Article

Not peer-reviewed version

(GAAN)Graph Adaptive Attention Network with Cross Entropy

Zhao Chen *

Posted Date: 14 June 2024

doi: 10.20944/preprints202406.0939.v1

Keywords: Non-Euclidean; GCN; Adaptive Attention Mechanism; Multi-Head Graph Convolution; Cross Entropy

 doi:10.20944/preprints202406.0939.v1

Article

(GAAN)Graph Adaptive Attention Network with Cross Entropy

Zhao Chen

School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, Hubei, China; oliver@whu.edu.cn

Abstract:Non-Euclidean data, such as social networks, and citation relationships between documents, has node information and structural information. Graph Convolutional Network(GCN) can automatically learn node features and association information between nodes. The core ideology of the graph convolutional network is to aggregate node information by using edge information, thereby generating a new node feature. In the process of updating node features, there are two core influencing factors. One is the number of neighboring nodes of the central node, the other is the contribution of the neighboring nodes to the central node. Due to the previous GCN methods not simultaneously considering the numbers and different contributions of neighboring nodes to the central node, we design the Adaptive Attention Mechanism(AAM). To further enhance the representational capability of the model, we utilize Multi-Head Graph Convolution(MHGC). Finally, we adopt cross-entropy(CE) loss function to describe the difference between the predicted results of node categories and the ground truth (GT). Combined with backpropagation, this ultimately achieves accurate node classification. Based on AAM, MHGC and CE, we contrive the novel Graph Adaptive Attention Network (GAAN). Experiments show that the classification accuracy has achieved outstanding performances on Cora, Citeseer and Pubmed datasets.

Keywords: Non-Euclidean; GCN; Adaptive Attention Mechanism; Multi-Head Graph Convolution; Cross Entropy

1.Introduction

Many data in real life have irregular spatial structures, known as non-Euclidean data, such as social networks, recommendation systems, citation relationships between documents, transportation planning, natural language processing, etc. This type of data has both node information and structural information, which traditional deep learning networks like CNN, RNN, Transformer, etc cannot well represent. Graph Convolutional Network (GCN) [1], shown in Figure 1, is a class of deep learning models used for processing graph data, and they have made significant progress in graph data in recent years. In the real world, many complex systems can be modeled as graph structures, such as social networks, recommendation systems, automatic modulation classification (AMC) of underwater acoustic communication signals[2], etc. The nodes and edges of these graph data represent entities and their relationships, which is of great significance for understanding information transmission, node classification, graph classification, and other tasks in graph structures. However, compared with traditional regularized data such as images and text, graph data processing is more complex. Traditional Convolutional Neural Networks(CNN)[3-4] and Recurrent Neural Networks(RNN)[5-6] cannot be directly applied to graph structures because the number of nodes and connections in the graph may be dynamic. Therefore, researchers have begun exploring new graph neural network models to effectively process graph data. Graph Convolutional Networks (GCN) were proposed in this context, making an important breakthrough in graph data. The main idea of GCN is to use the neighbor information of nodes to update their representations, similar to traditional convolution operations, but on graph structures. By weighted averaging of neighboring nodes, GCN achieves information transmission and node feature updates, allowing the model to

 doi:10.20944/preprints202406.0939.v1

better capture the local and global structures in the graph. More broadly, Graph Convolutional Networks (GCN) are a special case of Graph Neural Networks (GNN).

