BAYESIAN NODE CLASSIFICATION FOR NOISY GRAPHS

Hakim Hafidi[⋆†] *Mounir Ghogho*[⋆] *Philippe Ciblat*[†] *Ananthram Swami*[‡]

[⋆] Université Internationale de Rabat, College of Engineering & Architecture, TICLab, Morocco
[†]LTCI, Telecom Paris, Institut Polytechnique de Paris, France
[‡]United States DEVCOM Army Research Laboratory, USA

ABSTRACT

Graph neural networks (GNN) have been recognized as powerful tools for learning representations in graph structured data. The key idea is to propagate and aggregate information along edges of the given graph. However, little work has been done to analyze the effect of noise on their performance. By conducting a number of simulations, we show that GNN are very sensitive to the graph noise. We propose a graph-assisted Bayesian node classifier which takes into account the degree of impurity of the graph, and show that it consistently outperforms GNN based classifiers on benchmark datasets, particularly when the degree of impurity is moderate to high.

Index Terms— Node classification, noisy graphs

1. INTRODUCTION

Graphs offer a flexible framework for representing diverse data. They are described by a set of nodes representing entities and a set of edges connecting them representing relationships between them. For attributed graphs, nodes and edges may further be associated with several features. To leverage the richness of information that resides in graphs, the scientific community developed a set of methods and techniques that can be grouped under the umbrella of graph representation learning. These methods have found applications in molecular physics [1], quantum chemistry [2], finance [3] and human brain activity and function [4].

Most successful approaches for graph representation learning are based on Graph Neural networks (GNN), a class of deep learning methods designed to perform inference on data described by graphs. A classical example is the Graph Convolutional Network (GCN) [5]. The GCN filter can be seen as an aggregation operator, i.e. the representation of a node is obtained by averaging its intrinsic features with those of its first-order neighbors using the symmetrically normalized adjacency matrix. GCN was designed for the task of graph-based semi-supervised learning which consists of classifying nodes in a graph where labels are only available for a small subset of nodes.

The node classification task has then become a standard problem for studying the performance of GNN and several approaches have been proposed to build more sophisticated and expressive GNN. As an example, the authors in [6, 7] introduced GAT and AGNN that use an attention mechanism to update the adjacency matrix by giving different weights to neighbors based on nodes' and edges' features. Other researchers explored higher-order information of the graph by repeatedly mixing feature representations of neighbors at various distances [8] or by modifying the propagation strategy of GCN by exploring its relation to the PageRank algorithm [9].

Although these approaches achieved remarkable results on a number of benchmark datasets, we notice that their performance varies significantly across datasets. More specifically, the gain in terms of prediction accuracy of GNN when compared to that of a simple logistic regression (i.e. no contribution from the neighbors) highly depends on the dataset. To the best of our knowledge, no attempt has been made in the literature to investigate the reasons behind the variations of this gain across benchmark datasets. We hypothesize that these variations can be explained by the connectivity and the *degree of purity* of the graph. Connectivity can be measured by the probability of intra-class connection, p (i.e. the probability that two nodes from the same class are connected). The degree of purity of the graph can be measured by $1-q/p$, where q is the probability of inter-class connection q (i.e. the probability that two nodes from different classes are connected). We will refer to the degree of impurity, q/p, as the noise-to-signal ratio, which is typically lower than one. Datasets where $q < p$ are referred to as assortive in stochastic block models, which are generative models for random graphs that tend to produce graphs containing communities [10]. From this perspective, one can say that node classification is easier in graphs with strong community structure.

Table 1 shows estimations of p and q/p for two widely used citation datasets for benchmarking GNN algorithms, as well as the node classification accuracies when using a logistic regression classifier with a two-layer GCN, which consists of aggregating the feature vectors of the first and second order neighbors of a node before applying logistic regression. The results clearly suggest that the classification performance gain of GNN algorithms decreases with decreasing p and/or increasing noise-to-signal ratio q/p, i.e. performance gain of GNN over logistic regression increases with tighter commu-

	Cora	Citeseer
Inter class connectivity p	23×10^{-3}	12×10^{-3}
Intra class connectivity q	5.5×10^{-3}	4.3×10^{-3}
Noise-to-Signal ratio q/p	0.23	0.36
Logistic regression	56.0%	57.2%
Two layer GCN [5]	81.5%	70.3%
Gain	+45.5%	+22.9%

Table 1. Estimations of intra and inter class connection probabilities on two datasets and the accuracy of classification

nity structure.

