Graph Transformer: Learning Better Representations for Graph Neural Network

Boyuan Wang[1], Lixin Cui[1], Lu Bai[1]⋆, Edwin R. Hancock[2]

[1] School of Information, Central University of Finance and Economics, Beijing, China
[2] Department of Computer Science, University of York, York, UK

Abstract. Graph classifications are significant tasks for many real-world applications. Recently, Graph Neural Networks (GNNs) have achieved excellent performance on many graph classification tasks. However, most state-of-the-art GNNs face the challenge of the over-smoothing problem and cannot learn latent relations between distant vertices well. To overcome this problem, we develop a novel Graph Transformer (GT) unit to learn latent relations timely. In addition, we propose a mixed network to combine different methods of graph learning. We elucidate that the proposed GT unit can both learn distant latent connections well and form better representations for graphs. Moreover, the proposed Graph Transformer with Mixed Network (GTMN) can learn both local and global information simultaneously. Experiments on standard graph classification benchmarks demonstrate that our proposed approach performs better when compared with other competing methods.

Keywords: Graph Convolutional Networks · Graph Classification · Graph Transformer

1 Introduction

Graphs are widely used to model complex objects and their dependency relationships in many pattern recognition and machine learning tasks [18]. Along with recent success of deep learning networks, booming interests are focalized on utilizing these methods for analyzing the large-scale and high-dimensional regular or Euclidean data [18]. Particularly, Convolutional Neural Networks (CNNs) have become powerful tools to extract useful statistical patterns from large datasets of image, video, etc. However, because graph structure data is often irregular or non-Euclidean, directly applying CNNs for analyzing such data is difficult. Therefore, great efforts have been devoted to extending CNNs to the graph domain, and a great number of Graph Convolutional Networks (GCNs) [18, 2] have been developed for extracting meaningful features for graph classification.

In general, there are two main categories of GCNs, i.e., the spectral methods and the spatial methods [18]. Specifically, the former defines convolution operation based on the spectral graph theory [4, 5, 11] by calculating the eigenvectors

⋆ Corresponding Author: bailucs@cufe.edu.cn

of Laplacian matrix. However, due to the heavy computational complexity of calculating eigenvectors, these approaches cannot be expanded to big graphs well. Instead, the latter methods are more flexible by defining operations on neighbor vertices[2, 17, 18]. For example, Z.Zhang et al. [18] introduced a Subgraph Convolutional Network (SCN) with quantum walk to facilitate regular convolution operations computing on subgraphs. M.Zhang et al. [17] proposed a novel Deep Graph Convolutional Neural Network (DGCNN) model to consider vertex information both locally and globally SortPooling layer based on the Weisfeiler-Lehman (WL) algorithm. Nevertheless, two major problems arising along with GCNs, i.e., over-smoothing and lack of capturing distant relations. For instance, SCN may lose sights of remote latent connections, DGCNN also cannot learn these potential links well due to the over-smoothing problem. More detailed explanation of these two major issues will be discussed in Sec. 3.1.

To overcome theses issues arising in existing GCNs, we propose a novel Graph Transformer with Mixed Network (GTMN) for graph classification problems in this paper. One important characteristic of the proposed GTMN model is that it can learn latent relations between vertices without using the adjacency matrix. The framework of the proposed GTMN is shown in Fig. 1. In general, the main contributions of this paper are threefold.

First, we illustrate the problem why distant relations cannot be well-learned and propose a novel Graph Transformer unit (GT) to learn latent relations between substructures at arbitrary locations. We elucidate that the proposed GT is able to learn better graph representations with less concerns about over-smoothing than regular GCNs. Thus it can capture meaningful graph information both locally and globally in time. This part will be discussed in Sec. 3.2

Second, we develop a mixed neural network that innovatively combines random walk with graph convolution methods, e.g., SCN with DGCNN. These two approaches are mixed together through the proposed GT to have better views of graph, i.e., local and global perspectives. More generally, we show that any two variants of GCNs can be combined with the help of GT. And the corresponding mixed network takes advantages of each underlying variants. Details will be introduced in Sec. 4

Third, we empirically evaluate the performance of the proposed GTMN on graph classification benchmarks. Experimental results show that when compared to other state-of-the-art methods, our approach is effective.

2 Related Works

In this section, we briefly review some important related works of GTMN. Specifically, we first introduce the Subgraph Convolution Network[18] (SCN). Then we show the operation of the Deep Graph Convolutional Neural Network[17] (DGCNN). Finally, we elucidate the idea of transformer.

