Path Integration Enhanced Graph Attention Network

Hui Wang[1,2], Peng Zhou[1,2], and Junbo Ma[1,2](✉)

[1] Key Lab of Education Blockchain and Intelligent Technology, Ministry of Education, Guangxi Normal University, Guilin 541004, China
nudt_mjb@outlook.com

[2] Guangxi Key Lab of Multi-Source Information Mining and Security, Guangxi Normal University, Guilin 541004, China

Abstract. Graph attention networks are a deep learning method for processing graph data. By learning the relationships between neighbouring nodes in the graph, GATs have been widely used in many fields. However, the graph attention network has the problem of information lag in the process of information aggregation, which degrades the performance of the graph attention network. Referring to the ideas of Feynman path integral theory in physics, we proposed a new graph attention method called PaInGAT to solve the above issue by introducing a new neighbor information aggregation mechanism. Specifically, we improve the neighbour node aggregation mechanism of traditional graph attention networks by calculating the path integral from the source node to the target node to obtain the attention factor, and update the information of multi-order neighbours to the central node directly by the attention factor of the current state at each layer. Through experimental demonstration combining different downstream tasks, our method achieves excellent results on several datasets, demonstrating its effectiveness and advancement.

Keywords: Graph Attention Networks · Graph Data · Information Lag · Aggregation Mechanism

1 Introduction

In real word, data often has quite complex relationships and irregular structures, typically represented as graph structures on non-Euclidean space. The graph data structure can represent the characteristics of nodes and the relationships between nodes. It is often used in a wide range of data representations in various fields, such as index graphs between papers, wiring graphs of the circuit, membership graphs of social networks, molecular graphs of chemical substances, etc. Learning graph data representation has therefore become a popular topic of interest to researchers in recent years [23,45].

Graph neural networks (GNNs) [33] constitute an effective framework for learning graph representations and have been successfully applied to a variety of graph-based tasks. GNNs work by iteratively updating node or subgraph representations through message passing between neighboring nodes. Each node aggregates information from its neighbors and updates its representation based on the received messages. This process is repeated multiple times, allowing nodes to propagate information throughout the graph and refine their representations. Graph Convolutional Networks (GCNs) [28] are the basis for many complex graph neural network models, including autoencoder-based models [29,42], generative models [24] and spatio-temporal networks [43]. It can be divided into two main categories [38], spectral-based and spatial-based. Spectral-based approaches [28] define graph convolution by introducing filters from the perspective of graph signal processing, where the graph convolution operation is interpreted as removing noise from the graph signal. Spatial-based approaches [3,17] represent graph convolution as the aggregation of feature information from neighbourhoods. When algorithms for graph convolution networks run at the node level, graph pooling modules can be interleaved with the graph convolution layer to coarsen the graph into high-level substructures.

One of the most popular variants of GNNs is Graph Attention Networks (GATs) [37], which addresses the shortcomings of GCNs that treat all neighbours equally. Essentially, both GCNs and GATs aggregate features from neighbouring vertices onto the central vertex, but the difference is that GCNs uses a Laplace matrix and GATs uses attention coefficients. Specifically, GATs uses a self-attention mechanism, which means that each node in the graph computes its own attention coefficients based on the similarity between its own features and the features of its neighboring nodes. These attention coefficients are then used to compute a weighted sum of the neighboring node features, which is combined with the original node features to obtain a new representation of the node [37]. Because in real-world scenarios, each of the neighbouring nodes may play a different roles in the influence on the core node, while GCNs simply ignore the correlation of spatial information between nodes and focus only on the topology of the graph when combining the features of neighbouring nodes making the model less generalisable and performance. Overall, GAT has been shown to be effective in a wide range of graph-based learning tasks, such as node classification, graph classification, and link prediction. It has also been extended to handle more complex graph structures, such as heterogeneous graphs and dynamic graphs, and has been combined with other deep learning techniques, such as convolutional neural networks, to achieve state-of-the-art performance on a variety of benchmarks.

