ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 108-116
doi: 10.18178/wcse.2019.06.016

Hybrid Graph Convolutional Networks for Semi-Supervised Classification

Dongyang Bao [1], Wei Zheng [2 +] and Wenxin Hu [1 +]

[1] School of Computer Science and software Engineering, East China Normal University
[2] Information Technology services, East China Normal University

Abstract. In recent years, Graph Convolutional Network (GCN) have been successfully applied to many graph classification problems. It has the capability to learn many types data that Convolutional Neural Networks (CNN) cannot handle, such as irregular data. However, we found that GCN can not completely capture the graph structure information and especially for inference on data efficiently. In this paper, we analyze the advantages and disadvantages of several models and propose two different methods of combining models. Based on that, we propose a new model by using ensemble learning Based on GCN. This model has the ability to capture the advantages of multiple models. Finally, we conduct our experiment on several datasets, and the experimental results show that our approach is effective.

Keywords: Graph Neural Network, Representation Learning, Data Mining.

1. Introduction

Network is the most common information carrier and form in modern society. It exists in a board diversity of real-world scenarios, naturally, e.g., social networks [1], citation networks, protein-protein interaction networks [6], knowledge networks [5] and other networks [8]. Based on the widespread of complex information networks, research and analysis of such network has very high academic value and potential application value. Every network can be abstracted into a directed or undirected graph structure with weight. Traditional machine approaches often depend on summary graph statistics to extract structural information from graphs, [2], kernel functions [3], or well-designed manual or engineered features to evaluate graph local neighbourhood node structures [4] and limited by their high cost and inflexibility. However Deep Neural Network (DNN) which have recently proven to be powerful tools in many fields, such as natural language processing, computer vision and presentation learning, etc. Because the word embedding and other representation learning success, DeepWalk [9] is considered as the first graph embedding method based on representation learning, and SkipGram is applied to generate random walk. The goal of SikpGram model is to maximize the probability of word co-occurrence in windows and the probability of observing the neighbourhood of nodes under embedded conditions. node2vec [10], LINE [11] and other similar methods are also effective models. However, the parameters of these methods are in a low computational efficiency, because they are not shared among the nodes in the encoder, In the other hand, the embedding method directly lacks generalization ability in many scenarios [12].

Convolutional neural networks (CNN) which have use in computer vision and other fields, it is successful in the cases of data with basic Euclidean structure and in the cases where the invariance of these structures are built into the network used to model them. CNN increasing parameter sharing by performing spatial convolutions on data to exploit the regular structure data, and lowering their complexity. Recently, a

+ Corresponding author. Tel.: +86-021-62233797;
E-mail address: wzheng@admin.ecnu.edu.cn, wxhu@cc.ecnu.edu.cn

great number of methods have emerged to re-define the irregular structure data(graphics structure) convolution inspired by the success of CNN in the field of computer vision. These methods belong to graph convolutional network (GCN) [16] in geometric deep learning [13]. GCN has been applied to several real-world scenarios, successfully, including matrix completion [14], program induction [15], semi-supervised learning [16], modeling relational data [17], etc. GCN has the capability to use the graphical structure information to extract local features and process irregular data directly of one-time connection (described more in Section 2).

As same as other neural networks, GCN contains only one convolution model and interleaves linear layers, and with only one result output after each convolution. But the size of each layer, the activation function of each layer, and the number of layers all have an effect on the model's results, which are complex hyperparameters for the model. Although there are already many systematic methods and tools to adjust the hyperparameters, the adjusted parameters may still be unsatisfactory. In this paper we present a new model Hybrid Graph Convolutional Network (HGCN) inspired by ensemble learning based on GCN. We combine the advantages of multiple models and optimize the results.

The contribution of our paper can be summarized as follows:

- We analyze the advantages and disadvantages of the Graph Convolutional Network and Neural Network in the semi-supervised learning tasks.
- We propose two different model combining algorithms in this task, and then, we compare and analyze the two algorithms.
- Based on the above two points, we propose a novel graph semi-supervised classification model called the HGCN, which has the capability to combine the advantages of multiple models effectively.
- We conduct extensive experiments on citation network. Experimental results prove the effectiveness and efficiency of the propose HGCN model.

The rest of this paper is organized as follows. We describe preliminaries in Section 2. Our proposed model HGCN is presented with detailed analysis in Section 3. In section 4 we analysis the experimental results. At last we make a conclusion in Section 5.