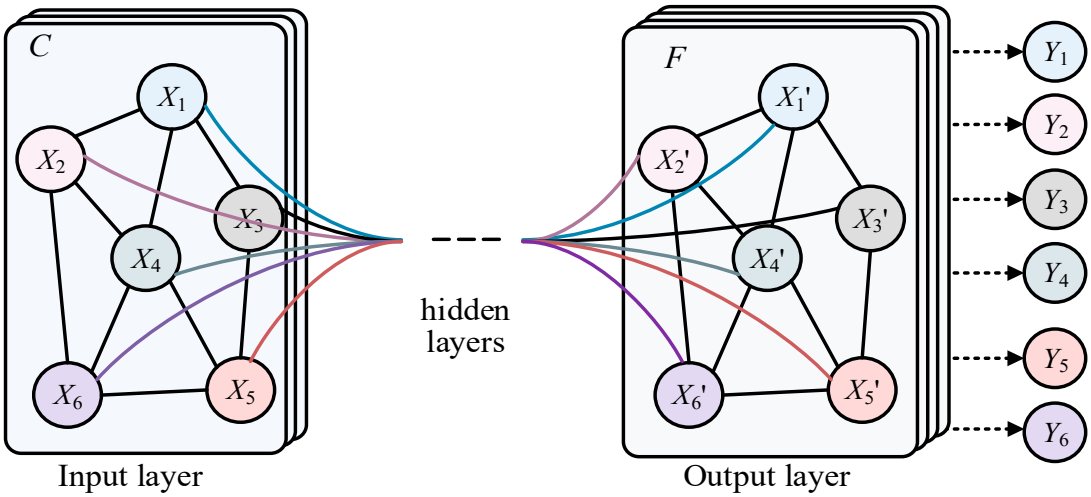

Figure 1. The structure of GCN.

The previous graph convolution methods have not fully considered the number and importance of neighboring nodes. To solve the above problems, we propose the novel GAAN (Graph Adaptive Attention Network), and our main contributions are in the following two areas:

1. To generate different weights for each neighbor node of the central node, we design the novel adaptive attention mechanism(AAM).
2. Based on AAM, we utilize Multi-head Graph Convolution(MHGC) to model and represent features better.
3. We adopt cross-entropy loss function to model the bias between the predicted values and the ground truth, which greatly improves the classification accuracy.

2. Related Work

Graph Convolutional Networks (GCNs) have emerged as a powerful tool for deep learning on graph-structured data, demonstrating impressive performance across various domains such as social network analysis, bioinformatics, and recommendation systems. GCN methods can be broadly categorized into two types based on their convolution approach: spectral-based methods and spatial-based methods. In this paper we synthesizes key literature on these two approaches, discussing their principles, advantages/disadvantages, and applications.

2.1 Spectral-based Methods

Spectral-based methods are rooted in spectral graph theory and graph signal processing, leveraging the Laplacian spectrum of graphs for convolution operations. The foundation for this approach can be traced back to Bruna et al.[7], who introduced Spectral Networks. They utilized the Fourier transform to perform convolutions on graphs, marking the initial foray into spectral methods. Defferrard et al.[8] advanced this concept with ChebNet, a method that uses Chebyshev polynomials to approximate the spectral convolution, significantly enhancing computational efficiency. This method addressed the scalability issue of the original spectral networks by localizing the convolution operation. Kipf and Welling[1] made a seminal contribution with their Semi-Supervised Graph Convolutional Networks (GCN), simplifying the spectral convolution process with a first-order approximation. This innovation allowed GCNs to operate efficiently on large-scale graph data and established a benchmark in the field. Their work demonstrated the practical applicability of spectral methods in semi-supervised learning tasks.

 doi:10.20944/preprints202406.0939.v1

Hammond et al.[9] extended the theoretical foundations of spectral methods by exploring wavelet transforms on graphs. This work enriched the theoretical landscape and provided new tools for signal processing on graphs. Henaff et al.[10] further showcased the potential of spectral methods in handling complex graph structures by applying deep convolutional networks to graph data.

Levie et al.[11] introduced CayleyNets, utilizing Cayley polynomials to increase the flexibility and expressive power of spectral convolutions. Their work highlighted the adaptability of spectral methods to various graph structures and provided a robust framework for further developments. Shuman et al.[12] offered a comprehensive overview of signal processing on graphs, systematically explaining the theoretical underpinnings of spectral methods.

Bianchi et al.[13] proposed ARMA-GNN, which employs Autoregressive Moving Average (ARMA) filters to improve the performance of spectral convolutions. This method demonstrated the potential of integrating classical signal processing techniques with deep learning on graphs. Defferrard et al.[14] explored the application of deep networks on toric graphs, illustrating the adaptability of spectral methods to specialized graph structures. Finally, Chung and Hu[15] provided the mathematical foundation for spectral methods with their work on spectral graph theory.

2.2. Spatial-based Methods

Spatial-based methods define convolutions directly on the graph nodes and their neighborhoods, circumventing the computational complexity associated with spectral transformations. Hamilton et al.[16] designed GraphSAGE, a seminal spatial-based method that uses sampling and aggregation of neighbor node features for efficient node representation learning on large-scale graph data. This approach highlighted the practicality of spatial methods in real-world applications where scalability is crucial.

Veličković et al.[17] made significant strides with the Graph Attention Network (GAT), incorporating attention mechanisms to assign different weights to neighbor nodes based on their importance. This innovation enhanced the expressive power of spatial methods, enabling more nuanced and effective learning on graphs.

