Node classification with graph neural network based centrality measures and feature selection

Asmaa M. Mahmoud, Abeer S. Desuky, Heba F. Eid, Hoda A. Ali
Department of Mathematics, Faculty of Science, Al-Azhar University, Cairo, Egypt

Article Info

Article history:

Received May 22, 2022
Revised Sep 27, 2022
Accepted Oct 20, 2022

Keywords:

Centrality measures
Feature selection
Graph neural network
Node classification
Normalization

ABSTRACT

Graph neural networks (GNNs) are a new topic of research in data science where data structure graphs are used as important components for developing and training neural networks. GNN always learns the weight importance of the neighbor for perform message aggregation in which the feature vectors of all neighbors are aggregated without considering whether the features are useful or not. Using such more informative features positively affect the performance of the GNN model. So, in this paper i) after selecting a subset of features to define important node features, we present new graph features' explanation methods based on graph centrality measures to capture rich information and determine the most important node in a network. Through our experiments, we find that selecting certain subsets of these features and adding other features based on centrality measure can lead to better performance across a variety of datasets and ii) we introduce a major design strategy for graph neural networks. Specifically, we suggest using batch renormalization as normalization over GNN layers. Combining these techniques, representing features based on centrality measures that passed to multilayer perceptron (MLP) layer which is then passed to adjusted GNN layer, the proposed model achieves greater accuracy than modern GNN models.

Corresponding Author:

Asmaa M. Mahmoud
Department of Mathematics, Faculty of Science, Al-Azhar University
Cairo, Egypt
Email: asmaadaoud.1959@azhar.edu.eg

1. INTRODUCTION

The graph convolutional network (GCN) is a new type of neural network. It develops a new representation for a node by aggregating feature vectors from all neighbors in the aggregation process, regardless of whether the features are effective. By picking a fixed size group of neighbors or assigning various weights to different neighbors in the aggregation process, recent methods have improved solutions to the above problem [1].

In recent years, the field of graph neural network (GNN) research has made significant development. Notably, a growing variety of GNN designs have been developed, including GCN [2], GraphSAGE [3], and graph attention networks (GAT) [4]. These designs are then used in a variety of applications, including social networks [5], [6], chemistry [7], [8], and biology [9], [10]. Node classification [2], [4], link prediction [11], [12], graph classification [13], [14], prediction of chemical characteristics, node ranking, and natural language processing are just a few of the challenges that GNNs have been used to solve. In this paper, we use graph neural networks to solve the node classification problem. Since the success of early GNN models like GCN [2] researchers have suggested lots of new modifications [15] to solve the model's numerous training issues and increase prediction capabilities. Neighbor sampling [3], [16], attention mechanism [4], usage of personalized PageRank matrix instead of the adjacency matrix [17], leveraging proximity in feature space [18], and simplified model design [19] are some of the strategies employed in these variations. In addition, there has been a growing tendency to make the models deeper by stacking additional layers to enhance the model's expressiveness [20], [21]. In general, GNN models use a weight matrix to combine the aggregation and transformation of features in the same layer, which is referred to as the graphical convolutional layer. As a learning framework of the graph data these layers are stacked together with the regularization (for example, Dropout) and non-linear transformation (for example, rectified linear units (ReLU)) [22].

By stacking the layer, power of the adjacency matrix is introduced, which aids in the generation of a new set of features for a node by aggregating neighbor's features over numerous hops. The number of these distinct features is dependent by the model's propagation stages or depth. The final node embeddings are either the result of only stacked layers or, in certain cases, the result of combining skip and residual connections at the final layer [22].