2. Preliminaries

In this section, we provide definitions of basic graphical concepts and briefly describe the concept and derivation of GCN. the definitions will be used next.

2.1. Graph and related concepts

A generic undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, ..., N\}$ is the set of $\mathcal{N}$ vertices, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges connecting the vertices. The graph is equivalently described by the adjacency matrix or edges weight matrix $\mathcal{A} \in \mathbb{R}^{N \times N}$, where its generic element $a_{ij} \geq 0$ if and only if nodes i and j are connected. A graph signal [31] is a function $f : V \to \mathbb{R}^F$ associating to vertex n a F-dimensional vector $\mathbf{x}_n$ and each dimension being denoted as a channel.

2.2. Graph convolutional network

The layer-wise propagation rule of Multi-layer Graph Convolutional Network (GCN) can be defined as:

$$Y^{(l+1)} = \sigma(M^{-\frac{1}{2}} S M^{-\frac{1}{2}} Y^{(l)} W^{(l)}) \tag{1}$$

where, A is the adjacency matrix, $S = A + I_N$ is mean all nodes of graph $\mathcal{G}$ are connected to themselves. $W^{(l)}$ is a weight matrix . M is a degree matrix, it is also a diagonal matrix. $M_{ii} = \sum_j S_{ij}$. $\sigma(\cdot)$ is an activation function. $Y^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the l^{th} layer and $Y^{(0)} = X$.

In the graph domain, we use the normalized Laplacian matrix $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ to define the convolution operation. $L = U \Lambda U^T$ is denoted eigendecomposition, and U is a matrix used to collect all of

eigenvectors of L. Λ is associated eigenvalues and is also a diagonal matrix. $h_\theta = diag(\theta)$, where $\theta \in \mathbb{R}^N$, is a filter. Defined a spectral convolution on graphs as a multiplication of signal x with filter in the Fourier domain:

$$h_\theta \star x = U h_\theta U^T x \tag{2}$$

Here $U^T x$ is the representation of x in the Fourier domain. the graph Fourier transform of x is define as $x = U^T x$. Because the eigenvectors form an orthonormal basis, an inverse graph Fourier transform can define as $X_i = U\hat{X}_i$ when we regard eigenvalues as frequencies, and also. In the equation 2, multiplication with the eigenvector matrix U and calculate its results computational complexity is $\mathcal{O}(N^2)$, so choice of $h_\theta(\cdot)$ determines the complexity of training. To alleviate this problem Hammond et al. [18] proposes to make the problem easy to handle by using a Chebyshev polynomial filter $T_k(x)$ up to K^{th} order on the eigenvalues:

$$h_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta' T_k(\hat{\Lambda}) \tag{3}$$

where $\hat{\Lambda} = 2\frac{\Lambda}{\lambda_{max}} - I$ and the λ_{max} is the largest eigenvalue of L. $T_k(\cdot)$ is the Chebyshev polynomial of order k defined as:

$$T_k(x) = 2xT_{K-1}(x) - T_{k-2}(x) \tag{4}$$

with $T_0(x) = 1$ and $T_1(x) = x$. In order to simplify expression results and avoid more complex recursive relationships, it is proposed in paper [16] to set $k = 1$ and $\theta'_0 = \theta'_1$ for all channels and further approximate $\lambda_{max} \approx 2$. Bring these expressions and values into the original equation (3), we get $h_\theta \star x \approx \theta(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x.$

As an example, a two layers GCN is given by:

$$\hat{Y} = \sigma_1(\hat{A}\sigma_0(\hat{A}xW^{(0)})W^{(1)}) \tag{5}$$

here, σ_0 and σ_1 are activation function, they can be *ReLU* and *softmax* respectively. A is a symmetric adjacency matrix. $W^{(0)} \in \mathbb{R}^{C \times H}$ and $W^{(1)} \in \mathbb{R}^{H \times F}$ are the input-to-hidden and hidden-to-output weight matrix. H is the number of features in hidden layer, and $\hat{A} = M^{-\frac{1}{2}} S M^{-\frac{1}{2}}$.

