution is to use edge information to aggregate node information, generating node's new representations.

GNN can be divided into four main categories, including Recurrent Graph Neural Networks (RecGNN), Convolutional Graph Neural Networks (ConvGNN), Graph Auto Encoders (GAE), and Spatial Temporal Graph Neural Networks (STGNN). Among them, convolutional graph neural network is the most basic one and is closely related to computer vision, so this part briefly introduces the development and principle of convolutional graph neural network.

Considering the similarity between graphs and images, many works try to transfer the traditional convolutional neural network (CNN) to the graph domain, by defining a certain way to communicate between adjacent nodes.

The features are aggregated to complete the corresponding learning tasks.

Convolution operations on graph structures can be divided into two categories: spectral methods and spatial methods. The spectral method realizes the convolution operation in the spectral domain through the chain of "spatial domain-spectral domain-spatial domain"; while the spatial method chooses to define the convolution operation directly between nodes. Based on the theory of spectral graph analysis, the corresponding filters in signal processing are introduced to define graph convolution, so that the convolution operation of graph can be interpreted as a signal denoising process. On the contrary, spatial methods are developed from Recurrent Graph Neural Networks (RecGNNs), which define graph convolutions by the propagation of messages along edges.

2.2. spectral methods

Firstly, in spectral methods, Defferrard approximated CNN with a Laplacian matrix and proposed ChebNet[3]. Then, in 2016, Kipf further proposed graph convolutional neural network (GCN) by approximating ChebNet and achieved good results on semi-supervised graph node classification tasks[8].

GCN bridges the gap between spectral and spatial methods, which makes it one of the most important models. Single layer GCN processes first-order neighbors in graphs. Seemingly, the k layers GCN deals with k-order neighbors. In addition, GCN is able to share convolutional parameters for each node in the graph. To sum up, GCN has a profound impact on the development of GNN.

However, Velivckovic believed that GCN didn't explicitly define convolution, and proposed graph attention network (GAT) in 2017 to learn the weight of each edge in the graph, and then complete the definition of the convolution kernel[15]. The GAT network is one of the spatial methods and will be introduced later.

2.3. spatial methods

The following part covers several spatial methods.

GraphSAGE contains two words, sample and aggregate, which stands for two important ideas: neighbor sampling and neighbor aggregation[6]. Neighbor sampling means that in each round of the aggregation process, the information of neighbors is not aggregated thoroughly, but only a part of them will be sampled, which reduces the computational complexity and avoids over-smoothing caused by each node has the same representation as the central node, after several rounds of convolution. During the neighbor aggregation process, a variety of aggregation functions are provided in GraphSAGE model, including average aggregation, max pooling aggregation, LSTM aggregation, etc.

GAT applies the popular attention mechanism in the convolutional neural network to the calculation of weight between nodes[9]. To be specific, it uses a flexible weight aggregation function instead of a fixed one. GAT also implements a multi-head attention mechanism, using multiple attention mechanism parameters to obtain multiple sets of weighting parameters, and averaging multiple feature vectors to further improve the model expression ability.

Message passing neural network(MPNN), which proposes a general framework for GNN spatial methods[5]. The representation of nodes are obtained through several rounds of message propagation with three functions: message function $M(l)$, reduce function $\sum$ and update function $U(l)$.

$$
\begin{aligned}
m_{u\to v}^{(l)} &= M^{(l)}\left(h_v^{(l-1)}, h_u^{(l-1)}, e_{u\to v}^{(l-1)}\right) \\
m_v^{(l)} &= \textstyle\sum_{u\in\mathcal{N}(v)} m_{u\to v}^{(l)} \\
h_v^{(l)} &= U^{(l)}\left(h_v^{(l-1)}, m_v^{(l)}\right)
\end{aligned}
$$

The team of Pietro Liò proved mathematically that the use of multiple aggregation functions can improve GNN models[2]. Based on that, the author proposed Principal Neighbourhood Aggregation (PNA) network, which combines various aggregators with degree-based scalers. Experiments on many tasks show that the performance of the PNA model is better than traditional GNN models.

Compared with CNN, GNN has the advantage to capture information hidden in relationships between nodes. In fact, relationship modeling is becoming more and more important in many computer vision tasks, because it reflects human recognition process which is accomplished through analyzing the connections with the surrounding environment. Graph networks, which have the ability to structurally represent node relationships, seem to be naturally suitable for such modeling tasks. The next chapter will further introduce the related applications and development of graph neural networks in the field of computer vision.

3. Common tasks

3.1. node classification

With several layers of message aggregation, learnable parameters W (MLP, LSTM, etc.) are added between layers to learn the hidden embedding of nodes. Next, a MLP or a softmax layer is used as the output layer, which enables GNN to complete the node classification task in an end-to-end way.