GNNs have become an essential tool for learning graph-centric data. Many prediction applications, such as node and graph classification, link prediction and so on [2]. Established a simple end-to-end training framework based on spectral graph convolution approximations. Since then, the research community has worked to enhance the performance of GNNs, and a number of strategies have been presented. Earlier GNN frameworks used a fixed edge propagation strategy, which was not necessarily scalable for bigger graphs. In graph neural networks, GraphSAGE [3] and fast graph convolutional networks (FastGCN) [16] propose neighbor sampling algorithms. The attention technique is used in GAT [4] to assign weights to features that gathered from neighbors. The feature propagation technique within layers of the model is improved by approximate personalized propagation of neural predictions (APPNP). Gasteiger *et al.* [17], similarity preserving graph convolutional networks (SimP-GCN) [18], and counter propagation neural network (CPGNN) [23], Geom-GCN [24] and jumping knowledge (JK) [25]. Recently, researchers have proposed that GNN models be made deeper [21]. DropEdge [20] suggests that a specific number of edges be dropped to lower the speed of oversmoothing convergence and relieve information loss. To allow deeper networks, graph convolutional network via initial residual and identity mapping (GCNII) [21] uses identity mapping and residual connections in GNN layers. To train deep GNN models, RevGNN [26] employs deep reversible architectures and uses noise regularization [27]. Traditional GNNs perform well in homophily networks, but they do not generalize to heterophily graphs, according to the researchers. Similar to our work, Zhu *et al.* [23] introduced the feature selection and feature extraction method for the GNNs. Chi *et al.* [28] proposes two unique techniques, the GCN res framework and embedding usage, to increase the test accuracy of the baseline in various datasets by leveraging residual networks and pre-trained embedding.

The curse of high dimensionality is a prevalent difficulty in machine learning. The quantity of training data needed might expand exponentially in parallel with the data's dimension. As a result, one of the key study issues in data science is feature selection, which tries to minimize high dimensionality by discovering and choosing the subset of the most important features in a dataset. The following are some of the downstream benefits of feature selection: i) faster data mining algorithms using the selected features, ii) improved accuracy of the trained model, iii) more interpretable model, and iv) alleviation of the overfitting problem by removing irrelevant or redundant features [29].

According to Duong *et al.* [30], GNN work well if there is a strong correlation between node labels and node features. So, in our work we proposed new features based on centrality measures then we choose features that strongly correlate with node labels. The primary benefit of this work is improving accuracy of node classification problem by combining:

- Centrality measures: Features calculated based on the central metrics used to capture rich information about the graph and give each node value that represents the importance of this node in this graph and this help in improving the solution of the problem of node classification;
- Selecting features: Choosing the subset of the most important features in a dataset using chi-square minimize high dimensionality and reduce used memory size, which lead to a faster model;
- GNN architecture: We suggest a simple GNN architecture; specifically using batch renormalization that helps in stabilizing the learning process in GNN.

2. METHOD

In this section, the comprehensive theoretical basis and the proposed method are discussed in appropriate subheadings. Consider $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ is an undirected graph with nodes V and edges E. Adjacency matrix is used to describe a graph and denoted by $\mathrm{A} \in \{0, 1\}$ with each element $\mathrm{A}_{ij} = 1$ if there exists an edge between node v_i and v_j, otherwise $A_{ij} = 0$. Diagonal degree matrix of A is denoted as D. Each node has a d-dimensional feature vector attached to it, and the feature matrix for all nodes is shown as $\mathrm{X} \in \mathrm{R}^{n \times d}$ [22].

2.1. Graph convolution neural network

GCN is multi-layer feed forward neural networks. It collects a node's neighborhood information, then transform it nonlinearly with a trainable weight matrix to get the nodes' final embeddings [31]. The feature propagation rule of GCN is

$$X^{(k+1)} = \sigma\,(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X^{(k)}W^{(k)}) \quad (1)$$

where W(k) is trainable weight matrix in the k-th layer, $\sigma\,(.)$ is an activation function such as ReLU, and $X^{(k)} = [x_1^{(k)}, \ldots\ldots, x_n^{(k)}]$ are the k-th layer node representations with $X^{(0)} = X$. However, because features are cumulatively aggregated, or combined with those of neighbors, this formulation is suitable for homophily datasets. The signal corresponding to the label is strengthened and prediction accuracy is increased by the cumulative aggregation of a node's self-features with those of its neighbors. $D^{-1}A$ is used as the normalized adjacency matrix instead of the symmetric one $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ [22], [32]. Therefore, the feature propagation scheme of GCN in layer *k* is:

$$X^{(k+1)} = \sigma\,(D^{-1}AX^{(k)}W^{(k)}) \quad (2)$$

2.2. Chi-square feature selection

Feature selection is a technique for identifying the features in the data that have the most effect on the prediction variable or output. The chi-square feature selection method is a common feature selection method. It has two variables, one for the frequency of occurrence of feature t and the other for the probability of occurrence of category C. When classifying nodes, we look to see if the features t and C are independent. The feature t cannot be used to identify whether a node belongs to category C if they're independent. In practise, determining the extent to which t and C are associated when they are non-independent is challenging. As a result, the chi-square test is applied to determine their relevance [33]. The (3) shows how to calculate the chi-square score of features t for category c.