3. Methodology

In the semi-supervised classification task, GCN can extract the structure information of the graph well, but the ability to fit the category information of the node is poor. The NN have better ability to fit the node category information, but the ability to obtain deeper graph structure information is poor, such as k-orders. In the models of GCN and NN, the number of nodes in each layer, the activation functions and other hyperparameters are fixed. There are two problems with this. The first question is how to choose the model, the second is how to set better hyperparameters. There are also a lot of techniques or methods for adjusting hyperparameters and model selection, but the best choice is not to make a choice. This is the best choice if we take advantage of all the models and combine the advantages of all models. However, this is the idea of ensemble learning. Therefore, we propose the HGCN model to combine the advantages of the two models. In the remainder of this section, we review the two models, and we will describe the HGCN model in detail later.

3.1. Two types of models

Neural Network (NN) Under normal circumstances, we can use the traditional Neural Network (NN) to predict the label y by input x. An L-layer neural network can be defined as:

$$h_l = \sigma_l(W_l h_{l-1} - b_l) \tag{6}$$

here, σ_l is nonlinear activation function, $h_{l-1} \in \mathbb{R}^{N_{l-1}}$ is the input form (l-1)-layer to l-layer, $w_l \in \mathbb{R}^{N_l \times N_{l-1}}$ are trainable weights, $b_l \in \mathbb{R}^{N_l}$ is bias. manifold regularization [20], neural graph machines [23], ladder networks [21], pseudo-labels [22], AtlasRBF [19], and many other techniques can be used to train neural networks in unlabeled data.

Graph Convolutional Network (GCN) As previously stated, the GCN efficiently obtains the result by converting the convolution of the signal on graph domain into the inner product on frequency domain and simplifying the operation by Chebyshev polynomial.

3.2. Hybrid graph convolutional network

Ensemble learning which integrates multiple learning algorithms is famous for better performance in machine learning and it has been successful applied to many scenarios, include classification, computer vision, bioinformatics, etc. We propose a new model, Hybrid Graph Convolutional Network (HGCN), using ensemble learning based on GCN, and combining the advantages of convolutional neural networks and traditional neural networks.

HGCN-Model1 In our model (HGCN), 1. We adjust the output of each different model, 2. Give different weights to the algorithms of different parameters and different models. 3. Integrate different algorithms and optimize results. Based on this, the model can be defined as:

$$\hat{Y} = \sigma(\sum_{p=1}^{P} w_p h_p) \tag{7}$$

where σ is nonlinear activation function, P is the number of algorithms, h_p is the output of the algorithm for each model with different parameters and different models. w_p is the trainable weight to different algorithms. Note that the output of each algorithm is not a label prediction for each input, it is only an intermediate variable in the process of join into the ensemble learning. Because if it is predicted for a labels, the dimension of the outputs will reduce (In general, the number of labels will be smaller than the number of nodes in the hidden layer) and it will lose some important features. The model convergence speed is reduced, the details is shown in Fig. 1. We define this model is **HGCN-Model1**.

HGCN-Model2 However, if we just linearly sum the algorithms for different parameters and different models, the output of different models may be in different intervals, so those models with larger output values may overwrite the model with smaller output values after weighted summation, it will make the results not get the advantages of all models and the accuracy of the overall results is degraded. Therefore, we sum the algorithms of different parameters of the same model and combine the results. So the model is defined as:

$$\hat{Y} = \sigma(\sum_{m=1}^{M} w_m \sigma_o(\sum_{p=1}^{P_m} w_{mp} h_{mp})) \tag{8}$$

where σ and σ_o is activation function, M is the number of models, and P_m is the number of different parameters of the same model m (we call it an algorithm). w_m is the trainable weight to different models, w_{mp} is the trainable weight to different parameters of the same model m. h_{mp} is the output of the algorithm for each model with different parameters and we see the output of each algorithm as a hidden layer of the overall model. We define this model is **HGCN-Model2** (**HGCN**). The model structure is shown in Fig. 1.