However, GATv2 [4] argue that the attention score computed by GAT [37] is only a restricted static attention and does not compute a dynamic attention that can truly express the relevance of nodes, because the attention function computed by GAT is monotonic for any query node with respect to the key, i.e., for different query nodes, the attention score ranking of their neighbouring nodes is fixed [4]. The GATv2 method performance is improved by modifying the internal order of operations to obtain a more expressive approximation of the attention function. Although it computes more expressive attention scores, we found that GAT and GATv2 still suffer from information lag in the computation of attention scores, and that the attention scores used in the computation lagged by $K-1$ layers (K is the path length between two nodes). We will elaborate on this in Sect. 3.

To overcome these drawbacks, we propose a new path integral based graph attention network(PaInGAT). Inspired by ideas from Feynman's path integral theory in physics, we calculate a more effective attention factor by considering the influence of the path length between nodes on the weights when weighting sums in a graph attention networks aggregating information. In continuous space we calculate the transformation of the energy of the path between two points by integration, extending to the discrete space of the graph structure we use the summation operation instead of the integration operation. In addition, by increasing the nonlinear transformation of neighbourhood information, the model can aggregate the neighbourhood information more effectively and improves the expressiveness of the model. At the same time, in traditional GNNs, after propagating multiple base layers, the node information is globally oversmoothed to white noise, resulting in a severe performance degradation. This is because the message passing mechanism of GNNs is based on a plain assumption that neighbouring nodes usually have the same category information. A shallow GNN therefore allows for more cohesive information within categories. Deepening GNNs, on the other hand, means expanding the receptive field of information and inevitably absorbing much inter-category information, which leads to each node tending to be similar. Our proposed method can naturally alleviate this problem, as aggregated higher-order neighbourhood information decays with the edge length of the path, especially when the attention value between some two intermediate nodes on the path drops sharply.

In summary, our main contribution is to propose a new graph attention framework called PaInGAT. Unlike previous graph attention networks, we use node features to compute the path integral between nodes as the attention score to update node representations to obtain new graph representations. Essentially, instead of training a separate feature vector for each node, we train a new set of aggregation functions that aggregate feature information from the nodes' local neighbours of different hop numbers in a base layer. We evaluated our algorithm on four node classification benchmarks and three graph classification benchmarks that tested PaInGAT's ability to generate useful embeddings. Experimental results show that our approach is more effective than previous graph attention models, achieving more expressive graph embeddings.

2 Related Work

Graph neural networks(GNNs) [33] play a very important role for the application of non-Euclidean data in deep learning. Generating node representations that actually rely on graph structure and feature information through graph neural networks is a hot topic of interest for researchers. Various graph neural networks have been proposed in recent years, with both spectral-based approaches [5,11, 21, 31, 42] represented by GCN and spatial-based approaches [1, 3, 8, 10, 18, 19, 26, 30, 46] represented by GAT achieving outstanding graph embedding results in the field of graph representation learning. Among them, GAT uses self-attentive mechanism to achieve modelling of relations for graph data.

Attention mechanisms have been used extensively in the field of deep learning in recent years, whether for computer vision [7, 36], speech processing [9, 44], natural language processing [12, 22] or a variety of other tasks. The operation of introducing attention mechanism in GNNs can be traced back to GAT, which introduced the self-attention mechanism into GNNs replacing the convolution operation in GCNs. Subsequently various graph attention methods have been proposed by researchers. AGNN [35] removes all intermediate fully connected layers of the GCN and calculates the attention factor by cosine similarity. The work on SuperGAT [27] summarises four attention scoring functions, respectively the original GAT function (GO), the node vector dot product function (DP), the scaled dot product function (SD) and the mixed function (MX), and adds a self-supervised task of link prediction to the model to better learn graph embeddings. There are also researchers who have introduced graph attention mechanisms into the field of heterogeneous graphs, such as RGAT [6]. GATv2 [4] computes improved dynamics of attention by modifying the order of operations of the linear mapping and nonlinear transformations of GAT.

3 Preliminaries

In this setion, we first introduce previous work on GAT and GAT2, then explain the existence of information lag in the attentional scores in GATs by analysing the process of aggregation of neighbourhood information.

In the following, we define the problem of interest and the corresponding notations that will be used in this paper. For convenience, we introduce the model on an undirected graph. Like GAT, our method can also be used for digraphs.

