Semi-Supervised Node Classification on Graph via Inconsistent Nodes Correction

Shuhao Shi
PLA information engineering university
Zhengzhou, China

Jian Chen
PLA information engineering university
Zhengzhou, China

Kai Qiao
PLA information engineering university
Zhengzhou, China

Shuai Yang
PLA information engineering university
Zhengzhou, China

Linyuan Wang
PLA information engineering university
Zhengzhou, China

Bin Yan*
PLA information engineering university
Zhengzhou, China
ybspace@hotmail.com

ABSTRACT

This article is a work-in-progress paper. In traditional graph convolutional networks (GCNs), node feature aggregation in graph convolutional learning is guided only by topology graphs. In reality, both network topology and node features offer unique and valuable information. Using topology alone cannot obtain fully and completely accurate neighborhood information. This paper advocates a representation learning method with output correction for GCNs (OC-GCN), simultaneously using topological structure and node features information. Specifically, we use two GCN encoders to extract node embeddings in feature space and topology space. Then, determine inconsistent nodes by the pseudo-labels generated by the two models. Finally, we use the representations of consistent neighbors to regenerate representations of inconsistent nodes in feature and topology views. Our experiments have shown that the OC-GCN can significantly improve the classification accuracy of inconsistent nodes in feature and topology views. We conducted extensive experiments on the benchmark datasets and demonstrated that OC-GCN is substantially better than state-of-the-art baselines at different label rates.

CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning settings**; *Neural networks.*

KEYWORDS

graph convolutional networks, output correction, semi-supervised classification

ACM Reference Format:
Shuhao Shi, Jian Chen, Kai Qiao, Shuai Yang, Linyuan Wang, and Bin Yan. 2022. Semi-Supervised Node Classification on Graph via Inconsistent Nodes Correction. In *Proceedings of 17th International Workshop on Mining and Learning with Graphs (MLG'22). (MLG'22).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MLG'22, Aug 15, 2022, Washington DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, graph convolutional networks (GCNs) have achieved remarkable success on graph-related tasks [3, 13, 21, 23]. The typical GCN [12] and its classical variants [1, 1, 8, 18] follow a message passing framework, which learns embeddings through feature aggregation from its topological neighbors.

GCN defines the convolution using a simple linear function of the graph Laplacian on a topology graph in the semi-supervised classification task, but this limits its capability to aggregate the information from nodes with similar features. The improvement of GCN focus on extracting more information from the topology graph. GraphSAGE [8] improves the full graph sampling to node-centric neighbor sampling. GAT [18] introduces the attention mechanism in message propagation, assigning weights to different neighbors. MixHop [1] learns by repeatedly mixing feature representations of the neighborhood at different distances. In fact, both graph structure and node features contain important information [11]. Recently, some studies have used both topology graph and feature graph as input, exploiting both feature and structural information to improve the model's performance. Wang et al. [12] set up a series of experiments to demonstrate that MLP [16] and DeepWalk [17] performed better than GCN when the correlation between node labels and features or topology structure were strong. [14] proposes a self-supervised graph representation learning method by maximizing the agreement of embeddings of the same node in the topology graph and the feature graph. Similarly, [5] proposed a self-supervised loss function to force the model to learn shared information about the feature and topological space.

Specifically, we argue that increasing the consistency of the two views could boost the performance of downstream tasks. In this paper, we advocate a representation learning method with output correction for GCNs (OC-GCN). The feature graph is first generated according to the similarity of node features. Then, with the feature graph and topology graph, we use two GCN encoders to extract node embeddings in feature space and topology space. Finally, we use the neighbors to regenerate the representations of inconsistent nodes in two views to maximize consistency between feature and

topology views. To summarize, we outline the main contributions in this paper as below:

- We are the first to explore output correction by simultaneously using topology and node feature information.
- We propose a novel method to correct output, performed only in the evaluating phase, and can significantly improve the classification accuracy of inconsistent nodes in feature and topology views.
- We conduct extensive experiments on a series of datasets and show that our method can continuously improve the model's performance on semi-supervised node classification.

2 RELATED WORKS

2.1 Graph Convolutional Neural Network

Spectral Network [2] migrates the convolution of traditional Euclidean spaces into graph networks. After that, ChebyNet [4] takes a simplified approximation for the complex Laplacian matrix. Further, GCN [12] uses a localized first-order approximation to simplify the convolution operation. GraphSAGE [8], GAT [18], MixHop [1] improve the aggregation of neighborhood nodes to extract more information from the topology graph.