The main contribution of the paper is to propose a new node classification method robust to the impurity of the graph. The method relies on a Bayesian approach taking into account the graph structure. The paper is organized as follows: in Section 2, we formulate the node classification problem and recall some GNN-based classifiers. In Section 3, we introduce our new graph-assisted Bayesian classifier. We show it leads to a GNN-based classifier only in the absence of noise (i.e., $q = 0$) and under further conditions. In Section 4, numerical results are provided. Comparison with existing GNN-based methods are done on real datasets for which the *degree of purity* has been modified by injecting artificial noise.

2. BACKGROUND

2.1. Problem formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V}$ is a set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges. Each node $u \in \mathcal{V}$ is represented by a feature vector $\boldsymbol{x}_u \in \mathbb{R}^{F\times 1}$ where F is the number of node's features. An adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ represents the topological structure of the graph where $N = |\mathcal{V}|$ is the number of nodes in the graph. Without loss of generality we assume the graph to be unweighted i.e $A_{u,v} = 1$ if $(u, v) \in \mathcal{E}$ and $A_{u,v} = 0$ otherwise. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^\top$. Let y_i denote the label of the ith node and let K denote the number of classes, i.e. $y_i \in \{1, \ldots, K\}$. We assume that only a small number of labels, say M ($\ll N$), are known.

The objective of node classification is to predict the class of all unlabeled nodes in the graph given the adjacency matrix $\boldsymbol{A}$, the feature matrix $\boldsymbol{X}$ and the set of available labels.

2.2. Graph Neural Networks

GNNs are a class of graph embedding architectures which use the graph structure in addition to node and edge features to generate a representation vector (i.e., embedding) for each node. GNNs learn node representations by aggregating the features of neighboring nodes and edges. The output of the l-th layer of these GNNs is generally expressed as:

$$\boldsymbol{h}_u^{(l)} = \phi^{(l)}(\boldsymbol{h}_u^{(l-1)}, \psi^{(l)}(\{(\boldsymbol{h}_u^{(l-1)}, \boldsymbol{h}_v^{(l-1)}) : v \in \mathcal{N}(u)\})) \quad (1)$$

where $\boldsymbol{h}_u^{(l)}$ is the feature vector of node u at the l-th layer initialized by $\boldsymbol{h}_u^{(0)} = \boldsymbol{x}_u$ and $\mathcal{N}(u)$ is the set of first-order neighbors of node u. Different GNNs use different formulations of $\psi^{(l)}$ and $\phi^{(l)}$ [11]. Note that a first-order GNN based classifier relies on one layer or equivalently considers only the one-hop neighborhood in the graph.

Graph Convolutional Neural Network (GCN). The convolutional propagation rule used in GCN defines the functions $\psi^{(l)}$ and $\phi^{(l)}$ respectively such that:

$$\begin{aligned} \bar{\boldsymbol{h}}_u^{(l)} &= \frac{\boldsymbol{h}_u^{(l-1)}}{d_u + 1} + \sum_{v\in\mathcal{N}(u)} \frac{\boldsymbol{h}_v^{(l-1)}}{\sqrt{(d_u+1)(d_v+1)}} \\ \boldsymbol{h}_u^{(l)} &= \sigma((\boldsymbol{W}^{(l)})^\top \bar{\boldsymbol{h}}_u^{(l)}), \end{aligned} \quad (2)$$

where $\boldsymbol{W}^{(l)}$ is a learnable weight matrix, σ is a rectified linear unit (ReLU), and d_u is the degree of node u [5]. All nodes' aggregation operations are computed in parallel resulting in the following matrix representation:

$$\boldsymbol{H}^{(l)} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)}), \quad (3)$$

where $\boldsymbol{H}^{(l)} = [\boldsymbol{h}_1^{(l)}, \boldsymbol{h}_2^{(l)}, \ldots, \boldsymbol{h}_N^{(l)}]^\top$ is the matrix of nodes' hidden feature vectors at the $l-$th layer and $\hat{\boldsymbol{A}} = \check{\boldsymbol{D}}^{-\frac{1}{2}}\check{\boldsymbol{A}}\check{\boldsymbol{D}}^{-\frac{1}{2}}$ is the symmetrically normalized version of the adjacency matrix with added self-loop $\check{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ with $\check{\boldsymbol{D}}$ being its diagonal degree matrix, i.e. $\check{\boldsymbol{D}}_{ii} = \sum_j \check{A}_{ij}$. These operations are applied in each layer resulting in a classifier whose last layer is given by:

$$\hat{\boldsymbol{Y}} = \text{Softmax}(\hat{\boldsymbol{A}}\boldsymbol{H}^{(L-1)}\boldsymbol{W}^{(L)}), \quad (4)$$

where $\hat{\boldsymbol{Y}} \in \mathbb{R}^{N\times K}$ and $\hat{y}_{u,k}$ is the probability that node u belongs to class k. The node is assigned to the class with the highest probability.

Graph Convolution Operator (GraphConv). In [12], the GraphConv is defined through the functions $\psi^{(l)}$ and $\phi^{(l)}$ such that:

$$\boldsymbol{h}_u^{(l)} = (\boldsymbol{W}_1^{(l)})^\top \boldsymbol{h}_u^{(l-1)} + (\boldsymbol{W}_2^{(l)})^\top (\sum_{v\in\mathcal{N}(u)} \boldsymbol{h}_v^{(l-1)}). \quad (5)$$

Unlike GCN, the GraphConv operator computes a transformation matrix of the central node that is different from the transformation of its neighbors.

Graph Isomorphism Network (GIN). In [13], the GIN is defined through the functions $\psi^{(l)}$ and $\phi^{(l)}$ such that:

$$\boldsymbol{h}_u^{(l)} = (\boldsymbol{W}^{(l)})^\top ((1+\alpha)\boldsymbol{h}_u^{(l-1)} + \sum_{v\in\mathcal{N}(u)} \boldsymbol{h}_v^{(l-1)}). \quad (6)$$

The GIN operator attributes a different learnable weight to the central node when combining information from its neighbors.

3. BAYESIAN NODE CLASSIFIER

3.1. Model definition

Unlike in GNN, we adopt a Bayesian approach to node classification. The *exact* Bayesian classifier for each node would depend on all nodes connected directly or indirectly to the node, in the sense that the probability that a node belongs to one of the classes depends on its intrinsic features, features of a large number of other nodes and the graph structure. Here, we introduce a simplified Bayesian classifier which classifies a node based only on its intrinsic features and those of its first-order neighbors (i.e., one-hop neighborhood), considering only the links between the node and its neighbors. By analogy with the conventional naive Bayesian classifier (which ignores the correlations between the feature), our classifier can also be considered as a naive graph-based node classifier in the sense that it ignores some graph information for the sake of simplicity. We leave higher-order (less naive) versions of this classifier for future work. We name the proposed classifier as first-order graph-assisted Bayesian (FO-GAB) classifier. Next, we derive this classifier and show that it may reduce to a first-order GNN-based classifier only when the graph is noise-free.