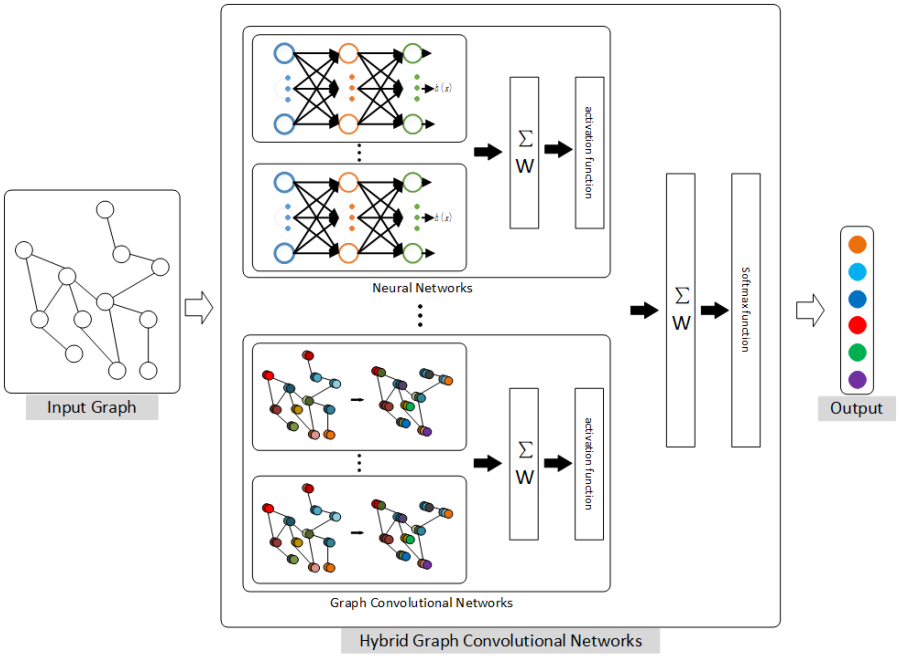

Fig. 1: Hybrid Graph Convolutional Network (HGCN) model structure

As an example, from the equation (5) and equation (6), a HGCN with two models (traditional neural networks with two layers and graph convolutional network with two layers) is given by:

$$\hat{Y} = \sigma(w_1\sigma_o(\sum_{p=1}^{P_1} w_{1p}\sigma_1(\hat{A}\sigma_0(\hat{A}XW_{1p}^{(0)})W_{1p}^{(1)})) + w_2\sigma_o(\sum_{p=1}^{P_2} w_{2p}\sigma_3(W_{2p}^{(3)}\sigma_2(W_{2p}^{(2)}X - b_{2p}^{(0)}) - b_{2p}^{(1)}))) \quad (9)$$

where σ is softmax function, w_i (i = 1, 2) is weight of the two models, σ_0 and σ_1 is the action function. p_1 is the number of graph convolutional network, p_2 is the number of traditional neural network. w_{1p} and w_{2p} are the weights of different algorithms for different models. We assign different weights to each algorithm for each model and integrate their results. Finally, we use softmax to calculate the results.

4. Experiment

Our experiment mainly investigate the effectiveness of the HGCN over several kinds of datasets to validate that it can perform better than HGCN-Model1 other model that already exists. In the remainder of this section, we introduce the experimental datasets and the baseline methods, analyse the experimental details and results at the end of this section.

4.1. Datasets and baseline methods

Citeseer and Pubmed [29] are the citation network. Every dataset contains sparse bag-of-words feature vectors for each document , and the vectors are describing the relative frequencies of words, the links of citation between documents are in a list for every dataset. We construct a binary, symmetric adjacency matrix $\mathcal{A}$ from the citation links, and each document has a class label. We closely follow the experimental setup in paper [25]. The statistics of the dataset are in Table 1, the last column of Table 1 is the labeled nodes percentage in the datasets.

Table 1: Dataset statistics

Dataset	Nodes	Edges	Classes	Features	Label rate
Citeseer	3,327	4,732	6	3,703	3.6%
Pubmed	19,717	44,338	3	500	0.3%

We test our model in semi-supervised document classification in citation networks, and compared the baseline method of Yang et al. [25] and Kipf et al. [16] including:

- DeepWalk [9], a sampling method based on random walk, using skip-gram on graph embedding.
- SemiEmb [26], Semi-supervised Embedding, combining an embedding-based regularizer with a supervised learner to perform semi-supervised learning.
- LP [27], Label Propagation, a method of parameter learning based on entropy minimization.
- ManiReg [28], attempt to use the geometry of the probability distribution by assuming that its support has the geometric structure of a Riemannian manifold.
- Planetoid [25], develop both transductive and inductive variants in the method.
- ICA [24], iterative classification algorithm using approximate reasoning.
- GCN [16], perform Fourier transform on the graph information and perform semi-supervised learning.
- KAF-GCN [7], based on GCN, and use kernel activation function that it is a non-parametric model.
- NN, traditional dense neural network.

a full all the baselines description can be found in the original papers.