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a undirected graph, where node $\mathcal{V} = \{1, 2, ..., n\}$, edge $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. In special, for undirected graphs, we consider each edge as two directed edges with opposite directions. Define the central node i and its neighbour node j, and their corresponding feature vectors are denoted by $\vec{h_i}$ and $\vec{h_j}$ $(\vec{h_i}, \vec{h_j} \subseteq R^F)$ respectively, and (j, i) denotes the edge from node j to node i.

GAT. It adopts a graph attention layer to update the node representation of a graph G by successive applications of the layer. A set $h_i \in R^F(i \in V)$and a corresponding set of edges $\varepsilon(\varepsilon \subseteq \mathcal{E})$ are taken as input to the layer, and the updated node embedding representation $h' \in RF$ is output after one or more layers of superimposed base layers.

In the graph attention layer, each node gives its own query representation for its neighbours, (its own node intermediate representation as query, and its neighbour node intermediate representation as key), i.e. for each vertex i, the similarity coefficients between his neighbours ($j \in N_j \cup i$) and itself is computed by learning the parameters W and the mapping a($\cdot$) as follows:

$$e_{ij} = \text{LeakyReLU}\left(\vec{a}^T \cdot [Wh_i \mid\mid Wh_j]\right), j \in Ni_i \tag{1}$$

where $\cdot^T$ represents matrix transpose and [$\cdot$——$\cdot$] denotes the concatenation operation. Then normalize them across all choices of j by using the softmax function:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp\Big(\text{LeakyReLU}\left(\vec{a}^T \Delta \left[W\vec{h}_i \mid\mid W\vec{h}_j\right]\right)\Big)}{\sum_{k \in N_i} \exp\Big(\text{LeakyReLU}\left(\vec{a}^T \Delta \left[W\vec{h}_i \mid\mid W\vec{h}_k\right]\right)\Big)} \tag{2}$$

Finally, the standardized attention coefficient is combined with the input features of the linear mapping, and then the nonlinear mapping is used to obtain the final graph embedding representation.

$$\vec{h}_i' = \sigma\left(\sum_{j \in \mathcal{N}_j} \alpha_{ij} \Delta W \vec{h}_j\right) \tag{3}$$

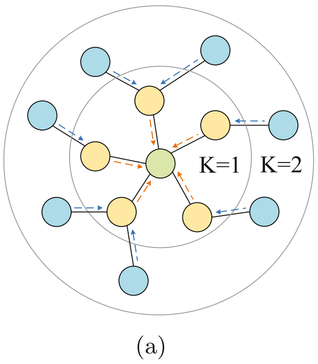
GATv2. The type of attention computed by GAT is restricted because the attentional scores is unconditional on the query node and the attentional function is monotonic with respect to the neighbourhood (key), thus limiting the expressiveness of the model. To address this problem, Shaked Brody et al. propose an improvement to the calculation of attention scores: changing the order of operation of the non-linear transformation and mapping $\vec{a}$ in Eq. 1.

$$e_{ij} = \vec{a}^T LeakyReLU\left(W\Delta\left[h_i || h_j\right]\right), j \in \mathcal{N}_i \tag{4}$$

They believed that using the learning layers W and $\vec{a}$ consecutively could collapse into a single linear layer, affecting the calculation of attention scores. Like the attention mechanism in Transformer, both GAT and GATv2 also consider the multi-heads attention. The output results of multiple attention heads are obtained through the concate operation or their average value is taken to obtain the output of the GAT layer.

$$\vec{h}_i' = \overset{K}{\underset{k=1}{||}} \sigma\left(\sum_{j \in N_j} \alpha_{ij} W^k \vec{h}_j\right) \tag{5}$$

Information Lag in the Aggregation Process. Taking the two-order neighbourhood information aggregation of node i in Fig. 1b as an example, according to the information aggregation formula of GAT and GATv2 we can obtain the representation of nodes i, j after the first base layer aggregation of its own and neighbourhood information as:



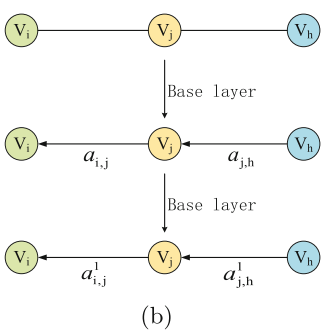


Fig. 1. Aggregation process of GATs

$$x_i^1 = \sigma(a_{i,i}Wx_i + \sum_{j\in N_i} a_{i,j}Wx_j) \tag{6}$$

$$x_j^1 = \sigma(a_{j,j}Wx_j + \sum_{h\in N_j} a_{j,h}Wx_h) \tag{7}$$

where $\sigma(\cdot)$ indicates a layer of nonlinear transformation. After the second base layer, the representation of node i is:

$$\begin{aligned} x_i^2 =& a^1_{i,i}W^1x^1_i + \sum_{j\in N_i} a^1_{i,j}W^1x^1_j \\ =& a^1_{i,i}W^1(a_{i,i}Wx_i + \sum_{j\in N_i} a_{i,j}Wx_j) + \sum_{j\in N_i} a^1_{i,j}Wa^1(a_{j,j}Wx_j + \sum_{h\in N_j} a_{j,h}Wx_h) \\ =& a^1_{i,i}W^1a_{i,i}Wx_i + a^1_{i,i}W^1\sum_{j\in N_i} a_{i,j}Wx_j + \sum_{j\in N_i} a^1_{i,j}W^1a_{j,j}Wx_j \\ &+ \sum_{j\in N_i} a^1_{i,j}W^1\sum_{h\in N_j} a_{j,h}Wx_h \\ =& a^1_{i,i}W^1a_{i,i}Wx_i + (a^1_{i,i}a_{i,j} + a^1_{i,j}a_{j,j})\sum_{j\in N_i} W^1Wx_j \\ &+ \sum_{j\in N_i}\sum_{h\in N_j} a^1_{i,j}a_{j,h}W^1Wx_h \end{aligned} \tag{8}$$

For ease of representation, we omit the operation of the non-linear transformation in Equation(8). We can intuitively know that our conclusions are unaffected due to the monotonically increasing nature of LeakyReLU.

It can be seen from the above calculation process that: The graph attention network uses the attention factors $a^1_{i,j}$ of the current state and $a_{j,h}$ of the previous state to calculate the representation of the node when aggregating information from the second-order neighbour node h to the central node i. This implies that there is a lag in the update of information from node i to node h. When expanding to kth order neighbours, we can obtain that the process lags by k-1st order.(K represents the path length between two nodes). In order to avoid the problem of information lag in the calculation process, in our method, we use the path integral method to update the node representation of the central node with the inter-node attention value of the current state.

4 Proposed Method

In this section, we first describe the structure of the base layer of the PaInGAT network.

Our model will be based on two illuminating assumptions:

1. Information exchange will exist between the nodes on each path in the graph.
2. Information transfer is not only a function of paths, but can be further understood as a function of path length.

We compose paths between nodes by edges on the graph. Generally, for two nodes in a graph, there are multiple paths between them. By the different paths they can be classified as neighbours of different orders. It is even possible to change into more jumping neighbours by using oneself as a folding point. Then we define the length of the path between central node i and neighbouring node j as K, i.e. set the perceptual field size of the model as K, and aggregate the 1st to Kth order neighbourhood information of the central node i through the attention mechanism.

4.1 Attention Module

In the previous graph attention methods, in order to obtain sufficient expressiveness, the input raw features need to be transformed in a learnable way to get intermediate representations. Finally the intermediate representation will be aggregated into the representation of the central node by using the attention factor as a coefficient. However, this transformation uses only a linear transformation of a fully connected layer and not a non-linear transformation, so it yields a limited expressive power. For this reason, we set up a separate feature transformation module W(·), which further improves the expressiveness of the features by boosting the linear transformation of the input features through two fully connected layers and adding an activation function between the two fully connected layers to do non-linear transformation (we use the LeakyReLU function with negative input slope $\alpha = 0.2$ in our experiments).

$$g_i = W(h_i) \in \mathbb{R}^{N \times F'} \tag{9}$$

We then use the transformed input features to calculate the attentional fraction between adjacent nodes. The attention score is:

$$e_{ij} = \vec{a}^T LeakyReLU\left(\left[\, g_i' \,||\, g_j' \right]\right), \quad j \in N_i \tag{10}$$

It represents the importance, i.e. the degree of influence, of node i on adjacent node j. Where $\vec{a} \in \mathbb{R}^{2F'}$ denotes a fully connected layer. Then the attention score is normalised by the softmax function as follows.

$$a_{ij} = \frac{exp\left(e_{ij}\right)}{\sum_{k \in N_k \cup i} e_{ik}} \tag{11}$$

4.2 Information Aggregation Module

Finally, we weight the aggregated neighbourhood information by the computed attention score. Feynman path integral theory tells us that the probability amplitude of a particle moving from A to B in continuous space is the integral of all possible paths, which is defined as summation in discrete space. Inspired by Feynman's path integral theory and PanConv [32], we extend the rule of motion of particles in space to the propagation of information in graph networks.