Monti et al.[18] demonstrated the application of spatial methods to the graph matrix completion problem with their Geometric Matrix Completion method. This work showcased the versatility of spatial methods in addressing various graph-related tasks. Monti et al.[19] further proposed the Mixture Model Network (MoNet),contriving a general framework for defining convolutional operations on graphs using a mixture model paradigm. This method provided a flexible and powerful tool for graph convolution, accommodating a wide range of graph structures.

Wu et al.[20] designed the Simplified Graph Convolutional Network (SGC), a method that reduces the complexity of traditional GCNs by removing the non-linear activation functions between layers. This simplification not only improved computational efficiency but also retained competitive performance in various tasks, emphasizing the potential of streamlined spatial methods.

Xu et al.[21] addressed the challenge of capturing higher-order dependencies in graphs with their Jumping Knowledge Network (JK-Net). By allowing the network to adaptively select and combine different neighborhood ranges, JK-Net enhanced the capability of spatial methods to learn from complex graph structures.

Graph Isomorphism Network (GIN)[22] tackled the expressiveness of spatial methods, ensuring that the network can distinguish different graph structures effectively. This method set a new standard for the expressiveness of spatial-based GCNs by drawing on insights from the Weisfeiler-Lehman graph isomorphism test.

Liao et al.[23] presented LanczosNet, which leverages the Lanczos algorithm to improve the efficiency and effectiveness of spatial convolutions. This method demonstrated the potential of integrating numerical optimization techniques with graph neural networks to achieve superior performance.

The development of the Spatial-Temporal Graph Convolutional Network (ST-GCN)[24] extended spatial methods to dynamic graphs, capturing both spatial and temporal dependencies.

 doi:10.20944/preprints202406.0939.v1

This expansion opened new avenues for applying GCNs to time-evolving graph data, such as in traffic prediction and action recognition.

Finally, Zhang and Chen[25] introduced the Diffusion Convolutional Neural Network (DCNN), which models the diffusion process on graphs to perform convolutions. This approach provided a novel perspective on spatial methods, emphasizing the importance of modeling the underlying processes governing graph data.

Both spectral-based and spatial-based methods have significantly advanced the field of graph convolutional networks, each offering unique advantages and applications. Spectral methods excel in leveraging mathematical foundations from graph theory and signal processing, providing a robust theoretical framework and powerful tools for graph convolution. Spatial methods, on the other hand, offer practical scalability and flexibility, making them suitable for a wide range of real-world applications. This paper aims to solve practical social network issues, so the research is based on spatial methods. To address the differences in the numbers and contributions of neighboring nodes of central nodes, in this paper we designed the novel Graph Adaptive Attention Network (GAAN).

3. Methodology

3.1. Overall

The overall structure of Graph Adaptive Attention Network (GAAN) is illustrated in Figure 2. The process starts with a graph where each node represents an entity and edges denote relationships between them. In the encoding stage, the input graph's information is transformed into features for each node. The input layer processes these encoded features. Within the graph layer and hidden layer, the network uses adaptive attention mechanism(AAM) to assign varying weights αij to the neighbors of a central node hi. These weights signify the importance of each neighboring node hj in contributing to the central node's updated features. The updated features are computed through an attention-based weighted average of neighboring node features. Finally, the output layers aggregate the refined node features to perform node classification, as illustrated in the output graph with colored nodes.

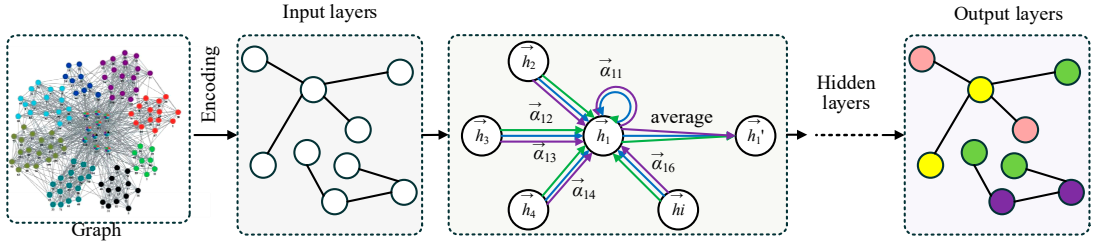

Figure 2. The structure of GAAN.