Subgraph Convolution Network According to Z.Zhang et al. [18], to operate regular convolution, every vertex is given a QS-Score through quantum walk and

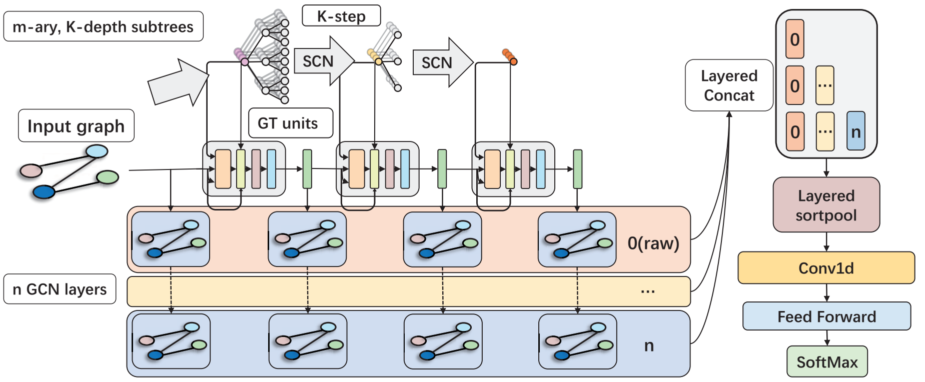

Fig. 1. Overview of the proposed Graph Transformer with Mixed Network (GTMN) architecture. At each step, the proposed Graph Transformer units(GT) takes in the input calculated by the Subgraph Convolution Network (SCN) model and the previous output or the input graph. Each GT calculates the latent relationships between substructures with different size. Up to K (depth of generated subtrees) steps of GT are performed. Outputs then is fed into a set of GCN layers. For n GCN layers, outputs at layer i is concatenated and then $n+1$ big feature matrices are formed. Furthermore, all of them are concatenated and fed into Layered SortPools hierarchically. Final part of the network is to transmute graph features into grid structures and make a prediction for the input graph.

each node forms a m-ary subtrees by grafting and pruning k-hop neighbor edges. Then a subgraph of K-depth is generated from the subtrees for a specific vertex. Thus regular convolution operation is able to process on this subgraph, as shown in Fig. 2.

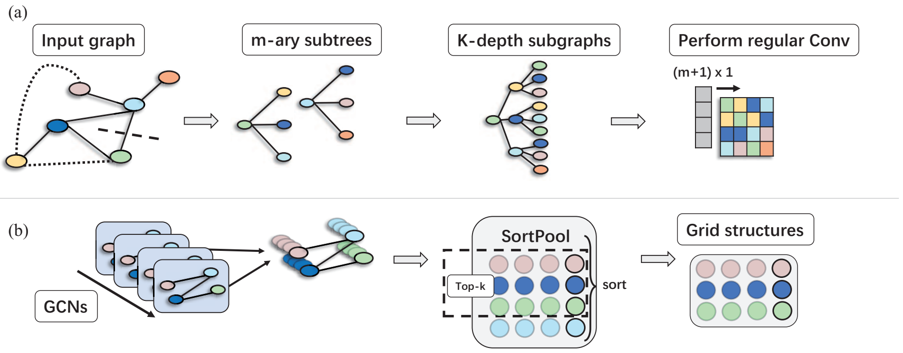

Fig. 2. Detailed procedures of (a) SCN and (b) DGCNN.

Deep Graph Convolutional Neural Network M.Zhang et al. [17] give an spatially-based operation of GCN, i.e.

$$\mathbf{Z} = f(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}), \tag{1}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix of the graph with added self-loops, $\tilde{\mathbf{D}}$ is its diagonal degree matrix, $\tilde{\mathbf{W}} \in \mathbb{R}^{c \times c'}$ denotes a matrix of trainable parameters, f is a nonlinear function, and $\mathbf{Z}$ is convolution's output. For the defined Eq. 1, it can be explained as each node's aggregation with neighbor features.

Moreover, DGCNN introduces a pooling layer called SortPool based on the Weisfeiler-Lehman (WL) algorithm. Vertices are sorted according to last outputs of GCNs, which represent color labels reflecting topological importance. Detailed procedures are shown in Fig. 2

Transformer In natural language processing task, Vaswan et al. [14] proposed an attention-based architecture Transformer to replace traditional Recurrent Neural Networks (RNNs) and improved its performance. The key idea of the Transformer is a Multi-Head attention model that can learn tokens' connections no evaluate how remote they are. Numerous approaches have been developed to apply the transformer to graph domain tasks and have achieved great success[16, 10]. Motivated by the idea of transformer, we propose the Graph Transformer unit of our version in Sec. 3.2.