However, these methods only use a single topology graph for node aggregation and do not fully utilize the rich feature information. Recently, some studies have improved the performance of GNNs by extracting embeddings from both topology graph and feature graph. Unlike the previous method that uses topology graph and feature graph to optimize the model during training, this paper introduces a method that only corrects the output during the evaluating phase.

2.2 Output Correction

Output Correction is to calibrating the outputs of original models that are likely to be misclassified. Recently, the output correction has attracted considerable attention in deep learning [7, 9, 22]. However, confidence correction has been rarely studied in deep graph learning. C&S [10] combines shallow models with correlation in the label structure during inference. In CaGCN [19], the outputs are first corrected according to the assumption that the outputs of neighboring nodes tend to be the same and then generate pseudo labels. Unlike previous methods, our proposed OC-GCN model focuses on correcting the inconsistent outputs of the two GCN encoders during evaluating phase. Then, OC-GCN can constantly improve the model's performance by increasing the consistency of the two views.

3 METHODS

This section will present our proposed method to learn node representations and correct output across feature and topology views, which consists of Multi-view Representation Learning Module, Node Feature Correction Module, and Output Aggregation Module. The overall framework is shown in Fig. 1.

3.1 Generating Feature Graph

Following [5, 20], we construct a feature graph based on node features. Cluster the node features by using KNN to capture the underlying structure of nodes in feature space. Then, the similarity between different feature representations is computed using the cosine similarity formula:

$$s\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\left|\boldsymbol{x}_i\right| \left|\boldsymbol{x}_j\right|}. \tag{1}$$

For each node, we choose top k similar nodes to form their neighbors and finally get the feature graph $\mathcal{G}_f = \left(\mathbf{A}_f, \mathbf{X}\right)$ from the original graph $\mathcal{G} = (\mathbf{A}, \mathbf{X})$. Note that the difference between $\mathcal{G}_f$ and $\mathcal{G}$ is only in the adjacency matrix.

3.2 Multi-View Representation Learning Module

Given two views of input graph $\mathcal{G}_f$ and $\mathcal{G}$, we use two GCN encoders to generate feature representations $\mathbf{Z}_f$ and topology representations $\mathbf{Z}_t$, respectively. The l-th layer output can be represented as:

$$\mathbf{Z}_f^l = \sigma\left(\tilde{\mathbf{D}}_f^{-\frac{1}{2}} \tilde{\mathbf{A}}_f \tilde{\mathbf{D}}_f^{-\frac{1}{2}} \mathbf{Z}_f^{l-1} \Theta_f^l\right). \tag{2}$$

$$\mathbf{Z}_t^l = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{Z}_t^{l-1} \Theta_t^l\right). \tag{3}$$

Where Θ_f^l and Θ_t^l are the learnable parameters of the l-th layer for two GCNs, and $\sigma(\cdot)$ denotes the activation function. $\hat{\mathbf{A}}_f = \mathbf{A}_f + \mathbf{I}$, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\hat{\mathbf{D}}_f$ and $\hat{\mathbf{D}}$ represent the degree matrix, $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$. We denote the last layer output embedding as $\mathbf{Z}_F$ and $\mathbf{Z}_T$.

In the training phase, since we have embeddings from two views, $\mathbf{Z}_F$ and $\mathbf{Z}_T$, which both contribute to the classification, we concatenate them to get the consensus representation, $\mathbf{Z} = \text{contact}\left(\mathbf{Z}_F, \mathbf{Z}_T\right)$. Convert the output embedding $\mathbf{Z}$ to predicted label $\hat{\mathbf{Y}}$ by utilizing a 2-layer MLP. Let $\mathcal{V}_L$ be the set of training nodes and $\mathbf{Y}$ be the one-hot label matrix, and Θ is obtained by minimizing the cross-entropy loss of the label:

$$L = -\sum_{v \in \mathcal{V}_L} \mathbf{Y} \ln \mathbf{Z}. \tag{4}$$

3.3 Node Feature Correction Module

We expand the output feature matrices of two GCNs to two times its dimensionality in the evaluating phase. Then, input the expended feature matrices into the MLP to obtain the pseudo labels. The pseudo-labels of node v obtained through the two GCNs are denoted as $l_F(v)$ and $l_T(v)$ respectively. MLPs in Node Feature Correction Module share parameters with that in training phase. The nodes with the same pseudo labels in the two encoders form the set $\mathcal{V}_s$, otherwise form the set $\mathcal{V}_i$.