3.2. Graph Layers with AAM

The specific computational process can be represented by Equation 1 to Equation 5.

$$h_{ij} = concat\left(\boldsymbol{W}\vec{h}_i, \boldsymbol{W}\vec{h}_j\right), \qquad i,j \in N \tag{1}$$

where $\vec{h}_i$ and $\vec{h}_j$ represent the ith and jth nodes in the graph, respectively. $\boldsymbol{W}$ is the shared weight matrix to uniform the node features. h_{ij} fuses the features of the ith and jth nodes. Based on h_{ij}, we can calculate the basic AAM between the ith and jth nodes with the following Equation 2.

$$e_{ij} = \frac{a * h_{ij}}{\sqrt{D_i * D_j}} \tag{2}$$

where D_i and D_j represent the degree of ith and jth nodes, respectively. a is the weight vector to reshape h_{ij}. e_{ij} is the weight coefficient between the ith and jth nodes. Then, we adopt the softmax function to normalize e_{ij}, as shown specifically in Equation 3.

 doi:10.20944/preprints202406.0939.v1

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp\left(LeakyReLU(e_{ik})\right)}{\sum_{k\epsilon h_i} exp\left(LeakyReLU(e_{ik})\right)} \quad (3)$$

Based on the normalized weight coefficients between nodes obtained from Equations 1 to 3, we can perform the graph convolution layer computation, as shown in Equation 4.

$$\vec{h}_i' = \sigma(\sum_{j\epsilon\mathcal{N}_i} \alpha_{ij}\, \boldsymbol{W}\vec{h}_j) \quad (4)$$

Building on Equation 4, we utilize Multi-head Graph Convolution(MHGC), which involves performing the computation of Equation 4 with MHGC and then averaging these results to obtain the node features of the subsequent layer. The computation process is shown in Equation 5.

$$\vec{h}_i' = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\epsilon\mathcal{N}_i} \alpha_{ij}^k\, \boldsymbol{W}^k\vec{h}_j) \quad (5)$$

Based on the aforementioned formula, we have completed the normalized weight coefficients and inter-layer computation processes for GAAN. We have designed a single hidden layer, thus constructing the GAAN with a total of two layers.

3.3. Cross Entropy Loss

In the node classification task, our goal is to correctly categorize each node into predefined categories. Suppose our model outputs the predicted probability that node v_i belongs to each category as $\hat{y}_i$ and the true category label as v_i, where $\hat{y}_i \in \mathbb{R}^C$ and $y_i \in \mathbb{R}^C$, C is the number of categories and N is the number of nodes. The cross-entropy loss function is defined as Equation 6.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{C} y_{i,c}\, log(\hat{y}_{i,c}) \quad (6)$$

where N is total number of nodes. C is the number of categories. y_i represents true category distribution of node v_i. $\hat{y}_i$ is predicted category distribution.

The derivation process of the cross-entropy loss function is as follows. At the final output layer of the network, the feature representation h_i of each node v_i will pass through a fully connected layer and the softmax function will be applied to generate the predicted probability distribution. The model output will be the predicted probability distribution of the node $\hat{y}_i$. It is described in Equation 7.

$$\hat{y}_{i,c} = \frac{exp(h_i \cdot W_c + b_c)}{\sum_{c'=1}^{C} exp(h_i \cdot W_{c'} + b_{c'})} \quad (7)$$

where W_c and b_c are the weight and bias of category, respectively. h_i is the feature representation of node v_i

The cross entropy loss measures the difference between the true category distribution and the predicted probability distribution. For each node v_i, the loss is defined as Equation 8.

$$\mathcal{L}_i = -\sum_{c=1}^{C} y_{i,c}\, log(\hat{y}_{i,c}) \quad (8)$$

In order to measure the categorization performance over the whole graph, the average loss over all nodes needs to be calculated as Equation 6.

Specific process could be described as the following four steps.

(1) Encoding features: encode the node features of the input graph to get the initial feature representation of the nodes.

(2) Attention mechanism: in the hidden layer, use the attention mechanism to weight the average of neighboring nodes to get the updated node feature representation h_i'.

(3) Fully Connected Layer: In the output layer, the updated node feature representation is transformed into a fully connected transformation and the softmax function is applied to get the predicted probability $\hat{y}_i$.

 doi:10.20944/preprints202406.0939.v1

(4)Calculate the loss: use the cross-entropy loss function to calculate the difference between the true category distribution y_i and the predicted probability distribution $\hat{y}_i$, and average to get the overall loss $\mathcal{L}$.