Let $\mathcal{X}_u = \{\boldsymbol{x}_u\} \cup \{\boldsymbol{x}_v, v \in \mathcal{N}(u)\}$ the set of features of node u and its first order neighbors, D_k the probability distribution that generates samples belonging to class k. The objective is to compute the probability that a node u belongs to class k knowing $\mathcal{X}_u$, which, using Bayes rule, expresses as:

$$P(y_u = k|\mathcal{X}_i) = \frac{P(\mathcal{X}_u|y_u = k)P(y_u = k)}{\sum_{k'=1}^{K} P(\mathcal{X}_u|y_u = k')P(y_u = k')}. \quad (7)$$

Let $\boldsymbol{T} \in \mathbb{R}^{K\times K}$ be the link probabilities matrix whose (k, k')-th entry, $t_{k,k'}$ denotes the probability that there is a link between nodes of class k and those of class k'. The probabilities of intra-class and inter-class connectivity are defined as the average of the diagonal elements and off-diagonal elements of $\boldsymbol{T}$, respectively, i.e., $p = \frac{1}{K}tr(\boldsymbol{T})$ and $q = \frac{1}{K(K-1)}(sum(\boldsymbol{T}) - tr(\boldsymbol{T}))$ with $sum(\cdot)$ the sum of all matrix elements.

The distribution of each neighboring node knowing that the central node belongs to class k is a *mixture* of the distributions associated with the different classes, where the weights of the mixture are determined by the link probabilities:

$$P(\boldsymbol{x}_v|y_u = k) = \sum_{k'=1}^{K} \tilde{t}_{k,k'} D_{k'}(\boldsymbol{x}_v), \quad (8)$$

where $\tilde{t}_{k,k'} = \frac{t_{k,k'}}{\sum_{k'=1}^{K} t_{k,k'}}$. The $t_{k,k'}$'s are normalised to make the sum of the weights of the mixture to be equal to one. Assuming that the feature vectors of the different nodes are independent, the likelihood of $\mathcal{X}_u$ given $y_u = k$ is:

$$P(\mathcal{X}_u|y_u = k) = D_k(\boldsymbol{x}_u) \prod_{v\in\mathcal{N}(u)} (\sum_{k'=1}^{K} \tilde{t}_{k,k'} D_{k'}(\boldsymbol{x}_v)). \quad (9)$$

Let $P(y_u = k) = \pi_k$, the prior probability of class k. Hence, the probability that node u belongs to class k knowing $\mathcal{X}_u$ is:

$$P(y_u = k|\mathcal{X}_u) = \frac{\pi_k D_k(\boldsymbol{x}_u) \prod_{v\in\mathcal{N}(u)} (\sum_{k'=1}^{K} t_{k,k'} D_{k'}(\boldsymbol{x}_v))}{\sum_{k'=1}^{K} \pi_{k'} D_{k'}(\boldsymbol{x}_u) \prod_{v\in\mathcal{N}(u)} (\sum_{k''=1}^{K} t_{k',k''} D_{k''}(\boldsymbol{x}_v))}. \quad (10)$$

The first-order GAB classifier consists thus of assigning to node u the class maximizing the above probability, i.e.,

$$\hat{y}_u = \arg\max_k P(y_u = k|\mathcal{X}_u). \quad (11)$$

3.2. Relation to a GNN based classifier

Here, we study the relation between the first-order GAB classifier and first-order GNN-based classifier in the case of binary classification (i.e. $K = 2$). By defining $\tilde{p} = \frac{1}{2}(\tilde{t}_{1,1} + \tilde{t}_{2,2})$ and $\tilde{q} = \frac{1}{2}(\tilde{t}_{1,2} + \tilde{t}_{2,1})$, Eq. (7) can be written as:

$$P(\mathcal{X}_u|y_u = k) = D_k(\boldsymbol{x}_u) \prod_{v\in\mathcal{N}(u)} \frac{\tilde{p} D_k(\boldsymbol{x}_v) + \tilde{q} D_{k'}(\boldsymbol{x}_v)}{\tilde{p} + \tilde{q}}, \quad (12)$$

which leads to:

$$P(y_u = 1|\mathcal{X}_i) = \frac{1}{1 + \frac{\pi_2}{\pi_1}\frac{D_2(\boldsymbol{x}_u)}{D_1(\boldsymbol{x}_u)} \prod_{v\in\mathcal{N}(u)} \frac{\frac{D_2(\boldsymbol{x}_v)}{D_1(\boldsymbol{x}_v)} + \frac{\tilde{q}}{\tilde{p}}}{1 + \frac{\tilde{q}}{\tilde{p}}\frac{D_2(\boldsymbol{x}_v)}{D_1(\boldsymbol{x}_v)}}}. \quad (13)$$