4.2. Experimental settings

We use Citeseer, Cora and Pubmed dataset to evaluate our network. All data sets have been described above. The experimental model uses two sub-models, including GCN and NN. GCN and NN contain 18 different parameters, respectively. It means we get 36 different cases of algorithms. We optimize a cross-entropy loss given by:

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} ln \hat{Y}_{lf}, \tag{10}$$

here, $\mathcal{Y}_L$ is the set of have labels node. The loss is optimized using Adam algorithm [30] and train neural network weights w_1, w_2, w_{mp}, $W_{mp}^{(0)}$, $W_{mp}^{(1)}$, $W_{mp}^{(2)}$, $W_{mp}^{(3)}$ and bias $b_{mp}^{(0)}$, $b_{mp}^{(1)}$ in equation (9).

4.3. Experimental results

In the experiment, we conducted a comparative experiment on HGCN-Model1, HGCN-Model2 and other baseline models. In the classification task, we exceeded the GCN model by 1.2% in the public dataset Citeseer, and also achieved good results in the dataset Pubmed. The experimental results are shown in Table 2 and the loss situation is shown in Fig. 2.

In Table 2, the NN model does not perform well in semi-supervised classification learning tasks, the accuracy rate is only 56.1% and 70.6% on datasets, while the accuracy of GCN is 70.3% and 79.0%. Under the integration of various models, HGCN-Model1 and HGCN-Model2 has the capability to take advantage of each algorithm, that GCN is good at extract the structure information of the graph, the NN have better ability to fit the node category information, based on these, HGCN-Model1 and HGCN-Model2 extract the hidden feature information in the training data more effectively, and converge faster.

In Fig. 1, since the NN model is merged in HGCN-Model1 and HGCN-Model2, and NN can obtain the node information of the node more effectively, we can see that HGCN-Model1 and HGCN-Model2 converge faster than GCN. In HGCN-Model1, it directly integrates each algorithm, the result interval of the algorithm of different models may be different. Therefore, when learning weights, in equation (7) is not stable in the process of learning. This will cause the learning speed rate to decrease and the convergence rate to slow down. In HGCN-Model2, the nonlinear activation functions are used between the algorithms of different models, and the results of each algorithm are uniformly represented. In the process of learning, the parameters are more stable and the convergence speed is faster.

Table 2: Results in terms of classification accuracy over the test set

Method	Citeseer	Pubmed
ManiReg	60.1	70.7
SemiEmb	59.6	71.1

LP	45.3	63.0
DeepWalk	43.2	65.3
ICA	69.1	73.9
Planetoid	64.7	77.2
NN	56.1	70.6
GCN	70.3	**79.0**
KAF-GCN	70.9	-
HGCN-Model1	70.5	**79.0**
HGCN-Model2	**71.5**	**79.0**

5. Conclusions

Network representation on large-scale information networks has important applications in tasks such as node classification and link prediction, and research is of great significance. In this paper, we proposed a new network representation method that combine multiple models and multiple model parameters including graph convolutional network and traditional neural networks to learn graph representations and to perform graph classification. Our Hybrid Graph Convolutional Network (HGCN) was tested on multiple datasets and the results showed that the model performed well. However, there is a limitation with the current work. Since the adjacency matrix that representing the graph structure should be input into the model during the initial training process, our model is difficult to extend. So in the future, we may be able to dynamically add network structures and other information by increasing the convolution depth, and the model should be improved so that it is an online learning algorithm which builds useful incremental result.