$$X^2(D,t,c) = \sum_{e_t\in\{0,1\}}\sum_{e_c\in\{0,1\}}\frac{\left(N_{e_te_c} - E_{e_te_c}\right)^2}{E_{e_te_c}} \quad (3)$$

where N is the observed frequency, E is the expected frequency, e_t takes the value 1 if the data has feature t and 0 otherwise and e_c takes the value 1 if the data belong to class c and 0 otherwise. According to the first-order degree of freedom chi-square distribution, the higher the chi-square score for category c, the more category information the feature t contains, and the stronger the relationship between t and c.

2.3. Centrality measures

According to Cui *et al.* [34], there is a various way to construct artificial features so in our model graph neural network using feature selection based centrality measures (GNNFC), we'll calculate several features based on centrality measures. Centrality is a key concept in graph analytics for finding important nodes in a graph. It is used to determine the relative significance of nodes in a network. Depending on how "importance" is defined, each node may now be significant from a different perspective [35]. Centrality is measured using many metrics, each of which describes the importance of a node from a different perspective and gives essential analytical information about the graph and its nodes [35].

2.3.1. Degree centrality

The degree of a node in a non-directed network is defined as the number of direct connections it has with other nodes. Now, the Degree Centrality metric measures the significance of a node in a network based on its degree, i.e., the greater a node's degree, the more significant it is in a graph [36]. Mathematically, degree centrality is defined as (4):

$$D(v) = \sum_u m(v,u) \quad (4)$$

where $m(v,u) = 1$ if there is a link from node v to node u.

2.3.2. Closeness centrality

Let's define the "Geodesic distance" between two nodes in a graph to better understand this measure. The Geodesic distance d between these two nodes is defined as the number of edges between them on the shortest path between them. Mathematically, Geodesic distance can be defined as (5).

$$\mathrm{d(u,v)} = \begin{cases} \text{no. of shortest path from u to v} \\ 0, \text{if u = v} \\ \infty, \text{if there is no paths from u to v} \end{cases} \tag{5}$$

Furthermore, the closeness centrality metric measures a node's relevance in a network by how near it is to all other nodes in the graph. For a given node, it is defined as the total of the geodesic distances between a node and all other nodes in the network [36]. Mathematically, Closeness centrality $\mathrm{C(v)}$ can be defined as (6).

$$\mathrm{C(v)} = \frac{1}{\sum_{\mathrm{u}} \mathrm{d(v,u)}} \tag{6}$$

2.3.3. Betweenness centrality

The betweenness centrality captures how much a given node u is in-between others. The number of shortest paths (between any two nodes in the graphs) that pass through the target node u over the total number of shortest paths that exist between any two nodes in the network [35]. Betweenness Centrality of node v may be described mathematically as (7):

$$\mathrm{g(v)} = \sum_{\mathrm{s \neq v \neq t}} \frac{\sigma_{\mathrm{st}}(\mathrm{v})}{\sigma_{\mathrm{st}}} \tag{7}$$

where $\sigma_{st}(v)$ is the number of shortest paths that pass-through v, σ_{st} is the total number of shortest paths from node s to node t.

2.3.4. Eigenvector centrality

The topological position of nodes in the network is taken into consideration in all previous centrality measurements, but not the relevance of the nodes themselves. Eigenvector centrality calculates a node's centrality based on its neighbors' centrality. In comparison to a node connected to less significant nodes, a node connected to highly important nodes would have a higher eigenvector centrality score [36]. Eigenvector centrality may be described mathematically as follows.

