RESEARCH ARTICLE OPEN ACCESS

Mixed Diffusion Graph Neural Network for Learning Graph Data

Abdullah M. Al-Gafri*
**(Department of Electrical Engineering, King Abdulaziz University, Kingdom of Saudi Arabia*

ABSTRACT
Machine learning algorithms and especially neural networks have proven to be excellent in processing complex data. Neural networks learn from the data presented to them and are then able to make predictions about the data. the various forms of data can heavily affect the choice of neural network model that will process them. Graph data lend themselves naturally to many physical phenomena. This necessitates the development of specialized neural network models that can handle this type of data. Information in graph data appear in two forms, individual node features, and the underlying structure of the graph. The proposed Mixed Diffusion Graph Neural Network (MDGNN) model utilizes diffusion process to combine the information available in the node features with the underlying graph structure. The model is tested on two benchmarking datasets, Cora and Citeseer. The model achieved an accuracy of 84.9% on Cora and 73.4% on Citeseer. Both results exceed those of previous models.
Keywords-Mixed Diffusion Graph Neural Network (MDGNN), semi-supervised learning, diffusion models, Citation networks

---

Date Of Submission: 09-05-2019 Date Of Acceptance: 24-05-2019

I. INTRODUCTION

The abundance of data that have accompanied the many technological advancements of this era led to increased demand for intelligent data processing algorithms. Artificial neural networks have demonstrated their ability to learn from data and make accurate decisions. The way the data is presented to a neural network generally affects the performance of the network. Taking the form of the data into consideration when designing the network can improve the end result. This approach aims to capture the most amount of information available in the data. Graph structured data appear in many aspects of life, such as chemical compounds [1], protein structure [2], social networks[3], wireless sensor networks [4], publications citations[5], and natural language [6].

1.1 Problem Definition

The problem of concern here is learning and making predictions from graph structured data. To that end we denote graph structure data with a graph G=(V,E,X,Y), where V is a set of n nodes, E is a set of edges, $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) \in \mathbb{R}^{n \times d}$ is the nodes feature matrix where $\mathbf{x}_i$ is a d-dimensional feature vector of node v_i, and $Y=(y_1,y_2,\dots,y_n)$ are the labels for the nodes in V. In real life problems, usually only a small subset of the data points is labeled. To mimic this scenario, we adopt a semi-supervised setting. This indicates that in the learning phase of the network only a subset of the node labels is available. In the testing phase, the network must predict the labels of the nodes whose labels were not available in the training phase.

1.2 Related Works

Graph domain representations have been used in modelling large databases. A rich literature exists for the problem of retrieving a data point from the database in response to a user query. The retrieved data point should be most similar to the query. A weighted graph is used to model the database where the initial edge weights encode the similarities of the data points. A popular approach to the problem is utilizing diffusion processes to improve the edge weights of the graph. The improved weights eventually improve the results of the retrieval operation. A diffusion process aims to diffuse/propagate the data throughout the graph utilizing its underlying structure[7]. Diffusion processes are also used in salient object detection problems in images [8].

Early works on learning graph structured data used to encode the graph into another format of data and then perform the training process. This encoding process often leads to loss of information that were available in the graph structure. A new neural network model was introduced in [9] and later on in [10] aptly named the graph neural network model (GNN). The model is considered as an extension to the recurrent neural network (RNN) architecture. RNNs are able to capture time-domain

sequence patterns, the researchers aimed to exploit this feature and extend it to graph structured data. In same way that RNNs have internal states for each time step, GNNs have internal state for each node in the graph

When learning longer time-domain sequences, RNNs suffered from vanishing gradients. Long-Term Short Memory (LSTM) [11]networks and gated recurrent units [12] introduced some gating mechanisms to solve the vanishing gradient issue. Naturally, this modification was carried to the GNN, introducing the Gated Graph Sequence Neural Networks (GGS-NN) [13]. Another extension to the GNN was the Graph Partition Neural Network (GPNN)[14]. Partitioning large graphs into smaller ones, and then aggregating the information from the smaller graphs allowed the network to better tackle large graphs. In the original GNN, data was aggregated between nodes using a fully connected neural network. Motivated by the idea that not all neighboring nodes are equally important to a given node, a new model was proposed. The new model relies on an attention mechanism instead of the fully connected layer to aggregate data from the neighboring nodes. The model was named Attention-based Graph Neural Network (AGNN)[15].

