Get Rid of Suspended Animation: Deep Diffusive Neural Network for Graph Representation Learning

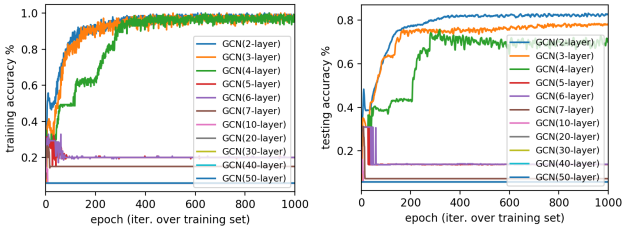
Jiawei Zhang [1]

Abstract

Existing graph neural networks may suffer from the "suspended animation problem" when the model architecture goes deep. Meanwhile, for some graph learning scenarios, e.g., nodes with text/image attributes or graphs with long-distance node correlations, deep graph neural networks will be necessary for effective graph representation learning. In this paper, we propose a new graph neural network, namely DIFNET (Graph Diffusive Neural Network), for deep graph representation learning and node classification. DIFNET utilizes both neural gates and graph residual learning for node hidden state modeling, and includes an attention mechanism for node neighborhood information diffusion. Extensive experimental results can illustrate both the learning performance advantages of DIFNET compared with existing methods, especially in addressing the "suspended animation problem".

1. Introduction

Graph neural networks (GNNs) (Wu et al., 2019; Zhang, 2019) have been one of the most important research topic in recent years, and many different types of GNN models have been proposed already (Kipf & Welling, 2016). Generally, via the aggregations of information from the neighbors (Hammond et al., 2011; Defferrard et al., 2016), GNNs can learn the representations of nodes in graph data effectively. The existing GNNs, e.g., GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2018) and their derived variants (Klicpera et al., 2018; Li et al., 2018; Gao et al., 2019), are mostly based on the approximated graph convolutional operator (Hammond et al., 2011; Defferrard et al., 2016). As pointed out in (Zhang & Meng, 2019; Zhang et al., 2020), most of the existing GNN models based on approximated graph convolutional operator will suffer from the "suspended animation problem" when the model architecture goes deep. Especially, when the GNNs' model depth reaches certain limit (namely, the GNNs' suspended animation limit), the model will not respond to the training data anymore and become not learnable.



(a) Training Accuracy (b) Testing Accuracy

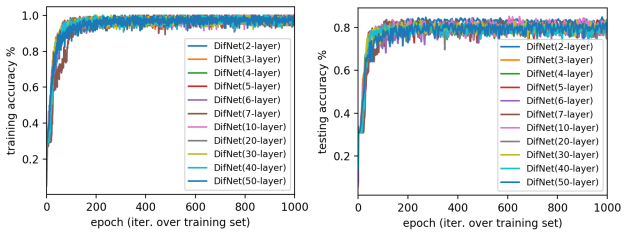
Figure 1: The learning performance of GCN (bias disabled) with 2-layer, . . . , 50-layer on the Cora dataset. X axis: training iterations; Y axis: training/testing accuracy.

As illustrate in Figure 1, we show a case study of the vanilla GCN model (with bias term disabled) on the Cora dataset (a well-known benchmark graph dataset). According to the results, the model can achieve the best performance with 2 layers, and its performance gets degraded as the depth increases and further get choked as the depth reaches 5 and more. Therefore, 5-layer is also called the "suspended animation limit" of GCN. Similar phenomena have been observed for the GCN model (with bias term enabled), GAT and their other derived models as well.

Some existing graph neural network models have been proposed to address the problem either with graph residual learning techniques (Zhang & Meng, 2019) or by extending BERT to the graph learning scenarios (Zhang et al., 2020). Meanwhile, in this paper, we will introduce a novel graph neural network model, namely DIFNET (Graph Diffusive Neural Net), for graph representation learning. DIFNET is not based on the approximated graph convolutional operator and can work perfectly with deep model architectures. To be more specific, DIFNET introduces a new neuron unit, i.e., GDU (gated diffusive unit), for modeling and updating the graph node hidden states at each layer. Equipped with both neural gates and graph residual learning techniques, GDU can help address the aforementioned "suspended animation problem" effectively. Furthermore, DIFNET also involves the attention mechanism for the neighboring node

[1]IFM Lab, Department of Computation, Florida State University, Tallahassee, FL, USA.. Correspondence to: Jiawei Zhang <jiawei@ifmlab.org>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.



(a) Training Accuracy (b) Testing Accuracy

Figure 2: The learning performance of DIFNET with 2-layer, 3-layer, . . . , 50-layer on the Cora dataset.

representation diffusion prior to feeding them into GDU. All the variables involved in GDU can be learned automatically with the backpropagation algorithm efficiently.

To demonstrate the effectiveness of DIFNET, in Figure 2, we also illustrate the learning performance of DIFNET on the Cora dataset (all the settings are identical to those of Figure 1). From the plots, we observe that DIFNET proposed in this paper can converge very fast and also outperforms GCN for the testing results. What's more, DIFNET generalizes very well to the deep architectures, and its learning performance can even get improved as the model depth increases. For instance, among all these layers, the highest testing accuracy score is achieved by DIFNET (10-layer). To be more specific, the layers for DIFNET denotes the number of GDU based layers. More detailed information and experimental results about DIFNET will be provided in the following sections of this paper.