The relative centrality score of vertex v can be defined as (8).

$$\mathrm{x_v} = \frac{1}{\lambda} \sum_{\mathrm{t \in M(v)}} \mathrm{x_t} = \frac{1}{\lambda} \sum_{\mathrm{t \in G}} \mathrm{a_{v,t} x_t} \tag{8}$$

where $M(v)$ is a set of the neighbors of v and λ is a constant. This may be represented in vector notation as the eigenvector equation.

$$\mathrm{Ax} = \lambda \mathrm{x} \tag{9}$$

2.4. Batch renormalization

Batch normalization is quite successful in speeding up and improving deep model training. When the training minibatches are small or do not contain independent samples, however, its efficacy is reduced [37]. Ioffe [38] introduces batch renormalization (BR), which solves the problem with small batches that BN has. When the batch size is small, BR introduces two more parameters that restrict the estimated mean and variance of BN within a given range, decreasing their drift.

The batch statistics (μ_B & σ_B) are used by BN. When the batch size is small, BR introduces two new parameters, r and d, with the goal of constraining the mean and std of BN and lowering the extreme difference. Normalization should ideally be done with the instance statistic:

$$\hat{\mathrm{x}} = \frac{\mathrm{x} - \mu}{\sigma} \tag{10}$$

By choosing $\mathrm{r} = \frac{\sigma\mathrm{B}}{\sigma}$ and $\mathrm{d} = \frac{\mu\mathrm{B} - \mu}{\sigma}$:

$$\hat{\mathrm{x}} = \frac{\mathrm{x} - \mu}{\sigma} = \frac{\mathrm{x} - \mu\mathrm{B}}{\sigma\mathrm{B}} . \mathrm{r} + \mathrm{d} \tag{11}$$

The authors recommend that the maximum absolute values of r and d be restricted. To begin, set the boundaries to 1 and 0, as if they were BN, and then progressively loosen them.

2.5. Proposed method

With an increase in the number of hops, the number of input feature combinations grows exponentially, making it computationally costly. In addition to feature selection, the model's prediction capacity must be improved. As a result, a GNN model may be created using feature selection strategy. Starting with all features as input, GNNFC learns to identify significant features while reducing the impact of unimportant features.

In GNNFC we calculate the Chi-square between each feature and the target and choose the features with the best Chi-square scores. We specify additional features with regard to graph centrality measurements such as betweenness and closeness since graph centralities have been used to characterize various properties of graphs. After calculating these features, we merge them with the selected features obtained from input features, and then input the new feature matrix for GNN as shown in Figure 1.

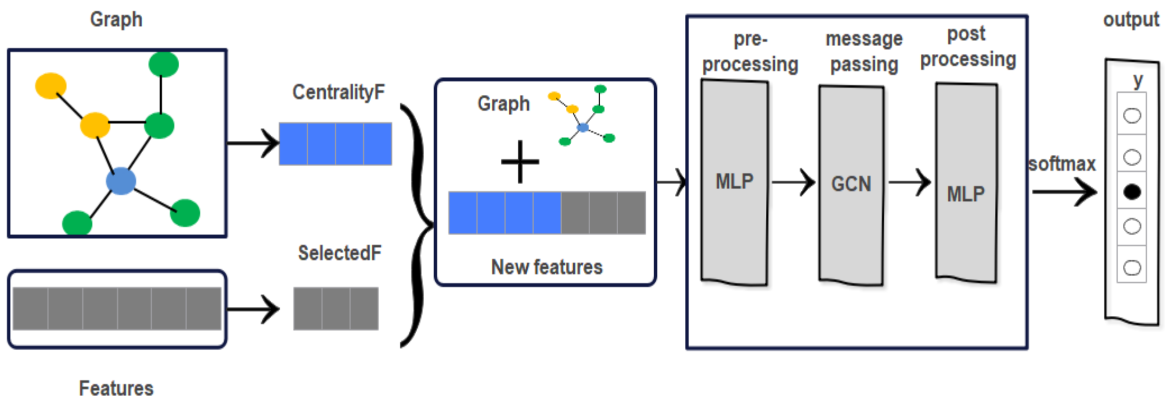


Figure 1. An illustration of the proposed method GNNFC

The merge process can be written as (12).

$$X = \mathrm{Concat}(\mathrm{selectedF}, \mathrm{CentralityF}) \quad (12)$$

where *selectedF* is the selected features based on Chi-square method and *CentralityF* is the features that calculated based on degree, closeness, betweenness and eigenvector centrality measures.