Similarly, then the propagation coefficient of the message from node i to its kth order neighbour node j is the sum of the coefficients on each path between them. Intuitively, we represent the attentional score $p(i, j; k)$ of a single path passing through multiple nodes as the product of the attentional scores of each edge on this path. For example, assuming that a particular path of length 2 from i to j goes through node h, then $p\left(i, j; 2\right) = a_{ih}a_{hj}$ for that path. By this method we can work out the attention fraction of any path between two nodes. Finally the information passed on all paths (including the node's own information) is summed to obtain the final node representation. By correcting the path length K, we can control the perceptual field of the model.

$$\vec{h}_i' = \sum_{k=1}^{K} \sum_{j \in N_{k(i) \cup i}} \sum p(i, j; k) g_i' \tag{12}$$

where $\sum p(i, j; k)$ denotes the sum of the attention scores of all paths of length k between nodes i, j. In graph data, a node can often be considered as a neighbour of a different order to the central node by following different paths, or even as a neighbour of a higher order by repeatedly passing an intermediate node in the path. Also inspired by the theoretical minimum action principle in physics, we determine the order between nodes in terms of their shortest path lengths in the real calculation process.

5 Experiments

In this section, we show test results of PaInGAT on graph datasets for node classification and graph classification tasks, and demonstrate the advanced performance of our approach by rigorously comparing it with previous graph attentive methods and other no-attention methods. All experiments were done with code written in pyTorch Geometric. Table 1 shows our dataset specifications for the node classification task, and Table 2 shows our dataset specifications for the graph classification task.

Table 1. Specification of the dataset used in the node classification task

	Cora	Citeseer	Pubmed	ogbn-arxiv
# Nodes	2708	3327	19717	169343
# Edges	5429	4732	44338	2315598
# Features	1433	3703	500	128
# Classes	7	6	3	40

Table 2. Specification of the dataset used in the grpah classification task

	PROTEINS	MUTAG	PTC
#Graphs	1113	188	344
#Nodes	∼39.1	∼17.9	∼25.6
#Edges	∼39.1	∼39.6	∼2.0
#Features	3	7	19
#Classes	2	2	2

5.1 Node Classification

We chose four commonly used graph datasets, Cora, Citeseer, Pumbed [34], and ogbn-arxiv. We tested the node classification task for graph data on the these datasets, comparing PaInGAT with GCN [28], GraphSAGE [19], Diff-Net [39], TAGCN [14], Graph-Bert [40], AGNN [30], SuperGAT [27], GAT [37], and GATv2 [4], which were previously graph neural network models. To be fair, we used the same datasets divisions as in the GAT experiments. Also, to prevent interference from other conditions, the same content was used for all parts of our code except for the model.

Table 3 shows our final experimental results. It can be seen that our model achieves the best experimental results compared with other models. In addition, we tested the effect of different attention heads for model performance on the ogbn arxiv dataset. The experimental results are shown in Table 4. It can be seen that the number of attention heads has an effect on the performance of the model.