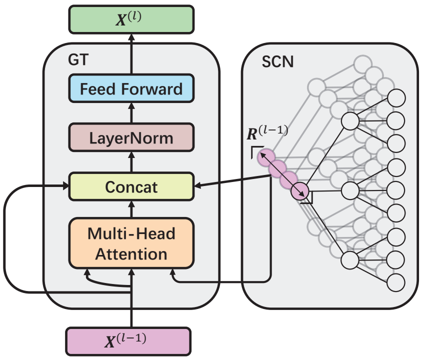

Fig. 3. Detailed structure of Graph Transformer(GT) unit with SCN. Given previous layer's output or the input vertices' features $\mathbf{X}^{(l-1)}$ and preceding roots of all vertices' sub-trees $\mathbf{R}^{(l-1)}$ computed from SCN, multi-head attention can be calculated to learn distant relations beyond original adjacency matrix. Then the output is concatenated with $\mathbf{X}^{(l-1)}$ and $\mathbf{R}^{(l-1)}$ and layer-normalized. Next the normed concatenation is fed into one linear layer to constrain dimensions. After that, the final output of GT $\mathbf{X}^{(l)}$ is formed.

3 The Proposed Graph Transformer Unit

In this section, we first discuss the major existing problems in GCNs. Moreover, we demonstrate the issue of learning latent relations between substructures. Then we introduce our idea of Graph Transformer(GT) unit associated with the random walk method. Finally, we illustrate how the GT can learn latent relations well from the practical and theoretical perspectives.

3.1 Problems of Existing Spatial GCNs

Over Smoothing Many spatial-based GCNs are facing the problem of over-smoothing. For example, consider typical graph convolution operation proposed by DGCNN[17] in Eq. 1. This equation aims to aggregate each vertex's neighbor information and output a new graph representation. However, as the network layer goes deeper, each vertex's features appears to be similar and loses its distinct information. This is called the over-smoothing problem [8]. As analyzed and proved by Liu et al. [8], with more neighbors of larger distances aggregated, each vertex will tend to lose its distinction ultimately.

Latent Relations Between Substructures To overcome the over-smoothing issue mentioned above, many methods are proposed[7, 8]. In particular, it has been shown that methods using random walks[18] are able to overcome this problem through considering sub-graphs or substructures locally. Although more efficient, the proposed methods suffer from the following issue, i.e., two distant substructures' latent relations cannot be learned well, or even can be ignored. Methods mentioned above either learned these links too late or just ignored these distant links. This lead to significant loss of the latent information.

3.2 Graph Transformer Unit

To learn latent relations timely, we propose the Graph Transformer(GT) unit. Indeed, the great success of the transformer[14] has inspired us deeply. One main advantage of GT is that it can take distant tokens into consideration. More specifically, the proposed GT unit is composed of a graph attention layer, a layer norm and a linear layer. Detailed structure of GT can be observed in Fig. 3. We specifically discuss the graph attention layer in this section.

Attention Layer Similar to the attention equation mentioned by Vaswan et al. [14], we propose our graph version:

$$\text{Attention}^{(l)}(\mathbf{R}^{(l-1)}, \mathbf{X}^{(l-1)}) = \text{softmax}(\frac{\mathbf{R}^{(l-1)}\mathbf{W}_{qk}^{(l)}\mathbf{X}^{(l-1)T}}{\sqrt{d}})\mathbf{X}^{(l-1)}\mathbf{W}_{v}^{(l)} \tag{2}$$

where $\mathbf{R}^{(l-1)} \in \mathbb{R}^{N\times d}$ is the root vertices of SCN's last output with shape $N \times d$, $X^{(l-1)} \in \mathbb{R}^{N\times d}$ is the last output of GT unit or the input graph features, d denotes the dimensions of each vertex and $\mathbf{W}_{qk}^{(l)} \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_{v}^{(l)} \in \mathbb{R}^{d\times d_{out}}$ are two learnable matrices. Usually, the attention layer requires three inputs including query, key, and value. Here we define the last output of SCN as query, and the final output of GT unit or input graph features are considered as key and value. Then we develop the graph multi-head layer as below.

$$\text{MultiHead}^{(l)}(\mathbf{R}^{(l-1)}, \mathbf{X}^{(l-1)}) = ||_{i=1}^{h} Attention_i^{(l)}(\mathbf{R}^{(l-1)}, \mathbf{X}^{(l-1)})\mathbf{W}^{O} \tag{3}$$

where $||$ denotes concatenation of all attention heads and $\mathbf{W}^{O} \in \mathbb{R}^{hd\times out}$ is a projection matrix. Then we concatenate the output and the input altogether and the concatenation is fed to the remaining layers.