3.2. Edge classification and link prediction

The message passing part is similar to the node classification task, while the main difference lies in the output

part. Unlike the node classification task, in edge classification and edge prediction tasks, it is necessary to design a function to calculate edge features, which takes two nodes' features as input, and then computes the corresponding edge feature as output. After getting the final representation of the edge, we can calculate the error, and perform back propagation.

3.3. graph classification and graph regression

Similarly, after the message passing aggregation is over, an entire graph feature computation layer needs to be added to the network, which is often called a readout-layer. The final embeddings of all nodes are used as input, and the representation of the entire graph is the output. The most common application is image classification in computer vision, especially medical images.

4. Applications in computer vision

4.1. image classification

Vijay Prakash Dwivedi used GNN to complete the most basic image classification task[4]. Based on MNIST and CIFAR10, this work converts raw images into superpixels through the SLIC algorithm, which is short for simple linear iterative cluster. SLIC converts the image from the RGB color space to the CIE-Lab color space. It combines color and coordinates of each pixel to form a 5-dimensional vector, then the similarity of two pixels can be measured by their vector distance, the larger the distance, the smaller the similarity. Based on that, the superpixel cluster is obtained based on k-means clustering. Next, those superpixels are used as the input of GNN, and the label of image is used as the output. The paper experiments different GNN models on this task. As a result, the accuracy on MNIST is about 97%, and is around 65% on CIFAR10.

Sarah uses GCN to predict diseases, such as Autism Spectrum Disorder and Alzheimer's disease[12]. Unlike traditional CNN simply based on imaging information, GNN allows to incorporate the wealth of imaging and non-imaging information as well as individual subject features simultaneously in disease classification tasks. As a result, their framework can improve over state-of-the-art results on both databases, with 70.4% classification accuracy for ABIDE and 80.0% for ADNI.

Jinkui Hao uses GAT to develop a method that not only overcomes the heavy memory and computational requirements of 3D CNNs, but also leverages the 3D information[7]. To be specific, they frame the distinction of different diseases as a graph classification problem. Each case is represented as a directed graph with a topological structure, where vertices represent the image features of slices, and edges encode the spatial relationship between them. Experiments show that their approach outperforms state-of-the-art methods a lot.

4.2. point cloud processing

The point cloud here is usually a group of 3D points scanned and recorded by LiDAR devices. Point cloud is the 3D data of an object. The data of each point includes 3D coordinates, RGB values and other information, which can be understood as "3D version" of image data. Point cloud can be transformed into an octree or a super point graph, then GCN can be used to explore its topology. Classifying and segmenting point clouds enables LiDAR devices to recognize their surroundings. Given some point clouds, we need to abstract them into graphs according to their topological relationships. This is often used in the field of auto driving, because the signal returned by the car's lidar is a point cloud. Similar to image classification and image segmentation, point cloud also has its own classification and segmentation tasks.

However, the 3D point cloud raw data is not as structured as the image data. The latter is generally arranged in a 2D grid, while the coordinates of the point cloud are continuously distributed. The early methods will first transform the original point cloud into a structured 3D grid, but this has brought problems such as quantization artifacts and huge memory consumption.

MIT's paper Dynamic Graph CNN for Learning on Point Clouds proposed to use a graph neural network to represent the relationship between point and its neighbor points, so as to capture the high-level information of point cloud[16]. It is worth noting that the graph established by the whole model is sparse, and each point is only connected to the nearest K neighbors, in order to reduce the amount of computation. An important innovation of this paper is to make the graph dynamic, that is, after updating the features of each point, the k nearest neighbors of each point will be recalculated to create a new graph. This advantage of the recalculation is that the receptive field of each point is the whole point cloud, but not limited to the local. As a result, the proposed approach achieves state-of-the-art performance on standard benchmarks including ModelNet40 and S3DIS.

Gusi Te also manipulates GNN for point cloud segmentation. However, they choose a spectral GNN to achieve this task[14]. They propose a regularized graph convolutional neural network (RGCNN) that directly consumes point clouds. Leveraging on spectral graph theory, they treat features of points in a point cloud as signals on graph, and define the convolution over graph by Chebyshev polynomial approximation. In the experiment part, they apply their model to point cloud classification and achieve competitive results on ModelNet40 dataset.

4.3. scene graph generation

The task of scene graph generation aims to parse an image into a semantic graph composed of objects and their semantic relationships. Similarly, another application reverses the process by generating realistic images for a given scene graph. Since natural language can be parsed as a semantic graph in which each word represents an object, it is a promising solution to synthesize images in a given text description.