When $q = 0$ (or equivalently $\tilde{q} = 0$) , the above expression reduces to:

$$P(y_u = 1|\mathcal{X}_i) = \frac{1}{1 + \frac{\pi_2}{\pi_1} \prod_{v\in\mathcal{N}(u)\cup\{u\}} \frac{D_2(\boldsymbol{x}_v)}{D_1(\boldsymbol{x}_v)}} \quad (14)$$

If, in addition, $\boldsymbol{x}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$, the first-order GAB classifier is found to be:

$$\beta_0 + \boldsymbol{\beta}_1^\top \left(\sum_{v\in\mathcal{N}(u)\cup\{u\}} \boldsymbol{x}_v \right) > 0 \Rightarrow \hat{y}_u = 1, \quad (15)$$

where

$$\begin{aligned} \beta_0 &= \log(\frac{\pi_2}{\pi_1}) + (|\mathcal{N}(u)| + 1)(\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1) \\ \boldsymbol{\beta}_1^\top &:= [\beta_{1,1}, \cdots, \beta_{1,F}] = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}. \end{aligned} \quad (16)$$

Therefore, the test statistic of the 1st-order GAB classifier in the case of Gaussian distributions with the same covariance

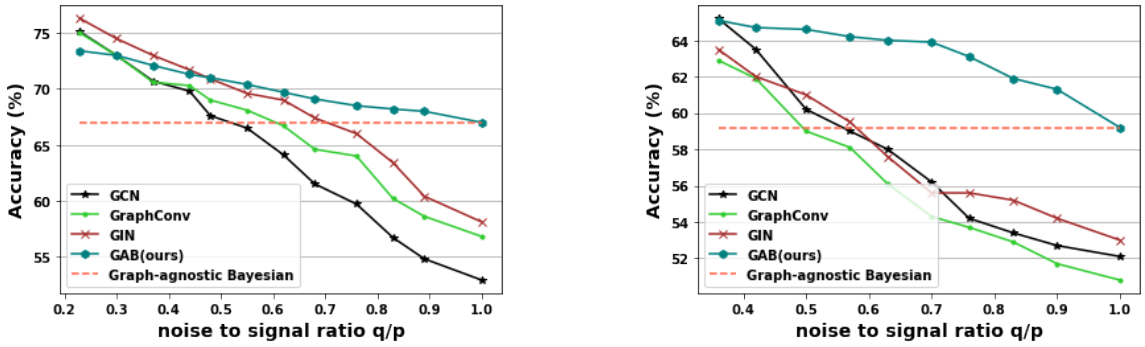

Fig. 1. Accuracy performance with added noisy edges in Cora (left) and Citeseer (right).

matrix across the classes, is based on averaging the node's feature vector and those of its neighbors (used in GNN classifiers) *only* when $q = 0$, i.e. when the graph is noise-free.

The averaging over the neighbors test statistic is also valid when the distribution is not normal. When the features $x_{u,f}$ are independent binary random probabilities $P(x_{u,f} = 1|y_u = 2) = \alpha_f^{(2)}$ and $P(x_{u,f} = 1|y_u = 1) = \alpha_f^{(1)}$, the first-order GAB classifier in the $q = 0$ case has the same form as in Eq. (15) but where the coefficients are given by:

$$\begin{aligned} \beta_0 &= \log \frac{\pi_2}{\pi_1} + F \sum_{f=1}^{F} \log \frac{1-\alpha_f^{(2)}}{1-\alpha_f^{(1)}} \\ \beta_{1,f} &= -\log \frac{\alpha_f^{(1)}}{\alpha_f^{(2)}} \frac{1-\alpha_f^{(2)}}{1-\alpha_f^{(1)}}. \end{aligned} \tag{17}$$

Hence, the test statistic is again based on averaging over the neighbors, as in GNN-based classifiers, *only* when $q = 0$.