The success of Convolutional Neural Networks (CNN) inspired many attempts at generalizing the convolutional operation to graph structured data. In [16] the convolutional operation was generalized using the spectrum of the graph Laplacian. Relying on the new convolutional operation, a deep convolutional network was designed for graph-structured data [17]. In order to reduce the computational complexity of the convolutional operation, Chebyshev polynomials were used to approximate the spectral convolutional filters [18]. The graph convolutional network (GCN) [19] restricted the convolutional operation to include only the 1st order neighborhood. Researchers in [20] extended the GCN to work with hyper graphs, aptly naming the model HyperGCN. A network of GCNs (N-GCN) was proposed to capture relations further than the 1st order neighborhood [21].

All of the previously discussed models aggregate data from neighboring nodes using one way or another. One model proposed using an attentional mechanism to aggregate the data from neighboring nodes (Graph Attention Network GAT) [5]. The learnable attention mechanism assigns weights to each of the neighboring nodes. This allows the node in concern to "attend" more to neighbors with more relevant features (higher weights). The attention mechanism was first introduced as an enhancement to RNNs [22] and later was proved to be enough to build a model for machine translation tasks [23]. One modification to the GAT model was the Gated Attention Networks (GaAN) [24]. This model employs a convolutional network to control the outputs of the attention heads in the GAT model. It is argued that not all of the attention heads contribute useful information to a given node, thus the gating network would assign weights to each attention head.

The discussion above was focused on neural network graph learning models. There are also non-neural network approaches to the graph learning problem in the literature. For example; the Label Propagation algorithm (LP) [25], DeepWalk[26], Planetoid [27], Manifold Regularization (ManiReg) [28], Semi-supervised Embedding (SemiEmb) [29], and Iterative Classification Algorithm (ICA) [30]. We refer the reader to following comprehensive graph embedding techniques surveys [31], [32].

1.3 Purpose and Contributions

The purpose of this work is to design a specialized neural network model that can efficiently learn graph structured data and make accurate predictions on such types of data. The contributions of this work are twofold. The first is introducing a new flexible mixed graph diffusion model. The second contribution in the design of the Mixed Diffusion Graph Neural Network (MDGNN) model. A benchmarking test on two standard datasets validates the performance of the new model.

II. MIXED DIFFUSION GRAPH NEURAL NETWORK (MDGNN) MODEL

The proposed MDGNN is inspired the Graph Diffusion-Embedding Network (GDEN) [33]. The GDEN model aims to combine the information found in the node features with the underlying structure of the data graph. This was accomplished by using graph diffusion techniques. The authors introduced three such diffusion techniques and compared the results of each technique. We propose using a combination of the diffusion techniques and show that the performance is indeed improved.

2.1 Diffusion Techniques

Diffusion techniques enable us to incorporate the features of the other nodes in the feature representation of a given node. The new diffused features representation is $X_d = A_d X$, where A_d is the diffusion matrix and X is the original features matrix. The three diffusion matrices defined in [33] are:

The Lablacian diffusion matrix:

$$A_d = \alpha(I + \alpha L)^{-1} \quad (1)$$

The normalized Lablacian diffusion matrix:

$$A_d = (I - \alpha \mathcal{L})^{-1} \quad (2)$$

The random walks with restart diffusion matrix:

$$A_d = (1 - \alpha)(I - \alpha P)^{-1} \quad (3)$$

Where α is a tunable parameter, I is an $n \times n$ identity matrix, A is the $n \times n$ adjacency matrix of the graph, and D is an $n \times n$ diagonal matrix with $d_{ii} = \sum_j A_{ij}$. The diagonal matrix D is known as the degree matrix of the graph. The matrix $L = D - A$ is the Lablacian of the graph, $\mathcal{L} = D^{-1/2} AD^{-1/2}$ is known as the normalized Lablacian of the graph, and the matrix $P = AD^{-1}$ is called the transition probability matrix. We propose using a combination of the previous diffusion matrices:

$$M_d = \beta A_{d1} + (1 - \beta) A_{d2} \quad (4)$$

Where $0 < \beta < 1$ determines the mixing ratio of the diffusion techniques, A_{d1} and A_{d2} can be any of the previously defined diffusion matrices.

2.2 Semi-Supervised Learning Neural Network

The diffused features $X_d = M_d X$ capture the underlying structure of the graph along with the node features. In a graph embedding scenario, the diffused features can be linearly transformed as follows:

$$H = X_d W = M_d XW \quad (5)$$

Where $A_d \in \mathbb{R}^{n \times n}$, $X \in \mathbb{R}^{n \times d}$, $W \in \mathbb{R}^{d \times k}$ is the dimensionality reduction matrix provided $k < d$, and $V \in \mathbb{R}^{n \times k}$ is the new dimensionally-reduced features embedding for every node. From (5) it is possible to define a layer-wise propagation rule to be used in building a neural network:

$$H^{(l+1)} = \sigma\left(M_d H^{(l)} W^{(l)}\right) \quad (6)$$

Where l is the layer number, $\sigma(\cdot)$ is an activation function, M_d is the mixed diffusion matrix as defined in equation (4), $H^{(0)} = X$ is the starting feature matrix of the nodes, $H^{(l)}$ is the output of the l^{th} layer, $W^{(l)}$ is the trainable weight matrix of the l^{th} layer.

In the setting of semi-supervised learning, the final layer must predict the labels of the graph nodes. Therefore, a final layer with a softmax function can be used to predict the labels of the inputs:

$$\hat{Y} = \text{softmax}\left(A_d H^{(L)} W^{(L)}\right) \quad (7)$$

Where L is the number of the last layer, and $\hat{Y} \in \mathbb{R}^{n \times c}$ is the predicted labels matrix provided that c is the number of possible labels. A cross-entropy error loss function [19] can then be used over the set of labeled inputs Y_L.

$$\mathcal{L} = -\sum_{i \in Y_L} \sum_{j=1}^{c} Y_{ij} \ln(\hat{Y}_{ij}) \quad (8)$$

III. EXPERIMENTS

3.1 Datasets

In order to test the performance of the proposed neural network, two benchmarking real world datasets were used; Cora and Citeseer [34]. Both of them are citations networks. Each document in these citation networks represents a node in the graph and is described by a bag-of-words feature vector. The feature vectors are gathered in a matrix X. The datasets also contain a list for the citations between the documents. The citations represent the undirected edges of the graphs and are captured in the adjacency matrix A. Each document also has a label specifying which category it belongs to.

The Cora dataset has 2708 documents (nodes), 5429 edges, 1433 features (for each document/node), and 7 classes (labels). Given the semi-supervised learning setting and following along with the same data split used in previous works [27], only 20 labeled instances from each class are available in the training phase, i.e. 140 labeled nodes. The adjacency matrix and all of the feature vectors are available for training.

The Citeseer dataset on the other hand, has 3327 documents (nodes), 4732 edges, 3703 features (for each document/node), and 6 classes (labels). With the 20 labeled instances per class condition, 120 labeled nodes are used in training in addition to the adjacency matrix and the all of the feature vectors. Table 1 shows a summary of the two datasets.