3.3 Discussions of the proposed GT unit

Practically As mentioned above, distant vertices' latent relations can hardly be learned through the existing methods. However, thanks to the advantages of multi-head attentions and the local information learned by SCN, these latent relations can be well-attained. According to Eq. 2, similarity between the local info and graph representations are calculated, thus similar substructures are guaranteed to be aggregated no matter how distant they are. Hence, **the GT unit with multi layers can capture meaningful potential relations and learn better representations of multiple scales.**

Theoretically We show that the core of GT, i.e. a multi-head layer, is a more generic type of GCNs. Because each head of multi-head layer can learn a unique adjacency matrix respectively, GT can aggregate vertices or substructures of similar representations and learn both the local and global information well. More specifically, for each $head_i$, the calculation of $softmax$ part in Eq. 2 is $\tilde{\mathbf{A}}_{latent} \in \mathbb{R}^{N \times N}$ and it can be considered as a normalized latent adjacency matrix, where $A_{i,j}$ denotes the hidden relation between $vertex_i$ and $vertex_j$. Thus, Eq. 2 can be transformed to the following equation.

$$\text{Attention}^{(l)}(\mathbf{R}^{(l-1)}, \mathbf{X}^{(l-1)}) = \tilde{\mathbf{A}}_{latent}\mathbf{X}^{(l-1)}\mathbf{W}_v^{(l)} \tag{4}$$

which is quite similar to Eq. 1. **Instead of using existing adjacency matrix, GT is able to learn a better graph representation through a learnable relation matrix.** Hence, each head can learn a distinct latent relation and a multi-head layer is able to learn multiple latent relations. With these better learned representations on hand, GCNs can solve the problems mentioned above better.

4 Mixed Graph Network

In this section we propose a novel mixed network structure, i.e., GTMN that combines random walk method with GCNs, e.g. SCN with DGCNN, through the proposed GT units. As shown in Fig. 1, the detailed procedure can be separated into three main sections: 1. feature extraction (GT with SCN), 2. neighbor aggregation(GCNs) , 3. Classification (Layered SortPools with rest parts). First, we elucidate each section concretely. Then we discuss advantages of the proposed mixed graph network.

Feature Extraction we use the proposed GT units with SCN to learn better graph representations, as GT can extract features without using the adjacency matrix. Specifically, for each vertex of a graph, we generate its m-ary K-depth subgraph for SCN[17]. Instead of using quantum walk, we sorted each node by its degree for simplicity. Then we feed both the root of subgraphs and graph vertices into GT for feature extraction. Note that the channels of GT's output is equal to its input. For K-depth subgraphs, we perform K-step GT extraction and have $K + 1$ outputs(including raw input).

Neighbor Aggregation For each output of the first section, we separately feed it to n-layer GCNs. Then we concatenate all outputs of GCN at layer i and denote the concatenation as $\mathbf{X}_i$ (layer 0 contains all raw inputs).The GCN layers help learn structural information that GT cannot well attain, as they concerns about the. After this section, n big feature maps are generated.

Classification For graph classification task, we need to convert the irregular feature maps to grid structures. Inspired by SortPool[17] and the hierarchical approach of Bai et al. [2], we propose a multiple SoortPooling layer called Layered SortPool. In general, this is a more generic version of DGCNN's. Detailed procedures can be seen in Fig. 4. Moreover, we expand each output of Layered SortPool and perform 1d-convolution with both step and kernel size equal to the channels of feature maps. This extracts more features for each vertex and outputs a reduction of a big feature map. Then for all conv1d outputs, we concatenate them altogether and feed the concatenation forward to get the final probabilities for each class.