Following that, we suggest a generic design space for GNNs that consists of four steps:

- A preprocessing layer that uses multilayer perceptron (MLP) to generate initial node representations.
- A message passing layer using GCN.
- A post-processing layer that uses MLP to generate final node embeddings.
- To predict the node class, feed the node embeddings into a SoftMax layer.

In the first step the initial node representation is processed $h_v^0 = X_0$ using a MLP to produce a message. Then in a message passing step a node's representation is iteratively updated by aggregating its neighbors' representations. A node's representation captures the structural information inside its $k-\mathrm{hop}$ network neighborhood after k iterations of aggregation.

$$h_v^{(k+1)} = \mathrm{AGG}\left(\left\{\mathrm{ACT}\left(\mathrm{BR}\left(W^{(k)}h_u^{(k)} + b^{(k)}\right)\right), u \in N(v)\right\}\right) \quad (13)$$

where $h_u^{(k)}$ is the k-th layer embedding of node, $W^{(k)}, b^{(k)}$ are trainable weights, and $N(v)$ is the local neighborhood of v.

The adopted GNN in (13), contains a linear layer, followed by a series of modules: batch renormalization BR; nonlinear activation function ACT (.), where ReLU is considered, aggregation function AGG (.) where SUM is considered.

In (13), BR is used to normalize node features during training. According to Hamilton [39], normalization is most useful in jobs where node feature information is significantly more important than structural information, or where there is a large range of node degrees. In GNNs, the BR is especially good in stabilizing the value, especially for deep GNNs. As a result, a growing number of GNNs are beginning to use normalization. With a deep GNN models, the result tends to get bad performance. One solution is to use the skip connections to concatenate the inputs and outputs of GNN; here, we look at dense connections called SKIP-CAT, which concatenate embeddings from all previous layers.

3. RESULTS AND DISCUSSION

In this section, the detailed analysis of our proposed model is discussed. Also, the findings are reported extensively. The complete result analysis has been discussed in appropriate subheadings. To test and check the efficiency of our experiment on GNNFC, two sets of citation network benchmark datasets are used, Cora and Citeseer which are standard citation networks with the same training/validation/testing division [40], as summarized in Table 1.

Table 1. Information about the datasets used in our experiment

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,372	3,703	6

3.1. Performance of GNNFC versus traditional GNN models

In our experiments, two suggested new tricks are used with other tricks that exist. Also, the effects of different collection strategies in different datasets are discussed. Kong *et al.* [41] refer to whether or not suggested tricks are effective depends on the distribution of the data. The expressive abilities of tricks will be significantly influenced by different graph structures. Therefore, there is a need to use different data sets for tricks and their collections.

Through studies via these datasets, we discover that GNN with BR in the citation network Cora and Citeseer successfully relieve sensitive to node degrees issue of aggregation operation, which may also be used with other tricks because to its flexibility and strong generality. In addition, normalization is usually used to stabilize the gradient of GNN. In our second trick the performance of feature selection algorithm is examined with respect to graph centrality measures that characterize diverse graphs and discovered that it may make GNN have a clear improvement in most situations, so that using these tricks in GNN is strongly recommended. For these datasets, combination of two proposed design tricks lead to better average accuracy as shown in Table 2. The combination of previous tricks can improve the performance of GNNFC on our datasets as shown in Table 3.

Table 2. Comparison of the outcomes of node classification of GNNFC and other popular GNN models in terms of Accuracy. Results for these GNN models are taken from [22], [42]