Through the above process, we are able to effectively classify the nodes in the graph and optimize the network parameters through back propagation to achieve accurate classification of nodes.

4. Experiments

4.1. Datasets

Cora, Citeseer, and Pubmed are widely used citation network datasets in graph-based machine learning research. In this paper, we use the Cora, Citeseer and Pubmed datasets, the specific statistics of which are shown in Table 1. Cora consists of 2,708 machine learning publications categorized into 7 classes, with each paper cited by or citing other papers. Each node represents a publication, and edges denote citation relationships. Nodes have 1,433 binary word attributes. Citeseer includes 3,327 research papers grouped into 6 categories. Similar to Cora, nodes signify publications, and edges indicate citations. Each node has a 3,703-dimensional binary word vector representing the presence or absence of specific words. Pubmed contains 19,717 scientific publications from the PubMed database, classified into 3 diabetes-related categories. The nodes represent papers, connected by citation edges. Each node is described by a TF-IDF weighted word vector from a 500-word dictionary.

Table 1. Summary of the datasets used in our experiments.

Dataset	Nodes	Edges	Features per node	Classes
Cora	2708	5429	1433	7
Citeseer	3312	4723	3703	7
Pubmed	19717	44338	500	3

4.2. Ablation Experiments

As shown in Table 2, we conduct ablation experiments on Cora dataset. AAM and MHGC respectively increased by 1% and 1.2%. Our GAAN has achieved an excellent performance of 85.6%.

Table 2. Ablation experimental results on Cora. 'n' and 'n_hidden' represent the number of graph convolution heads and the dimensionality of node feature vectors in hidden layers, respectively.

	AAM	MHGC(n=8)	n_hidden			Accuracy(%)
			64	96	128	
Group1	×	×	√	×	×	83.9
	×	×	×	√	×	83.5
	×	×	×	×	√	83
Group2	×	√	√	×	×	84.6
	×	√	×	√	×	85.1
	×	√	×	×	√	84.6
Group3	√	×	√	×	×	84.9
	√	×	×	√	×	84.3
	√	×	×	×	√	84.6

 doi:10.20944/preprints202406.0939.v1

	√	√	√	×	×	84.6
Group4	√	√	×	√	×	85.4
	√	√	×	×	√	**85.6**

4.3. Comparison with Other Methods

The Table 3 presents the classification accuracy (%) of various methods on three datasets: Cora, Citeseer, and Pubmed. The methods compared include MLP, SemiEmb, DeepWalk, ICA, Planetoid, Chebyshev, GCN, MoNet, and GAT. GAAN (no MHGC) achieves an accuracy of 84.9% on Cora, 74.3% on Citeseer, and 79.5% on Pubmed. GAAN (with MHGC) further improves the performance, achieving the highest accuracy of 85.6% on Cora, 75.5% on Citeseer, and 80.5% on Pubmed, indicating the effectiveness of integrating MHGC. GAT also shows strong results with 83.7% on Cora, 73.2% on Citeseer, and 79.3% on Pubmed. According to Table 3, we can see that GAAN gets better performance than other methods.

Table 3. Compared experimental results on Cora, Citeseer, and Pubmed.

Method	Cora	Citeseer	Pubmed
MLP[26]	55.1	46.5	71.4
SemiEmb[27]	59	59.6	71.7
DeepWalk[28]	67.2	43.2	65.3
ICA[29]	75.1	69.1	73.9
Planetoid[30]	75.7	64.7	77.2
Chebyshev[8]	81.2	69.8	74.4
GCN[1]	81.5	70.3	79
MoNet[19]	82.2	_	79.1
GAT[17]	83.7	73.2	79.3
GAAN(no MHGC)	84.9	74.3	79.5
GAAN(with MHGC)	**85.6**	**75.5**	**80.5**

5. Conclusion and Future Work

We have proposed AAM and MHGC to construct GCNA, which solves the differences between neighbor nodes to the central node. Experimental results show that our method is superior in accuracy.

Over-smoothing occurs when multi-layers are stacked, leading to the features of all nodes being almost the same. However, many situations are necessary to capture the features of distant neighbors. Therefore, stacking multiple layers of GCN is inevitable. The strategy of stacking multi-layers, designed to prevent over-smoothing, is urgent.