We summarize the contributions of this paper as follows:

- We introduce a novel graph neural network model DIFNET in this paper, which can help address the "suspended animation problem" effectively on graph representation learning.
- We propose a new neuron model, i.e., GDU, which involves both neural gates and residual learning for node representation modeling and updating.
- DIFNET introduces an attention mechanism for the node representation diffusion and integration, which can work both effectively and efficiently.
- We test the effectiveness of DIFNET on several commonly used graph benchmark datasets and compare against both classic and state-of-the-art GNN models.

2. Method

To address the semi-supervised node classification problem, we will introduce the DIFNET (deep diffusive neural network) model in this part. We will first talk about the GDU neuron used in DIFNET, and also provide necessary information analyzing its effectiveness in addressing the suspended animation problem. For the neighborhood

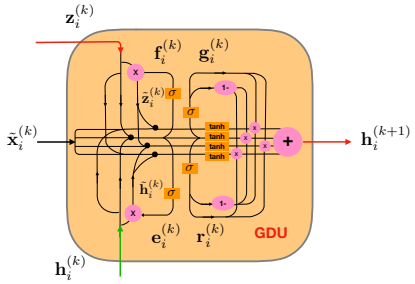


Figure 3: Architecture of the GDU neuron.

information diffusion, we will propose an attention mechanism for information propagation in DIFNET. Finally, detailed information about learning the DIFNET model will be introduced at the end.

2.1. Gated Diffusive Unit (GDU)

GDU initially proposed in (Zhang et al., 2018) is a new neuron model proposed for graph representation learning specifically. Formally, given the input graph data G involving the node set $\mathcal{V}$ and link set $\mathcal{E}$, we can denote the raw feature vector and neighbor set of node v_i as $\mathbf{x}_i$ and $\Gamma(v_i)$, respectively. The DIFNET model to be build involves multiple layers, and we can denote the hidden state of node v_i at the k_{th} layer of DIFNET as $\mathbf{h}_i^{(k)} \in \mathbb{R}^{d_h}$. For the nearby neighbors of v_i, we can denote their hidden states as $\{\mathbf{h}_j^{(k)}\}_{v_j \in \Gamma(v_i)}$. DIFNET introduce an attention based diffusion operator to propagate and aggregate the information among such neighbor nodes, which can be denoted as

$$\mathbf{z}_i^{(k)} = \text{Diffusion}\left(\left\{\mathbf{h}_j^{(k)}\right\}_{v_j \in \Gamma(v_i)}\right).$$

The above operator will be defined in detail in Section 2.2.

As illustrated in Figure 3, the GDU (e.g., for node v_i) has multiple input portals. Besides the neighborhood aggregated information $\mathbf{z}_i$, GDU will also accept v_i's lower-layer representation $\mathbf{h}_i^{(k)}$ and graph residual term $\tilde{\mathbf{x}}_i$ as the other two inputs. For the aggregated input $\mathbf{z}_i^{(k)}$ from the neighbors, certain information in the vector can be useless for learning v_i's new state. Therefore, GDU defines an *adjustment gate* $\mathbf{f}_i^{(k)}$ to update $\mathbf{z}_i^{(k)}$ as follows:

$$\tilde{\mathbf{z}}_i^{(k)} = \mathbf{f}_i^{(k)} \otimes \mathbf{z}_i^{(k)}, \text{ and } \mathbf{f}_i^{(k)} = \sigma\left(\mathbf{W}_f \left[\tilde{\mathbf{x}}_i \sqcup \mathbf{z}_i^{(k)} \sqcup \mathbf{h}_i^{(k)}\right]\right).$$

Here, $\sigma(\cdot)$ is the sigmoid function parameterized by $\mathbf{W}_f$.

Meanwhile, for the representation input from the lower-layer of v_i, i.e., $\mathbf{h}_i^{(k)}$, GDU also introduces a similar *evolving gate* $\mathbf{e}_i$ for adjusting the hidden state vector:

$$\tilde{\mathbf{h}}_i^{(k)} = \mathbf{e}_i^{(k)} \otimes \mathbf{h}_i^{(k)}, \text{ and } \mathbf{e}_i^{(k)} = \sigma\left(\mathbf{W}_e \left[\tilde{\mathbf{x}}_i^{(k)} \sqcup \mathbf{z}_i^{(k)} \sqcup \mathbf{h}_i^{(k)}\right]\right).$$

In addition, to involve the graph residual learning (Zhang & Meng, 2019) in DIFNET to overcome the suspended animation problem, GDU can also accept the graph residual terms as the third input, which can be denoted as

$$\tilde{\mathbf{x}}_i^{(k)} = \text{Graph-Residual}\left(\{\mathbf{x}_j\}_{v_j \in \mathcal{V}}, \left\{\mathbf{h}_j^{(k)}\right\}_{v_j \in \mathcal{V}}; G\right).$$

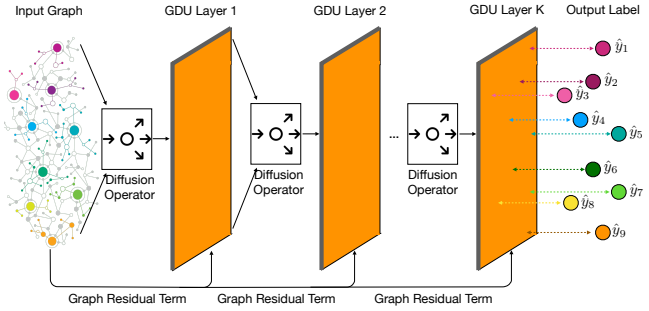


Figure 4: Framework architecture of DIFNET.