3.2 Model Setup

In the implementation of the model, two layers of the MDGNN were used as defined in (6). A dropout rate of 0.5 was used on the first layer [35]. The two layers used an exponential linear unit (ELU) as activation functions [36]. The first layer reduces the dimension of the input to 16 and the second layer reduces it further to the number of possible classes. Both layers use L2 regularization of 10^{-5}. The output of the two layers was passed to a softmax function to generate the predictions of the labels. The cross-entropy error loss function was then applied on the labeled set of the inputs. The network was trained using the Adam optimizer [37] with a learning rate of 0.01. The values of α were set to 1, 0.65 and 0.91 for the Lablacian, normalized Lablacian and random walk diffusion matrices respectively. We note that multiple values for the parameter α were tested, the values mentioned achieved the best performance when using each diffusion matrix on its own without mixing ($\beta = 1$).

3.2.1 MDGNN Variations Testing

Given the three defined diffusion matrices, there are three possible variations of the MDGNN model,namely,MDGNN-L+RW, MDGNN-NL+RW, and MDGNN-L+NL, where L stands for the Lablacian matrix, LN stands for the normalized Lablacian matrix, and RW stands for the random walk base diffusion matrix. The three model variations were tested on the two datasets against different values of the mixing parameter β.

Table 1: Citation Datasets Statistics Summary

	Documents (Nodes)	Links (Edges)	Features	Classes (Labels)	Labels Used in Training	Labels Percentage
Cora	2708	5429	1433	7	140	5.17%
Citeseer	3327	4732	3703	6	120	3.61%

Fig. 1 below shows the achievable classifying accuracies of the different MDGNN variations for varying values of β for the Cora dataset. Several combinations already achieve an accuracy above the 83% mark (highest accuracy in the literature). The best performance is made by the MDGNN-NL+RW model with $\beta = 0.4$ for the Cora dataset achieving an accuracy of 84.9%.

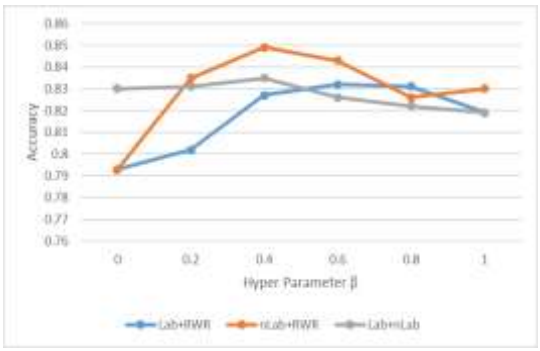

Fig.1: MDGNN variations accuracies for Cora dataset

Fig. 2 shows the performance of the MDGNN variations on the Citeseer dataset. It is clear that two MDGNN variations that utilize the random walk diffusion are performing better than the variation that does not. The best variation achieves a classification accuracy of 73.2%. The model is MDGNN-L+RW with $\beta = 0.4$.

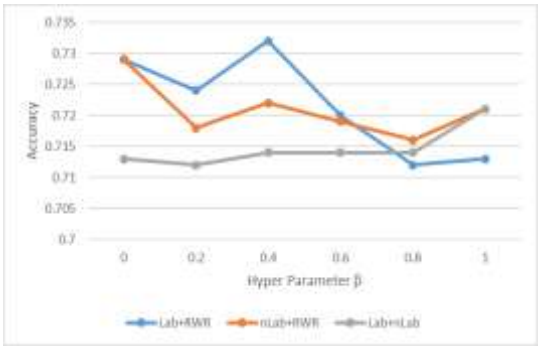

Fig.2: MDGNN variations accuracies for Citeseer dataset

3.3 Baselines

The results of the proposed model are compared against a number of baselines. Namely the baselines are; DeepWalk[26], CNN-Cheby[18], Planetoid [27], Graph Convolutional Network (GCN) [19], Manifold Regularization (ManiReg) [28], Semi-supervised Embedding (SemiEmb) [29], Label Propagation (LP) [25], Iterative Classification Algorithm (ICA) [30], and Diffusion-Convolutional Neural Network (DCNN) [38]. We also include the following baselines; Network of GCN (N-GCN) [21], Graph Attention Network (GAT) [5], and Graph Diffusion-Embedding Network (GDEN) [33].