For each node $v \in \mathcal{V}_i$, it is difficult to obtain the correct label directly using the original output because it gets a different pseudo-label in the two GCNs. Use $\hat{\boldsymbol{z}}$ to denote the corrected representation of $\boldsymbol{z}$. The purpose of output correction is to regenerate the output for node $v \in \mathcal{V}_i$. Note that there is no output correction for the node $u \in \mathcal{V}_s$, $\hat{\boldsymbol{z}}_u = \boldsymbol{z}_u$. As shown in Fig. 2, the embedding of the node $v \in \mathcal{V}_i$ is regenerate by aggregating the embeddings of the neighbouring nodes in $\mathcal{V}_s$.

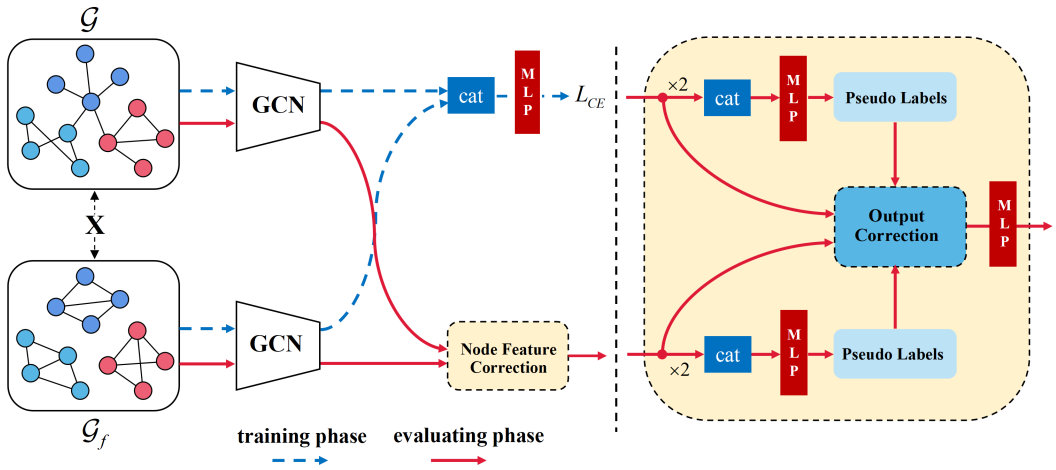

Figure 1: Schematic overview of OC-GCN, which inputs are the graphs $\mathcal{G}$ and $\mathcal{G}'$. The cat indicates concatenate operation. The diagram on the right shows the exact structure of the node feature correction module.

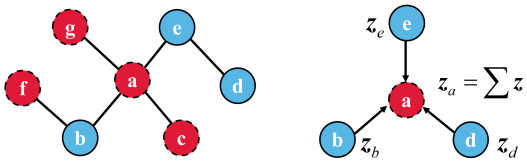

Figure 2: Regenerate the output for nodes in $\mathcal{V}_i$. The red nodes indicate nodes in $\mathcal{V}_i$ and the blue nodes indicate nodes in $\mathcal{V}_s$. As shown on the right, embedding of node a is regenerated by summing the embeddings of the nodes in $\mathcal{V}_s$, which also in its 2-hop neighbors.

$$\hat{z}_v = \sum_{u \in \mathcal{N}^m(v) \cap \mathcal{V}_s} z_u. \tag{5}$$

$\mathcal{N}^m(v)$ represents the set of nodes in the m-hop neighbors of node v (m was set to 2 in the experiments). The two GCNs construct neighbourhoods for output correction based on $\mathcal{G}_f$ and $\mathcal{G}$ respectively. Then, the output of the two views are contacted to obtain the final embeddings, $\hat{\mathbf{Z}} = \text{contact}\left(\hat{\mathbf{Z}}_F, \hat{\mathbf{Z}}_T\right)$. Finally, convert the embeddings $\hat{\mathbf{Z}}$ to predicted labels $\hat{\mathbf{Y}}$ through MLP. Algorithm 1 shows the overall algorithm.

Algorithm 1: Main steps of the OC-GCN algorithm

Input : Node feature matrix $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, adjacency matrix $\mathbf{A}$, node label matrix $\mathbf{Y}$, graph convolution networks f_{θ_1} and f_{θ_2}.

Output : Classification results $\mathbf{Y}'$.

Step 1: Generating Feature Graph

Calculate similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Build $\mathcal{G}_f$ based on the similarity of feature representations.

Step 2: Training phase

$\mathbf{Z}_F = f_{\theta_1}\left(\mathcal{G}_f\right), \mathbf{Z}_T = f_{\theta_2}\left(\mathcal{G}\right)$.

Concatenate $\mathbf{Z}_F$ and $\mathbf{Z}_T$ together, and input them into MLP to get the classification result.

Update all parameters of the model by minimizing the Equ. 4.