Author Contributions: Zhao Chen is the only author for this paper, and Zhao Chen has completed all work about this paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability Statement: The Cora dataset is available at the following https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz. The CiteSeer dataset is available at the following https://linqs-data.soe.ucsc.edu/public/lbc/citeseer.tgz. The PubMed dataset is available at the following https://linqs-data.soe.ucsc.edu/public/lbc/pubmed_diabetes.tgz.

Conflicts of Interest: The authors declare no conflicts of interest.

Reference

1. Kipf, Thomas and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks." ArXiv abs/1609.02907 (2016): n. pag.
2. Yao, X.; Yang, H.; Sheng, M. Feature Fusion Based on Graph Convolution Network for Modulation Classification in Underwater Communication. Entropy 2023, 25, 1096. https://doi.org/10.3390/e25071096
3. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.
4. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
5. Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arxiv preprint arxiv:1409.2329* (2014).
6. Graves, Alex, and Alex Graves. "Long short-term memory." *Supervised sequence labelling with recurrent neural networks* (2012): 37-45.
7. Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." *arxiv preprint arxiv:1312.6203* (2013).
8. Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." *Advances in neural information processing systems* 29 (2016).
9. Hammond, David K., Pierre Vandergheynst, and Rémi Gribonval. "Wavelets on graphs via spectral graph theory." *Applied and Computational Harmonic Analysis* 30.2 (2011): 129-150.
10. Henaff, Mikael, Joan Bruna, and Yann LeCun. "Deep convolutional networks on graph-structured data." *arxiv preprint arxiv:1506.05163* (2015).
11. Levie, Ron, et al. "Cayleynets: Graph convolutional neural networks with complex rational spectral filters." *IEEE Transactions on Signal Processing* 67.1 (2018): 97-109.
12. Shuman, David I., et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." *IEEE signal processing magazine* 30.3 (2013): 83-98.
13. Bianchi, Filippo Maria, et al. "Graph neural networks with convolutional arma filters." *IEEE transactions on pattern analysis and machine intelligence* 44.7 (2021): 3496-3507.
14. Defferrard, M., Milani, F., Gusset, F., & Perraudin, N. (2018). Deep Networks on Toric Graphs. arXiv preprint arXiv:1808.03965.
15. Spielman, Daniel. "Spectral graph theory." *Combinatorial scientific computing* 18 (2012): 18.
16. Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems* 30 (2017).
17. Veličković, Petar, et al. "Graph attention networks." *arxiv preprint arxiv:1710.10903* (2017).
18. Monti, Federico, Michael Bronstein, and Xavier Bresson. "Geometric matrix completion with recurrent multi-graph neural networks." *Advances in neural information processing systems* 30 (2017).
19. Monti, Federico, et al. "Geometric deep learning on graphs and manifolds using mixture model cnns." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
20. Wu, Felix, et al. "Simplifying graph convolutional networks." *International conference on machine learning*. PMLR, 2019.
21. Xu, Keyulu, et al. "Representation learning on graphs with jumping knowledge networks." *International conference on machine learning*. PMLR, 2018.
22. Xu, Keyulu, et al. "How powerful are graph neural networks?." *arxiv preprint arxiv:1810.00826* (2018).
23. Liao, Renjie, et al. "Lanczosnet: Multi-scale deep graph convolutional networks." *arxiv preprint arxiv:1901.01484* (2019).
24. Yan, Sijie, Yuanjun Xiong, and Dahua Lin. "Spatial temporal graph convolutional networks for skeleton-based action recognition." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. No. 1. 2018.
25. Atwood, James, and Don Towsley. "Diffusion-convolutional neural networks." *Advances in neural information processing systems* 29 (2016).

 doi:10.20944/preprints202406.0939.v1

26. Taud, Hind, and Jean-Franccois Mas. "Multilayer perceptron (MLP)." *Geomatic approaches for modeling land change scenarios* (2018): 451-455.
27. Weston, Jason, Frédéric Ratle, and Ronan Collobert. "Deep learning via semi-supervised embedding." *Proceedings of the 25th international conference on Machine learning*. 2008.
28. Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014.
29. Bandyopadhyay, Sanghamitra, et al. "Link-based classification." *Advanced methods for knowledge discovery from complex data* (2005): 189-207.
30. Yang, Zhilin, William Cohen, and Ruslan Salakhudinov. "Revisiting semi-supervised learning with graph embeddings." *International conference on machine learning*. PMLR, 2016.