Discussions of the Proposed Mixed Graph Network Compared with state-of-the-art spatial GCN models, the proposed GTMN has a number of advantages. **First**, because the proposed GT units can learn latent relations between substructures, the proposed GTMN can get better graph representations before feed into GCN layers. This helps avert the problem of over-smoothing as only few convolution operations need to perform for learning neighboring information with the adjacency matrix. As discussed earlier, the GT units tend to learn relations globally, and GCN can extract more local features. **Second**, we combine SCN with DGCNN through the proposed GT units. More generally, each two methods of graph approaches can be mingled by the proposed GT. This helps the network to take advantage of the two underlying methods and relieve their individual drawbacks. Specifically, the method that GT takes in just plays subsidiary role to learn better local information. The other method that GT outputs to is the main model to learn the representations learned by GT. **In conclusion**, GTMN can learn both local and global information simultaneously. With any two methods combined, GTMN is able to well-extract latent relations between substructures and output better graph representations.

5 Experiments

In this section, we evaluate the performance of the proposed GTMN model, and compare it with state-of-the-art methods, i.e., some graph kernels and deep learning method for graph classification on five graph benchmarks[9]. Specifically, these benchmarks are abstracted from the bioinformatics and social networks. Details of these datasets are summarized in Table.1.

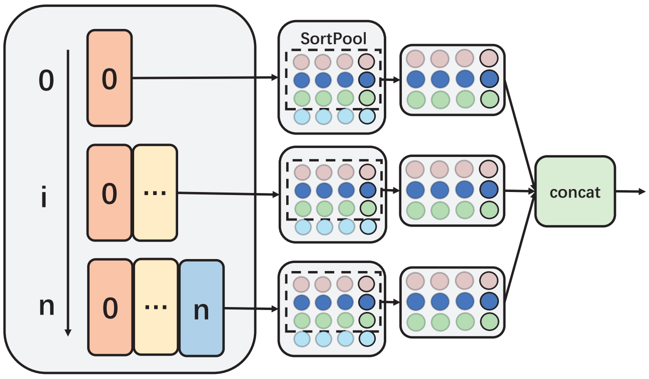

Fig. 4. Detailed procedures for Layered SortPool. Specifically, DGCNN's SortPool uses concatenation of all GCNs' outputs. However, this may lead to some feature loss. Hence we generalize the SortPool by making it layered, to learn features at different levels.

Table 1. Details of the Graph Benchmarks

Datasets	Graphs	Classes	Avg.Nodes	Avg.Edges	Labels	Description
MUTAG	188	2	17.93	19.79	7	Bioinformatics
PTC	344	2	14.29	14.69	19	Bioinformatics
PROTEINS	1113	2	39.06	72.82	3	Bioinformatics
IMDB-B	1000	2	19.77	96.53	-	Social

Experimental Setup We compare the performance of the proposed GTMN model on graph classification tasks with a) four alternative state-of-the-art graph kernels and b) five alternative SOTA deep learning approaches for graphs. Concretely, the graph kernels include 1) the Weisfeiler-Lehman subtree kernel (WL-SK)[12], 2) the shortest path graph kernel (SPGK)[3], 3) the random walk graph kernel (RWGK)[6], and 4) the graphlet count kernel (GK)[13]. The deep learning methods include 1) the deep graph convolutional neural network (DGCNN)[17], 2) the quantum-based subgraph convolutional neural networks (Qs-CNN)[18], 3) the backtrackless aligned-spatial graph convolutional networks (BASGCN), 4) the deep graphlet kernel(DGK)[15], and 5) the diffusion convolutional neural network(DCNN[1].

Table 2. Classification Accuracy (In%± Standard Error) for Comparisons

Datasets	MUTAG	PROTEINS	PTC	IMDB-B
WLSK	82.88±0.57	73.52±0.43	58.26±0.47	71.88±0.77
SPGK	83.38±0.81	75.10±0.50	55.52±0.46	71.26±1.04
RWGK	80.77±0.72	74.20±0.40	55.91±0.37	67.94±0.77
GK	81.66±2.11	71.67±0.55	52.26±1.41	65.87±0.98
DGCNN	85.83±1.66	75.54±0.94	58.59±2.47	70.03±0.86
Qs-CNN	93.13±4.67	78.80±4.63	65.99±4.43	-
BASGCN	90.05±0.82	76.05±0.57	61.51±0.77	74.00±0.87
DGK	82.66±1.45	71.68±0.50	57.32±1.13	66.96±0.56
DCNN	66.98	61.29±1.60	58.09±0.53	49.06±1.37
GTMN	91±3.14	**81.08**±0.363	**68.66**±2.49	69.9±1.15