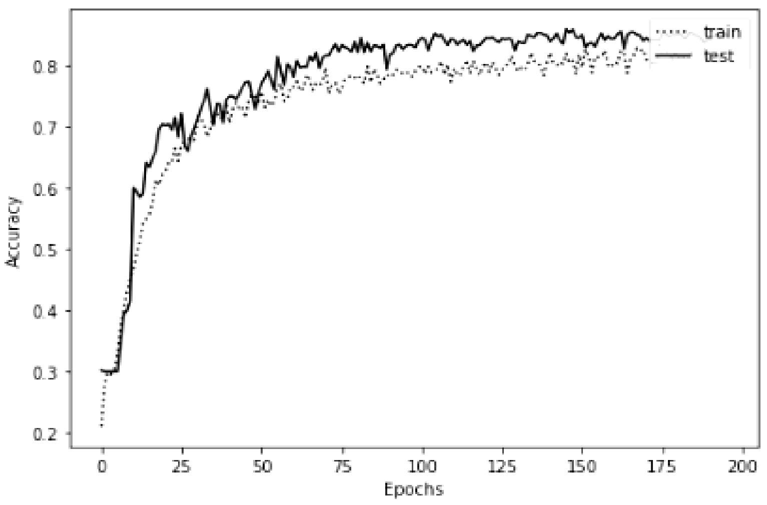
	Cora	Citeseer
GCN (2017)	87.28±1.26	76.68±1.64
GraphSAGE (2017)	86.90±1.04	76.04±1.30
GAT (2018)	82.68±1.80	75.46±1.72
FSGCN (2019)	83.50	73.00
GEOM-GCN (2020)	85.27	77.99
WRGAT (2021)	88.20±2.26	76.81±1.89
GPRGNN (2021)	88.49±0.95	77.08±1.63
GNNFC (Our method)	89.5±1.90	80.10±1.5

Table 3. The outcomes of node classification on Cora and Citeseer datasets in terms of AUC

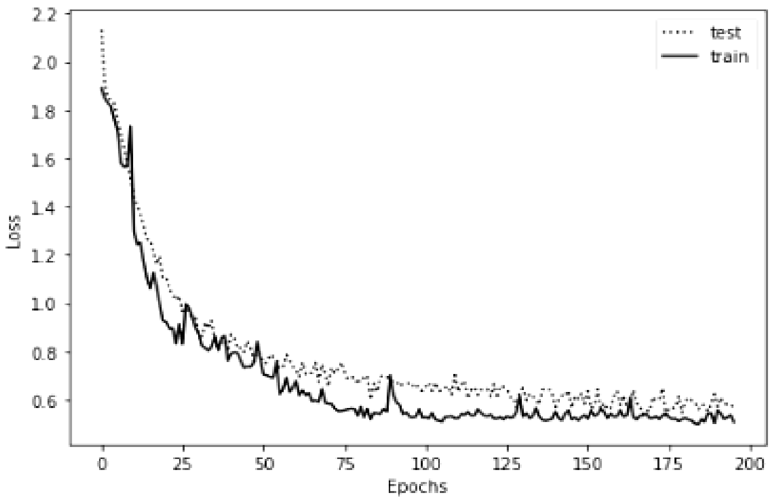
	Cora	Citeseer
GNN with BR	87.1	76.9
GNN With FS, CM	87.9	78.6
GNN With BR, FS, CM	89.5	80.10

3.2. Convergence of model accuracy and loss function

In Figure 2 after roughly 200 epochs, GNNFC model achieves the best AUC score in the test set. The validation loss converged to a stable value after 300 epochs of running the code. This is shown in Figure 2(a) which shows the convergence achieved by the model’s accuracy. The neural network's weights were updated using a binary cross-entropy loss function. Figure 2(b) depicts the convergence of the loss function.



(a)



(b)

Figure 2. The convergence of (a) model accuracy and (b) loss function over training and validation data respectively

4. CONCLUSION

In this work, a general GNN design space has developed based on feature selection using centrality measures. The significance of feature selection-based centrality measures in GNN training has been investigated. The fact that feature selection is a good direction in our modified GNN model is confirmed by the experiment results. Based on the experimental observations, a new GNN model called GNNFC has been proposed. The proposed GNNFC proved that it outperforms existing GNN models on the node classification problem using extensive tests.

REFERENCES

[1] L. Zhang, H. Song, N. Aletras, and H. Lu, “Graph node-feature convolution for representation learning,” *Prepr. arXiv1812.00086*, Nov. 2018.

[2] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *Prepr. arXiv1609.02907*, Sep. 2016.

[3] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Prepr. arXiv1706.02216*, Jun. 2017.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” *Prepr. arXiv1710.10903*, Oct. 2017.

[5] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” *Proceedings of the 24th ACM SIGKD*, Jun. 2018, doi: 10.1145/3219819.3219890.

[6] J. You, Y. Wang, A. Pal, P. Eksombatchai, C. Rosenberg, and J. Leskovec, “Hierarchical temporal convolutional networks for dynamic recommender systems,” *Prepr. arXiv1904.04381*, Apr. 2019.