4. EXPERIMENT

We empirically compare the robustness to noise of the proposed first-order graph-assisted Bayesian classifier (GAB) with that of first-order GNNs (GCN, GraphConv, GIN) and a graph-agnostic Bayesian classifier which classifies the nodes based only on their intrinsic features. The experiment is as follows, starting from a given graph, we add links between previously unconnected nodes that belong to different classes with the goal of gradually varying the noise-to-signal ratio from its initial value to 1. For each corrupted graph, we train the above mentioned classifiers and report their accuracy on Figure 1.

Datasets. We use Cora and Citeseer, two citation networks where nodes are bag of words representations of documents and edges correspond to (undirected) citations. Each document is assigned a unique label based on its topic. Statistics of the datasets are given in Table 2.

Experimental design. We follow the standard train/val/test splits in [5]. To implement the GNN based classifiers we use Pytorch. We initialize all models using Glorot initialization and trained them to minimize the cross entropy loss using the Adam optimizer with an initial learning rate of 0.005. To implement our graph based Bayesian classifier (Eq. (10)), we consider a Bernoulli distribution for each component of the feature vectors. We estimate the parameters of Bernoulli distributions and the link probabilities from the training set. It is worth point out that due to the scarcity of the available labelled nodes, we opt for the following estimates of the normalized link probabilities. We first estimate the normalized link probability matrix denoted by $\tilde{\boldsymbol{T}}$. We then force its diagonal terms to be identical and equal to $\check{p} = \frac{1}{K} tr(\tilde{\boldsymbol{T}})$ and its off-diagonal to be identical and equal to $\check{q} = \frac{1}{K-1}(1-\check{p})$.

Table 2. Description of the datasets

Dataset	Nodes	Edges	Features	Classes (K)	Train/Val/Test Nodes
Cora	2,708	5,429	1,433	7	140/500/1,000
Citeseer	3,327	4,732	3,707	6	120/500/1,000

Experimental results. Figure 1 shows that GAB exhibits a strong learning ability that is robust to the degree of impurity of the graph. When the graph has maximum impurity with $q/p = 1$, it ignores the neighbors and thus reduces to a the conventional graph-agnostic Bayesian classifier. The comparison demonstrates that GAB significantly outperforms the GNN-based classifiers, particularly when the degree of impurity is moderate to high, i.e. when the network does not show strong community structure.

5. CONCLUSION

We have studied the effects of the degree of purity of graphs on the performance of GNN-based node classification. Further, we have proposed a graph-assisted Bayesian node classifier, which takes into account the degree of impurity of the graph. The proposed classifier is shown to significantly outperform the GNN-based classifiers, particularly when the degree of impurity is moderate to high. As future work, we will investigate the performance of higher-order GAB.

6. REFERENCES

[1] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia, "Graph networks as learnable physics engines for inference and control," *arXiv preprint arXiv:1806.01242*, 2018.

[2] Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller, "Schnet–a deep learning architecture for molecules and materials," *The Journal of Chemical Physics*, vol. 148, no. 24, pp. 241722, 2018.

[3] Anish Khazane, Jonathan Rider, Max Serpe, Antonia Gogoglou, Keegan Hines, C Bayan Bruss, and Richard Serpe, "Deeptrax: Embedding graphs of financial transactions," in *IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 126–133.

[4] Yanlin Li, Rasoul Shafipour, Gonzalo Mateos, and Zhengwu. Zhang, "Supervised graph representation learning for modeling the relationship between structural and functional brain connectivity," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 9065–9069.

[5] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[6] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[7] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018.

[8] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," *arXiv preprint arXiv:1905.00067*, 2019.

[9] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.

[10] Albert-László Barabási, "Network science," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, 2013.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[12] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 4602–4609.

[13] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, "How powerful are graph neural networks?," *arXiv preprint arXiv:1810.00826*, 2018.