5.2 Graph Classification

To further evaluate the effectiveness of our model, we implemented experiments on several real-world graph classification problems. PROTEINS [13] dataset is a collection of protein molecules that are classified as enzymes or non-enzymes. MUTAG [25] and PTC [20] datasets are composed by small molecule compounds. In the former dataset, the task is to identify mutagenic molecular compounds for potentially commercial drugs, while in the latter the goal is to identify chemical compounds based on their carcinogenicity in rodents. Three different sizes of graph classification datasets were chosen to validate the performance of

Table 3. Results of node classification tests on the Cora, Citeseer, Pumbed

Method	Cora	Citeseer	Pubmed
GCN	81.5% ± 0.7%	70.3% ± 1.0%	79.0% ± 0.7%
GraphSAGE	82.1% ± 0.6%	71.9% ± 0.9%	78.0% ± 0.7%
Diff-Net	85.1% ± 0.4%	72.7% ± 0.6%	78.3% ± 0.6%
TAGCN	83.3% ± 0.7%	71.4% ± 0.5%	81.1% ± 0.5%
GRAPH-BERT	84.1% ± 0.9%	71.0% ± 0.5%	79.5% ± 0.4%
AGNN	81.0% ± 0.3%	69.8% ± 0.4%	78.0% ± 0.5%
SuperGAT	84.3% ± 0.6%	75.6% ± 0.8%	81.7% ± 0.5%
GAT	83.0% ± 0.4%	72.5% ± 0.5%	79.0% ± 0.4%
GATv2	83.5% ± 0.3%	72.6% ± 0.3%	79.3% ± 0.6%
PaInGAT(ours)	**85.4% ± 0.7%**	**74.1% ± 0.3%**	**81.8% ± 0.6%**

Table 4. The effect of different heads on classification accuracy

Method	head number	ogbn-arxiv
GAT	1	70.71 ± 0.19
	8	70.73 ± 0.18
GATv2	1	71.01 ± 0.14
	8	70.91 ± 0.20
PaInGAT(ours)	1	72.13 ± 0.41
	8	**72.33 ± 0.15**

our model. We chosed no-attention method SPI-GCN [2], GCN [28], DGCNN [41], GIN [15], PANConv [32] and attentional methods hGANet [16], GAT [37], GATv2 [4] to compare with our model, the experiments results show that our method outperforms other graph attention methods under the same experimental conditions (Table 5).

Table 5. The Result of graph classification on Dataset

Method	PROTEINS	MUTAG	PTC
SPI-GCN	74.05%	85.30%	57.72%
GCN	76.23%	84.33%	63.45%
DGCNN	76.26%	85.83%	58.59%
GIN	72.32%	89.47%	64.86%
PANConv	74.60%	88.54%	65.92%
hGANet	77.64%	88.96%	64.76%
GAT	74.54%	89.63%	66.40%
GATv2	76.83%	89.94%	67.22%
PaInGAT(ours)	**78.76%**	**91.26%**	**68.58%**

6 Conclusion

In this paper, we analyse previous work on GATs and find that they fail to compute better attention scores and do not efficiently aggregate information about the higher-order neighbours of nodes. In order to solve these problems, we add a non-linearly transformed node embedding module to the process of computing attention scores, and use the attention product on the path to compute attention scores among higher-order neighbours, allowing PaInGAT to implement the approximator attention function.

It has been demonstrated experimentally that our model achieves good performance on various datasets. However, we have to admit that PaInGAT has a higher computational complexity compared to other models such as GAT, which will be the next direction of our research.

Acknowledgment. This work is supported in part by the Project of Guangxi Science and Technology with Grant Number GuiKeAB23026040, the Research Fund of Guangxi Key Lab of Multi-source Information Mining & Security with Grant Number MIMS20-04 and the Research Fund of Guangxi Key Lab of Multi-source Information Mining & Security with Grant Number 20-A-01-02.

References

1. Abu-El-Haija, S., et al.: Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing. In: International Conference on Machine Learning, pp. 21–29. PMLR (2019)
2. Atamna, A., Sokolovska, N., Crivello, J.C.: SPI-GCN: a simple permutation-invariant graph convolutional network (2019)
3. Atwood, J., Towsley, D.: Diffusion-convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
4. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? arXiv preprint arXiv:2105.14491 (2021)
5. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
6. Busbridge, D., Sherburn, D., Cavallo, P., Hammerla, N.Y.: Relational graph attention networks. arXiv preprint arXiv:1904.05811 (2019)
7. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 213–229. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_13
8. Chen, J., Ma, T., Xiao, C.: FastGCN: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247 (2018)
9. Chorowski, J., Bahdanau, D., Cho, K., Bengio, Y.: End-to-end continuous speech recognition using attention-based recurrent NN: First results. arXiv preprint arXiv:1412.1602 (2014)
10. Dai, H., Kozareva, Z., Dai, B., Smola, A., Song, L.: Learning steady-states of iterative algorithms over graphs. In: International Conference on Machine Learning, pp. 1106–1114. PMLR (2018)

11. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
12. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
13. Dobson, P.D., Doig, A.J.: Distinguishing enzyme structures from non-enzymes without alignments. J. Mol. Biol. **330**(4), 771–783 (2003)
14. Du, J., Zhang, S., Wu, G., Moura, J.M., Kar, S.: Topology adaptive graph convolutional networks. arXiv preprint arXiv:1710.10370 (2017)
15. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
16. Gao, H., Ji, S.: Graph representation learning via hard and channel-wise attention networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 741–749 (2019)
17. Gasteiger, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
18. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR (2017)
19. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
20. Helma, C., King, R.D., Kramer, S., Srinivasan, A.: The predictive toxicology challenge 2000–2001. Bioinformatics **17**(1), 107–108 (2001)
21. Henaff, M., Bruna, J., LeCun, Y.: Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163 (2015)
22. Hu, D.: An introductory survey on attention mechanisms in NLP problems. In: Bi, Y., Bhatia, R., Kapoor, S. (eds.) IntelliSys 2019. AISC, vol. 1038, pp. 432–448. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29513-4_31
23. Hu, R., et al.: Graph self-representation method for unsupervised feature selection. Neurocomputing **220**, 130–137 (2017)
24. Hu, Z., Dong, Y., Wang, K., Chang, K.W., Sun, Y.: GPT-GNN: generative pre-training of graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1857–1867 (2020)
25. Kazius, J., McGuire, R., Bursi, R.: Derivation and validation of toxicophores for mutagenicity prediction. J. Med. Chem. **48**(1), 312–320 (2005)
26. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P.: Molecular graph convolutions: moving beyond fingerprints. J. Comput. Aided Mol. Des. **30**(8), 595–608 (2016)
27. Kim, D., Oh, A.: How to find your friendly neighborhood: graph attention design with self-supervision. arXiv preprint arXiv:2204.04879 (2022)
28. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
29. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
30. Li, R., Wang, S., Zhu, F., Huang, J.: Adaptive graph convolutional neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
31. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting. arXiv preprint arXiv:1707.01926 (2017)

32. Ma, Z., Xuan, J., Wang, Y.G., Li, M., Liò, P.: Path integral based convolution and pooling for graph neural networks. Adv. Neural. Inf. Process. Syst. **33**, 16421–16433 (2020)
33. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Networks **20**(1), 61–80 (2008)
34. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–93 (2008)
35. Thekumparampil, K.K., Wang, C., Oh, S., Li, L.J.: Attention-based graph neural network for semi-supervised learning. arXiv preprint arXiv:1803.03735 (2018)
36. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
37. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
38. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst. **32**(1), 4–24 (2020)
39. Zhang, J.: Get rid of suspended animation problem: deep diffusive neural network on graph semi-supervised classification. arXiv preprint arXiv:2001.07922 (2020)
40. Zhang, J., Zhang, H., Xia, C., Sun, L.: Graph-Bert: only attention is needed for learning graph representations. arXiv preprint arXiv:2001.05140 (2020)
41. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. vol. 32 (2018)
42. Zhang, X., Liu, H., Li, Q., Wu, X.M.: Attributed graph clustering via adaptive graph convolution. arXiv preprint arXiv:1906.01210 (2019)
43. Zheng, W., Zhu, X., Zhu, Y., Hu, R., Lei, C.: Dynamic graph learning for spectral feature selection. Multimedia Tools Appl. **77**(22), 29739–29755 (2018)
44. Zhou, P., Yang, W., Chen, W., Wang, Y., Jia, J.: Modality attention for end-to-end audio-visual speech recognition. In: ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6565–6569. IEEE (2019)
45. Zhu, X., Zhu, Y., Zhang, S., Hu, R., He, W.: Adaptive hypergraph learning for unsupervised feature selection. In: IJCAI, pp. 3581–3587 (2017)
46. Zhuang, C., Ma, Q.: Dual graph convolutional networks for graph-based semi-supervised classification. In: Proceedings of the 2018 World Wide Web Conference, pp. 499–508 (2018)