3.4 Results

Table 2 shows the accuracy of the different methods in predicting the labels of the unlabeled nodes (node classification). The accuracies of the models were taken as reported in the literature.

Table 2: Accuracy of node classification on citation networks comparison in a semi-supervised learning setting

Method	Cora dataset	Citeseer dataset
SemiEmb	59.0%	59.6%
ManiReg	59.5%	60.1%
DeepWalk	67.2%	43.2%
LP	68.0%	45.3%
ICA	75.1%	69.1%
Planetoid	75.7%	64.7%
CNN-Cheby	79.2%	68.1%
GDEN-RWR	79.3%	72.9%
DCNN	81.3%	71.1%
GCN	81.5%	70.3%
GDEN-L	81.9%	71.3%
N-GCN	83.0%	72.2%
MDGNN-L+RW (proposed model)	82.7%	**73.4%**
MDGNN-NL+RW (proposed model)	**84.9%**	72.2%

IV. DISCUSSION

The semi-supervised graph learning challenge and specifically the node classification problem has been under investigation for an extended period of time as apparent by the numerous proposed methods. As Table 2 shows, the proposed model (MDGNN) achieves better node classification accuracy than the other models. To further validate the performance of the new model we used two different real world citation datasets, namely; Cora and Citeseer.

The performance of the model is heavily affected by the choices of A_{d1}, A_{d2} and β in (4). The best diffusion matrices combinations were found to include the random walk based diffusion and either

one of the Lablacian diffusions. The two Lablacian diffusion matrices originates from an optimization problem that seeks to balance the similarity of neighboring diffused features with retaining the original node features [33]. The random walk diffusion matrix originates from the random walk model. The random walk model originally aims to encode the structure of a given graph without the need for any node features. Combining the features focused Lablacian diffusions with the graph structure focused random walk diffusion generated the best performance. The chosen β value of 0.4 shows that the best performance lies in between the features focused and the graph structure focused diffusion models.

In regards to the computational complexity of the proposed model, the model needs only to perform the diffusion matrices calculations one time. The heaviest required calculations in the model are the matrices inversions in the diffusion equations. However, it is possible to calculate the two diffusion matrices in parallel to cut the computation time.

Diffusion models in general have shown promising results in the graph node classification problem [33]. Other diffusion models could be explored and tested in the semi-supervised learning setting [7]. The proposed model (MDGNN) showed that combining diffusion processes can further improve the performance of the node classifier. There is still a need to investigate other combining/fusing methods. There is also a need for a systematic way to calculate the optimum value for the fusing parameter β. Another possible future direction would be to extend the MDGNN model to handle directed graphs and graphs with edge features.

V. CONCLUSION

In this work, the proposed Mixed Diffusion Graph Neural Network (MDGNN) showed great promise in the graph node classification problem under semi-supervised learning. The new model outperformed the previous models from the literature. The model was defined in a layer-wise fashion that is easy to incorporate in any neural network design.

The diffusion processes presented in this workexhibited promising performance inthe task of semi-supervised graph data learning, however, there is still a need to findthe optimum value for the tuning parameter for each diffusion process. Thecombining parameter β could also be further investigated to achieve optimumperformance. The proposed combined diffusion model could also be applied to otherproblems such as graph auto-encoders and graph learning (instead of node learning).The neural network model used in the paper is similar to a simple multi-layerperceptron neural network, other network models such as recurrent neural networks(RNN) could be tested to adopt the learning model for different problem settings.

REFERENCES

[1]. A. Srinivasan, S. Muggleton, R. D. King, and M. J. E. Sternberg, "Mutagenesis: ILP experiments in a non-determinate biological domain," in Proceedings of the 4th international workshop on inductive logic programming, 1994, vol. 237, pp. 217–232.

[2]. P. Baldi and G. Pollastri, "The principled design of large-scale recursive neural network architectures--dag-rnns and the protein structure prediction problem," J. Mach. Learn. Res., vol. 4, no. Sep, pp. 575–602, 2003.