[7] W. Jin, R. Barzilay, and T. Jaakkola, “Junction tree variational autoencoder for molecular graph generation,” in *Proceedings of the 35th International Conference on Machine Learning*, 2018, vol. 80, pp. 2323–2332.

[8] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, “Graph convolutional policy network for goal-directed molecular graph generation,” *Prepr. arXiv1806.02473*, Jun. 2018.

[9] M. Zitnik and J. Leskovec, “Predicting multicellular function through multi-layer tissue networks,” *Bioinformatics*, vol. 33, no. 14, pp. i190----i198, Jul. 2017, doi: 10.1093/bioinformatics/btx252.

[10] J. You, R. Ying, and J. Leskovec, “Design space for graph neural networks,” *Prepr. arXiv2011.08843*, Nov. 2020, [Online]. Available: http://arxiv.org/abs/2011.08843.

[11] X. Liu, X. Li, G. Fiumara, and P. De Meo, “Link prediction approach combined graph neural network with capsule network,” *Expert Systems with Applications*, vol. 212, Feb. 2023, doi: 10.1016/j.eswa.2022.118737.

[12] R. van den Berg, T. N. Kipf, and M. Welling, “Graph convolutional matrix completion,” *Prepr. arXiv1706.02263*, Jun. 2017.

[13] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” in *NIPS’18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 4805–4815.

[14] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, “An end-to-end deep learning architecture for graph classification,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, doi: 10.1609/aaai.v32i1.11782.

[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.

[16] J. Chen, T. Ma, and C. Xiao, “FastGCN: Fast learning with graph convolutional networks via importance sampling,” *Prepr. arXiv1801.10247*, Jan. 2018.

[17] J. Gasteiger, A. Bojchevski, and S. Günnemann, “Predict then propagate: Graph neural networks meet personalized PageRank,” *Prepr. arXiv1810.05997*, Oct. 2018.

[18] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, “Node similarity preserving graph convolutional networks,” in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, Mar. 2021, pp. 148–156, doi: 10.1145/3437963.3441735.

[19] F. Wu, T. Zhang, A. H. de Souza, C. Fifty, T. Yu, and K. Q. Weinberger, “Simplifying graph convolutional networks,” *Prepr. arXiv1902.07153*, Feb. 2019.

[20] Y. Rong, W. Huang, T. Xu, and J. Huang, “DropEdge: Towards deep graph convolutional networks on node classification,” *Prepr. arXiv1907.10903*, Jul. 2019.

[21] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, “Simple and deep graph convolutional networks,” in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 1725–1735.

[22] S. K. Maurya, X. Liu, and T. Murata, “Simplifying approach to node classification in Graph Neural Networks,” *Journal of Computational Science*, vol. 62, p. 101695, Jul. 2022, doi: 10.1016/j.jocs.2022.101695.

[23] J. Zhu *et al.*, “Graph neural networks with heterophily,” *Prepr. arXiv2009.13566*, Sep. 2020.

[24] H. Pei, B. Wei, K. C.-C. Chang, and B. Y. Yu Lei, “Geom-GCN: Geometric graph convolutional networks,” *ICLR 2020 Conference*, 2019.

[25] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” *Prepr. arXiv1806.03536*, Jun. 2018.

[26] G. Li, M. Müller, B. Ghanem, and V. Koltun, “Training graph neural networks with 1000 layers,” *Prepr. arXiv2106.07476*, Jun. 2021.

[27] J. Godwin *et al.*, “Simple GNN regularisation for 3D molecular property prediction & beyond,” *Prepr. arXiv2106.07971*, Jun. 2021.

[28] H. Chi, Y. Wang, Q. Hao, and H. Xia, “Residual network and embedding usage: New tricks of node classification with graph convolutional networks,” *Journal of Physics: Conference Series*, vol. 2171, no. 1, Jan. 2022, doi: 10.1088/1742-6596/2171/1/012011.

[29] D. B. Acharya and H. Zhang, “Feature selection and extraction for graph neural networks,” in *Proceedings of the 2020 ACM Southeast Conference*, Apr. 2020, pp. 252–255, doi: 10.1145/3374135.3385309.