For the evaluation, we adjust a number of hyperparameters to get the best performance of each dataset, as shown in Table.5 . Note that every GT's head number are set equal to its input channel, and all outputs of the first two sections(SCN, GCN, GT) is equal to convolution. We use tanh function in SCN and GCNs, leakyReLU in each linear layer as well as in conv1d layers. Also, each linear layer is followed by a dropout rate. To optimize the GTMN model, we use the Adam optimizer with the default parameters. For our model, we perform 10-fold cross-validation to compute the classification accuracy, with nine training folds and one validating fold. For each dataset, we repeat the experiment 10 times and report the average classification accuracy and standard errors in Table.2. For the alternative graph kernels and deep learning methods except Qs-CNN, we report the best results collected and experimented by Bai et al. [2]. We report the best results for Qs-CNN from the original paper[18]. Classification accuracy and standard error of each competing approach are also shown in Table.2.

Table 3. Hyperparameters settings for each dataset.

Parameters	K	m	gcn_num	conv	conv1d	fc	fc_num	batch	lr	L2norm	dropout
MUTAG	5	6	6	256	32	32	5	16	0.0001	0	0
PROTEINS	4	9	2	32	32	256	3	64	0.0003	0	0.05
PTC	4	5	3	256	64	64	3	128	0.0001	0	0
IMDB-B	4	9	2	64	256	256	2	256	0.001	0	0.5

Experimental Results and Discussions Table.2 indicates that the proposed GTMN significantly outperforms either the competing graph kernel methods or the deep learning methods for graph classification.

Overall, the reasons for the effectiveness of our method are fourfold. First, the graph kernels with C-SVM classifier are shallow learning methods, while the proposed GTMN can provide an end-to-end deep learning architecture. Thus GTMN can learn better graph characteristics. Second, as elucidated earlier, most deep learning approaches of graph classification can not well-avert problems of over-smoothing and learning distant relations. Instead, the proposed GT units can relieve these problem and learn better graph representations. Third, consider the proposed Qs-CNN and DGCNN, GTMN simplify the quantum walk procedure, generalize the SortPool layer and obtain better performance. This empirically demonstrate the effectiveness of the proposed GTMN.

6 Conclusions

In this paper, we have introduced a novel spatially-based GCN model, i.e., the Graph Transformer with Mixed Network (GTMN), to learn the latent relations between substructures without using the adjacency matrix and alleviate the problem of over-smoothing. Unlike most existing spatially-based GCN models, We propose an attention-based Graph Transformer with a Mixed Network to learn these potential features and learn better graph representations. Experimental results on graph benchmarks indicate the effectiveness of the proposed GTMN.

References

1. Atwood, J., Towsley, D.: Diffusion-convolutional neural networks. In: Advances in neural information processing systems. pp. 1993–2001 (2016)
2. Bai, L., Cui, L., Jiao, Y., Rossi, L., Hancock, E.: Learning backtrackless aligned-spatial graph convolutional networks for graph classification. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
3. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE international conference on data mining (ICDM'05). pp. 8–pp. IEEE (2005)
4. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
5. Henaff, M., Bruna, J., LeCun, Y.: Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163 (2015)
6. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: Proceedings of the 20th international conference on machine learning (ICML-03). pp. 321–328 (2003)
7. Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9267–9276 (2019)
8. Liu, M., Gao, H., Ji, S.: Towards deeper graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 338–348 (2020)
9. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: Tudataset: A collection of benchmark datasets for learning with graphs. In: ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020) (2020), www.graphlearning.io
10. Nguyen, D.Q., Nguyen, T.D., Phung, D.: Universal Self-Attention Network for Graph Classification. arXiv preprint arXiv:1909.11855 (2019)
11. Rippel, O., Snoek, J., Adams, R.P.: Spectral representations for convolutional neural networks. In: Advances in neural information processing systems. pp. 2449–2457 (2015)
12. Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(9) (2011)
13. Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: Artificial Intelligence and Statistics. pp. 488–495 (2009)
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
15. Yanardag, P., Vishwanathan, S.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1365–1374 (2015)
16. Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J.: Graph transformer networks. In: Advances in Neural Information Processing Systems. pp. 11983–11993 (2019)
17. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
18. Zhang, Z., Chen, D., Wang, J., Bai, L., Hancock, E.R.: Quantum-based subgraph convolutional neural networks. Pattern Recognition **88**, 38–49 (2019)