[3]. H. Ohtsuki, C. Hauert, E. Lieberman, and M. A. Nowak, "A simple rule for the evolution of cooperation on graphs and social networks," Nature, vol. 441, no. 7092, p. 502, 2006.

[4]. A. E. C. Redondi, "Radio Map Interpolation using Graph Signal Processing," IEEE Commun. Lett., vol. 22, no. 1, pp. 153–156, 2017.

[5]. P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," arXiv Prepr. arXiv1710.10903, vol. 1, no. 2, 2017.

[6]. A. Bua, M. Gori, and F. Santini, "Recursive neural networks applied to discourse representation theory," in International Conference on Artificial Neural Networks, 2002, pp. 290–295.

[7]. M. Donoser and H. Bischof, "Diffusion processes for retrieval revisited," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 1320–1327.

[8]. L. Zhou, Z. Yang, Q. Yuan, Z. Zhou, and D. Hu, "Salient region detection via integrating diffusion-based compactness and local contrast," IEEE Trans. Image Process., vol. 24, no. 11, pp. 3308–3320, 2015.

[9]. M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," Proc. Int. Jt. Conf. Neural Networks, vol. 2, pp. 729–734, 2005.

[10]. F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model.," IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 61–80, 2009.

[11]. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[12]. K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv Prepr. arXiv1406.1078, 2014.

[13]. Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," arXiv Prepr. arXiv1511.05493, 2015.

[14]. R. Liao, M. Brockschmidt, D. Tarlow, A. L. Gaunt, R. Urtasun, and R. Zemel, "Graph Partition Neural Networks for Semi-Supervised Classification," arXiv Prepr. arXiv1803.06272, 2018.

[15]. K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based Graph Neural Network for Semi-supervised Learning," arXiv Prepr.

arXiv1803.03735, 2018.
[16]. J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," arXiv Prepr. arXiv1312.6203, 2013.
[17]. M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," arXiv Prepr. arXiv1506.05163, 2015.
[18]. M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.
[19]. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv Prepr. arXiv1609.02907, 2016.
[20]. N. Yadati, M. Nimishakavi, P. Yadav, A. Louis, and P. Talukdar, "Hypergcn: Hypergraph convolutional networks for semi-supervised classification," arXiv Prepr. arXiv1809.02589, 2018.
[21]. S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification," arXiv Prepr. arXiv1802.08888., 2018.
[22]. D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," pp. 1–15, 2014.
[23]. A. Vaswani et al., "Attention Is All You Need," no. Nips, 2017.
[24]. J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs," arXiv Prepr. arXiv1803.07294, 2018.
[25]. X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in Proceedings of the 20th International conference on Machine learning (ICML-03), 2003, pp. 912–919.
[26]. B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," 2014.
[27]. Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting Semi-Supervised Learning with Graph Embeddings," vol. 48, 2016.
[28]. M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," J. Mach. Learn. Res., vol. 7, no. Nov, pp. 2399–2434, 2006.
[29]. J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in Neural Networks: Tricks of the Trade, Springer, 2012, pp. 639–655.
[30]. Q. Lu and L. Getoor, "Link-based classification," in Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 496–503.
[31]. P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," Knowledge-Based Syst., vol. 151, pp. 78–94, 2018.
[32]. H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications," IEEE Trans. Knowl. Data Eng., 2018.
[33]. B. Jiang, D. Lin, and J. Tang, "Graph Diffusion-Embedding Networks," arXiv Prepr. arXiv1810.00797, 2018.
[34]. P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," AI Mag., vol. 29, no. 3, p. 93, 2008.
[35]. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 2014.
[36]. D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," arXiv Prepr. arXiv1511.07289, 2015.
[37]. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv Prepr. arXiv1412.6980, 2014.
[38]. J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in Advances in Neural Information Processing Systems, 2016, pp. 1993–2001.

Abdullah M. Al-Gafri" Mixed Diffusion Graph Neural Network for Learning Graph Data" International Journal of Engineering Research and Applications (IJERA), Vol. 09, No.05, 2019, pp. 35-40