Step 3: Evaluating phase

Obtain pseudo labels in the two encoders.

Construct the sets $\mathcal{V}_s$ and $\mathcal{V}_i$.

if s **then**

  Output correction according to Equ. 5.

end

Convert the output embedding $\hat{\mathbf{Z}}$ to predicted label $\mathbf{Y}$.

4 EXPERIMENT

4.1 Datasets

we conduct extensive experiments on four widely used datasets including three citation networks (Cora, Citeseer, Pubmed) [12] and one copurchase network (Amazon Computers) [15]. Table 1 shows an overview of datasets.

Table 1: Details of the datasets used in the paper.

Dataset	Classes	Features	Nodes	Edges
Cora	7	1,433	2,708	5,429
Citeseer	6	3,703	3,327	4,732
Pubmed	3	500	19,717	44,338

4.2 Baseline and Setup

We compare the proposed method with previous supervised state-of-the-art methods. The comparison models include: Label Propagation (LP) [24], Graph Convolutional Network (GCN) [12], Graph

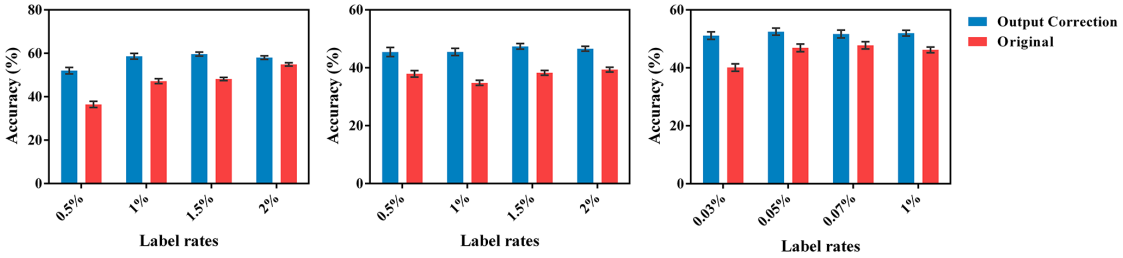

Figure 3: The the accuracy before and after output correction of the insistent nodes in two views.

Sample and Aggregate Network (SAGE) [8], Graph Attention Network (GAT) [18], and Higher-Order Graph Convolutional Architectures (MixHop) [1]. We also conduct ablation analysis. We use OC-GCN-w/o to denote the model without output correction for inconsistent node in two views.

For citation networks, we closely follow the evaluation protocol of [12]. For co-purchase network, nodes that are not training sets are split into 1:9, and each split is used for validation and test, respectively. In this paper, all models use a 2-layer GCN with the same hidden layer dimension (nhid1) and output dimension (nhid2) simultaneously, where $nhid1 \in \{512, 768\}$ and $nhid2 \in \{128, 256\}$. We use ReLU as the activation function, and set the learning rate ranging from 0.0001 to 0.0005. In addition, the dropout rate is 0.5, $weight\ decay \in \{5e-3, 5e-4\}$. We train the model for a fixed number of epochs, specifically 200, 200, 500 epochs for Cora, Citeseer, and Pubmed, respectively. All models are initialized with Glorot initialization [6], and trained using Adam optimizer on all datasets. When constructing the feature graph, k is set to be 6, 6, 3 for Cora, Citeseer and Pubmed, respectively.

4.3 Node Classification Results

In order to evaluate our model more comprehensively, we set up training sets with different label rates on datasets. For Cora and Citeseer: {0.5%, 1%, 1.5%, 2%}, for Pubmed: {0.03%, 0.05%, 0.07%, 0.1%}. The results of the semi-supervised node classification for the different models are summarised in Tables 2 to 4.

Table 2: Classification Accuracy on Cora.

Cora Dataset				
Label Rate	0.5%	1%	1.5%	2%
LP	57.6	61.2	62.4	63.4
GCN	54.2	61.0	66.2	72.8
SAGE	53.6	60.2	65.2	72.1
GAT	54.3	60.3	66.5	72.5
MixHop	55.2	61.1	65.8	72.8
OC-GCN	60.3	64.2	68.4	73.8
GAIN	2.7	3.0	1.9	1.0

We have the following observations: our proposed OC-GCN consistently outperforms LP, GCN, SAGE, GAT, and MixHop on all the datasets. It is evident that when labeled data is insufficient, the performance of the GCN suffers significantly. In some cases, GCN performs even worse than LP [24] when the training size is limited. Compared with GCN, OC-GCN achieves a maximum improvement of 6.1%, 4.5%, and 4.1% for accuracy on Cora, Citeseer, Pubmed and Am. Comp, respectively. Furthermore, OC-GCN achieves an improvement over OC-GCN-w/o, showing the effectiveness of regenerating the representations of inconsistent nodes in two views.