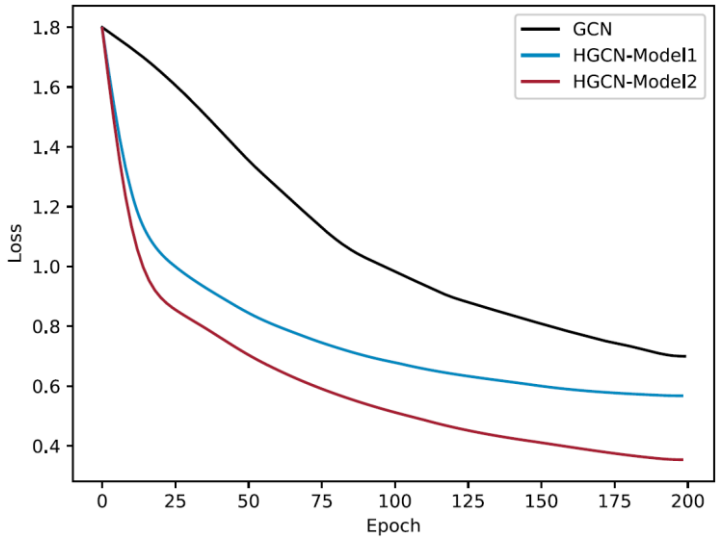

Fig. 2: Hybrid Graph Convolutional Network (HGCN) model structure

6. Acknowledgements

This research is funded by the Science and Technology Commission of Shanghai Municipality (No. 18511106202 and 18511105502), and by Xiaoi Research. The computation is performed in ECNU Public Platform for Innovation (001).

7. References

[1] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Conference on Neural Information Processing Systems*, 2017, pp. 1024–1034.

[2] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks, *Social Network Data Analytics*, pp. 115–148.

[3] S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, and K.M. Borgwardt. Graph kernels, *JMLR*, 2010, 11:1201–1242.

[4] D. Liben-Nowell, J. Kleinberg. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology*, 2007, 58(7):1019–1031.

[5] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto. Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach, *IJCAI*, 2017, 1802–1808.

[6] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur. Protein interface prediction using graph convolutional networks, *NIPS*, 2017, 6530–6539.

[7] S. Scardapane, S. Van Vaerenbergh, D. Comminiello and A. Uncini, Improving Graph Convolutional Networks with Non-Parametric Activation Functions. *2018 26th European Signal Processing Conference (EUSIPCO)*, Rome, 2018, pp. 872-876.

[8] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. *NIPS*, 2017.

[9] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *SIGKDD*, 2014, pp. 701–710.

[10] A. Grover and J. Leskovec, node2vec: Scalable feature learning for networks. *SIGKDD*, 2016, pp. 855–864.

[11] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, Line: Large-scale information network embedding. *WWW*, 2015, pp. 1067–1077.

[12] Zhou, Jie , et al. Graph Neural Networks: A Review of Methods and Applications. 2018.

[13] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Process. Mag.*, 2017, vol. 34, no. 4, pp. 18–42.

[14] R. v. d. Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion, 2017.

[15] M. Allamanis, M. Brockschmidt, and M. Khademi. Learning to represent programs with graphs. *ICLR*, 2018.

[16] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[17] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks, 2017.

[18] David K. Hammond, Pierre Vandergheynst, and Remi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 2011, 30(2):129–150.

[19] Pitelis, N., Russell, C., and Agapito, L. Semi-supervised learning using an unsupervised atlas. *Machine Learning and Knowledge Discovery in Databases ECML PKDD*, 2014, pp. 565–580.

[20] Y. Yuan, L. Mou, and X. Lu, Scene recognition by manifold regularized deep learning architecture. *IEEE Trans. Neural Netw. Learn. Syst*, 2015, vol. 26, no. 10, pp. 2222–2233.

[21] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. *Advances in Neural Information Processing Systems*, 2015, pp. 3546–3554.

[22] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *Workshop on Challenges in Representation Learning, ICML*, 2013, vol. 3, p. 2.

[23] T. D. Bui, S. Ravi, and V. Ramavajjala. Neural graph machines: Learning neural networks using graphs. *arXiv preprint arXiv:1703.04818*, 2017.

[24] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008, 29(3):93.

[25] Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *International Conference on Machine Learning (ICML)*, 2016.

[26] Jason Weston, Frederic Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. *Neural Networks: Tricks ofthe Trade*, 2012, pp. 639–655.

[27] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *In International Conference on Machine Learning (ICML)*, 2003, volume 3, pp. 912–919.

[28] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric frame- work for learning from labeled and unlabeled examples. *Journal of machine learning research (JMLR)*, 2006, 7(Nov):2399–2434.

[29] Qing Lu and Lise Getoor. Link-based classification. *International Conference on Machine Learning (ICML)*, volume 3, 2003, pp. 496–503.

[30] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. *MIT Press*, 2016.

[31] A. Sandryhaila and J. M. Moura. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *IEEE Signal Process. Mag.*, 2014, vol. 31, no. 5, pp. 80–90.