Scene graph is a structured form of an image, in which each node represents an object on the graph, and the edges between nodes represent the relationship between objects.

Danfei Xu applies the basic idea of graph neural network to deal with the problem of scene graph generation[17]. In their framework, graph edges and nodes have the same priority, so that the model can capture relationships more comprehensively. The network learns to iteratively improves its predictions via message passing, during which the model takes advantage of contextual cues to make better predictions on objects and their relationships. The experiments show that their model significantly outperforms previous methods on generating scene graphs .

4.4. object detection

Traditional object detection model only tends to consider the object itself and ignore the relationship between the object and the context. For example, traditional model would mistake a boat on the river for a car, because if only based on the appearance of the object , a boat floating on the river does look like a car.

Obviously, a more intelligent object detection algorithm should also take the surroundings into account. In this way, object detection is no longer just a simple identification problem.

Yong Liu formulates object detection as a problem of graph structure inference, where given an image the objects are treated as nodes in a graph and relationships between the objects are modeled as edges in such graph[11]. Experiments shows that scene context and object relationships truly improve the performance of object detection with more desirable and reasonable outputs.

Zhao-Min Chen proposes a multi-label classification model based on Graph Convolutional Network (GCN)[1]. The model builds a directed graph over image labels, in order to explore the label dependencies to improve the recognition performance. And experiments show that their approach obviously outperforms many CNN methods.

4.5. action recognition

The action recognition task hopes to abstract the human posture into a graph, in which the joints are the nodes, and the skeleton is the edge between the nodes in the graph.

Sijie Yan proposes a novel model of dynamic skeletons called Spatial-Temporal Graph Convolutional Networks (ST-GCN), which automatically learn both the spatial and temporal patterns from data, leading to greater expressive power and stronger generalization capability[18].

Similarly, Maosen Li designs the actional-structural graph convolution network (AS-GCN), which stacks actional-structural graph convolution and temporal convolution as a basic building block, to learn both spatial and temporal features for action recognition[10].

Lei Shi represents the skeleton data as a directed acyclic graph based on the kinematic dependency between the joints and bones in the natural human body[13]. They use a directed GNN to extract the information of nodes(joints, bones) and their relations, so that prediction can be made based on those extracted features. Tests carried out on NTU-RGBD and Skeleton-Kinetics datasets exceeds state-of-the-art performance.

5. Conclusion

In addition to what has been mentioned above, there are many other exciting applications of graph neural networks in computer vision. But in general, graph neural networks are still a new approach for computer vision tasks. Compared with traditional convolutional neural networks, graph neural networks can better capture information in relationships, which makes graph neural networks have the advantages in high-level visual tasks. Finally, we are looking forward to see more and more interesting application of GNN in computer vision, bring out all its high-level potential.

References

[1] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo. Multi-label image recognition with graph convolutional networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5172–5181, 2019.

[2] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Velickovic. Principal neighbourhood aggregation for graph nets. *CoRR*, abs/2004.05718, 2020.

[3] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[4] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *CoRR*, abs/2003.00982, 2020.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017.

[6] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[7] J. Hao, J. Liu, E. Pereira, R. Liu, J. Zhang, Y. Zhang, K. Yan, Y. Gong, J. Zheng, J. Zhang, Y. Liu, and Y. Zhao. Uncertainty-guided graph attention network for

parapneumonic effusion diagnosis. *Medical Image Analysis*, 75:102217, 2022.

[8] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[9] B. Knyazev, G. W. Taylor, and M. R. Amer. Understanding attention in graph neural networks. *CoRR*, abs/1905.02850, 2019.

[10] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. *CoRR*, abs/1904.12659, 2019.

[11] Y. Liu, R. Wang, S. Shan, and X. Chen. Structure inference net: Object detection using scene-level context and instance-level relationships. *CoRR*, abs/1807.00119, 2018.

[12] S. Parisot, S. I. Ktena, E. Ferrante, M. Lee, R. Guerrero, B. Glocker, and D. Rueckert. Disease prediction using graph convolutional networks: Application to autism spectrum disorder and alzheimer's disease. *Medical Image Analysis*, 48:117–130, 2018.

[13] L. Shi, Y. Zhang, J. Cheng, and H. Lu. Skeleton-based action recognition with directed graph neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7904–7913, 2019.

[14] G. Te, W. Hu, Z. Guo, and A. Zheng. RGCNN: regularized graph CNN for point cloud segmentation. *CoRR*, abs/1806.02952, 2018.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. 10 2017.

[16] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[17] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. *CoRR*, abs/1701.02426, 2017.

[18] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *CoRR*, abs/1801.07455, 2018.