Table 3: Classification Accuracy on Citeseer.

Citeseer Dateset				
Label Rate	0.5%	1%	1.5%	2%
LP	39.6	43.2	45.5	48.2
GCN	46.6	56.3	59.8	64.8
SAGE	46.7	56.4	60.1	65.2
GAT	46.7	56.6	60.3	65.2
MixHop	48.2	58.4	61.8	65.8
OC-GCN	51.1	60.2	64.1	67.2
GAIN	2.7	3.0	2.3	1.0

Table 4: Classification Accuracy on Pubmed.

Pubmed Dateset				
Label Rate	0.03%	0.05%	0.07%	0.1%
LP	59.4	61.8	62.3	63.6
GCN	57.2	59.8	63.4	67.6
SAGE	57.5	60.1	63.9	67.6
GAT	57.8	60.4	64.5	67.8
MixHop	58.3	61.0	64.6	68.2
OC-GCN	61.3	62.9	66.3	69.5
GAIN	1.9	1.1	0.9	1.2

4.4 Effect of Node Feature Correction

In this section, we calculate the classification accuracy of inconsistent nodes in two views before and after the output correction. In the training phase, training epochs is taken as 100, 100, 200 for

Cora, Citeseer and Pubmed under different label rates to learn a label distribution for each node.

In Fig. 3, the accuracy before and after output correction of the inconsistent samples is drawn separately for Cora, Citeseer, and Pubmed under different label rates. Models that misclassify are often samples where classifiers predict different results in feature and topology views, Only about 40% accuracy on average. The classification accuracy of inconsistent samples after the correction improved significantly. The lower the label rate, the more significant the improvement in accuracy.

5 CONCLUSION

This paper proposes an output correction method across feature and topology views named OC-GCN. OC-GCN could select inconsistent samples in topology graph and feature graph, and correct their output based on consistent samples in the neighborhood. The consistency of the feature and structure views is increased by correcting for inconsistent samples. Experiments on several benchmarks demonstrate that output correction can significantly improve the classification accuracy of inconsistent nodes. Our method is superior to state-of-the-art semi-supervised learning for GCN at different label rates.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Hrayr Harutyunyan, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and A. G. Galstyan. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *ICML*.

[2] Joan Bruna, Wojciech Zaremba, Arthur D. Szlam, and Yann LeCun. 2013. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.

[3] Nicola De Cao and Thomas Kipf. 2018. MolGAN: An implicit generative model for small molecular graphs. *ArXiv* abs/1805.11973 (2018).

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*.

[5] Xiaolong Fan, Maoguo Gong, Yue Wu, and Hao Li. 2021. Maximizing Mutual Information Across Feature and Topology Views for Learning Graph Representations. *ArXiv* abs/2105.06715 (2021).

[6] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

[7] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. *ArXiv* abs/1706.04599 (2017).

[8] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[9] Zongbo Han, Changqing Zhang, H. Fu, and Joey Tianyi Zhou. 2021. Trusted Multi-View Classification. *ArXiv* abs/2102.02051 (2021).

[10] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. 2020. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993* (2020).

[11] Zhao Kang, Xiao Lu, Jian Liang, Kun Bai, and Zenglin Xu. 2020. Relation-Guided Representation Learning. *Neural networks : the official journal of the International Neural Network Society* 131 (2020), 93–102.

[12] Thomas Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv* abs/1609.02907 (2017).

[13] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR*.

[14] Changshu Liu, Liangjiang Wen, Zhao Kang, Guangchun Luo, and Ling Tian. 2021. Self-supervised Consensus Representation Learning for Attributed Graph. In *Proceedings of the 29th ACM International Conference on Multimedia*.

[15] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[16] Sankar K. Pal and Sushmita Mitra. 1992. Multilayer perceptron, fuzzy sets, and classification. *IEEE transactions on neural networks* 3 5 (1992), 683–97.

[17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. 2018. Graph Attention Networks. *ArXiv* abs/1710.10903 (2018).

[19] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. 2021. Be Confident! Towards Trustworthy Graph Neural Networks via Confidence Calibration. In *NeurIPS*.

[20] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[21] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *AAAI*.

[22] Jize Zhang, Bhavya Kailkhura, and Thomas Yong-Jin Han. 2020. Mix-n-Match: Ensemble and Compositional Methods for Uncertainty Calibration in Deep Learning. In *ICML*.

[23] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *UAI*.

[24] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *ICML*.