[30] C. T. Duong, T. D. Hoang, H. T. H. Dang, Q. V. H. Nguyen, and K. Aberer, “On node features for graph neural networks,” *Prepr. arXiv1911.08795*, Nov. 2019.

[31] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *ICML’17: Proceedings of the 34th International Conference on Machine Learning*, 2017, vol. 70, pp. 1263–1272.

[32] H. Wang and J. Leskovec, “Unifying graph convolutional neural networks and label propagation,” *Prepr. arXiv2002.06755*, Feb. 2020.

[33] Y. Zhai, W. Song, X. Liu, L. Liu, and X. Zhao, “A chi-square statistics based feature selection method in text classification,” in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, Nov. 2018, pp. 160–163, doi: 10.1109/ICSESS.2018.8663882.

[34] H. Cui, Z. Lu, P. Li, and C. Yang, “On positional and structural node features for graph neural networks on non-attributed graphs,” *Prepr. arXiv2107.01495*, Jul. 2021.

[35] K. Nozawa, M. Kimura, and A. Kanemura, “Node centralities and classification performance for characterizing node embedding algorithms,” *Prepr. arXiv1802.06368*, Feb. 2018.

[36] S. Gómez, “Centrality in networks: Finding the most important nodes,” in *Business and Consumer Analytics: New Ideas*, Cham: Springer International Publishing, 2019, pp. 401–433.

[37] Y. Wu and K. He, “Group normalization,” *Prepr. arXiv1803.08494*, Mar. 2018.

[38] S. Ioffe, “Batch renormalization: Towards reducing minibatch dependence in batch-normalized models,” *Prepr. arXiv1702.03275*, Feb. 2017.

[39] W. L. Hamilton, *Graph representation learning*. Cham: Springer International Publishing, 2020.

[40] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, “Collective classification in network data,” *AI Magazine*, vol. 29, no. 3, Sep. 2008, doi: 10.1609/aimag.v29i3.2157.

[41] K. Kong *et al.*, “FLAG: Adversarial data augmentation for graph neural networks,” *ICLR 2021 Conference Blind Submission*, 2020, pp. 1-14.

[42] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, “Beyond homophily in graph neural networks: Current limitations and effective designs,” *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, pp. 1–12, 2020.

BIOGRAPHIES OF AUTHORS

Asmaa M. Mahmoud received the B.Sc. degree in science, in 2007, and the M.Sc. degrees in computer science, in 2019. She is currently an assistant lecturer in computer science with the Mathematics Department, Faculty of Science, Al-Azhar University, Cairo, Egypt. She has published several research papers in the field of AI and machine learning. She can be contacted at: asmaadaoud.1959@azhar.edu.eg.

Abeer S. Desuky received the B.Sc. degree in science, in 2003, and the M.Sc. and Ph.D. degrees in computer science, in 2008 and 2012, respectively. She is currently an Associate Professor in computer science with the Mathematics Department, Faculty of Science, Al-Azhar University, Cairo, Egypt. She has published several research papers in the field of AI, machine learning, meta-heuristic optimization, and data mining and analysis. She is also a supervisor of some master's and Ph.D. theses. She is a reviewer in many Scopus-indexed journals, such as IEEE Access and Peerj. She can be contacted at email: abeerdesuky@azhar.edu.eg.

Heba F. Eid is an Associate Professor at Faculty of Science, Al-Azhar University, Egypt. She received her Ph.D. degree in Network Intrusion Detection and M.S. degree in Distributed database systems, both from Faculty of Science, Al-Azhar University, Egypt. Her research interests include multi-disciplinary environment involving computational intelligence, pattern recognition, computer vision, bio-inspired computing and cyber security. Dr. Heba has served as a reviewer for various international journals and a program committee member of several international conferences at: heba.fathy@azhar.edu.eg

Hoda A. Ali received the B.Sc. degree ini8 science, and the M.Sc. and Ph.D. degrees in mathematics, in 1987 and 1990, respectively. She is currently a professor and department head in a Mathematics Department, Faculty of Science, Al-Azhar University, Cairo, Egypt. She has published several research papers in the field of functional and real analysis. She is also a supervisor of some master's and Ph.D. theses. She is a reviewer for various international journals and a program committee member of several international conferences. She can be contacted at: hodaabdeldiam.59@azhar.edu.eg.