Formally, $\tilde{\mathbf{x}}_i^{(k)}$ can represent any graph residual terms defined in (Zhang & Meng, 2019). For instance, if the *naive residual term* is adopted here, we will have $\tilde{\mathbf{x}}_i^{(k)} = \mathbf{h}_i^{(k)}$; whereas for the *raw residual term*, we have $\tilde{\mathbf{x}}_i^{(k)} = \mathbf{x}_i$. For the *graph-naive residual term* and *graph-naive residual term*, $\tilde{\mathbf{x}}_i^{(k)}$ can be defined accordingly based on the graph adjacency matrix. In this paper, we will use the *graph-raw residual term*, as it can achieve the best performance as shown in (Zhang & Meng, 2019).

Based on these three inputs, $\tilde{\mathbf{x}}_i^{(k)}$, $\tilde{\mathbf{z}}_i^{(k)}$ and $\tilde{\mathbf{h}}_i^{(k)}$, GDU can effectively integrate the useful information from them to update the hidden state of node v_i. Instead of simple summation, integration of such information is achieved in GDU via two *selection gates* $\mathbf{g}_i$ and $\mathbf{r}_i$ denoted as follows:

$$\begin{aligned}\mathbf{h}_i^{(k+1)} &= \text{GDU}\left(\mathbf{z}_i, \mathbf{h}_i, \tilde{\mathbf{x}}_i; G\right)\\ &= \mathbf{g}_i^{(k)} \otimes \mathbf{r}_i^{(k)} \otimes \tanh\left(\mathbf{W}_u\left[\tilde{\mathbf{x}}_i^{(k)} \sqcup \tilde{\mathbf{z}}_i^{(k)} \sqcup \tilde{\mathbf{h}}_i^{(k)}\right]\right)\\ &+ (\mathbf{1} - \mathbf{g}_i^{(k)}) \otimes \mathbf{r}_i^{(k)} \otimes \tanh\left(\mathbf{W}_u\left[\tilde{\mathbf{x}}_i^{(k)} \sqcup \mathbf{z}_i^{(k)} \sqcup \tilde{\mathbf{h}}_i^{(k)}\right]\right)\\ &+ \mathbf{g}_i^{(k)} \otimes (\mathbf{1} - \mathbf{r}_i^{(k)}) \otimes \tanh\left(\mathbf{W}_u\left[\tilde{\mathbf{x}}_i^{(k)} \sqcup \tilde{\mathbf{z}}_i^{(k)} \sqcup \mathbf{h}_i^{(k)}\right]\right)\\ &+ (\mathbf{1} - \mathbf{g}_i^{(k)}) \otimes (\mathbf{1} - \mathbf{r}_i^{(k)}) \otimes \tanh\left(\mathbf{W}_u\left[\tilde{\mathbf{x}}_i^{(k)} \sqcup \mathbf{z}_i^{(k)} \sqcup \mathbf{h}_i^{(k)}\right]\right),\end{aligned}$$

where the selection gates

$$\begin{cases}\mathbf{g}_i^{(k)} &= \sigma\left(\mathbf{W}_g\left[\tilde{\mathbf{x}}_i^{(k)} \sqcup \mathbf{z}_i^{(k)} \sqcup \mathbf{h}_i^{(k)} \sqcup \tilde{\mathbf{z}}_i^{(k)} \sqcup \tilde{\mathbf{h}}_i^{(k)}\right]\right);\\ \mathbf{r}_i^{(k)} &= \sigma\left(\mathbf{W}_r\left[\tilde{\mathbf{x}}_i^{(k)} \sqcup \mathbf{z}_i^{(k)} \sqcup \mathbf{h}_i^{(k)} \sqcup \tilde{\mathbf{z}}_i^{(k)} \sqcup \tilde{\mathbf{h}}_i^{(k)}\right]\right).\end{cases}$$

In the above equation, term $\mathbf{1}$ denotes a vector filled with value 1 of the same dimensions as the *selection gate* vectors $\mathbf{g}_i^{(k)}$ and $\mathbf{r}_i^{(k)}$. Here, $\tanh(\cdot)$ denotes the hyperbolic tangent activation function and $\otimes$ denotes the entry-wise product.

2.2. Attention based Neighborhood Diffusion

In this part, we will define the Diffusion$(\cdot)$ operator used in Equation (2.1) for node neighborhood information integration. The DIFNET model defines such an operator based on an attention mechanism. Formally, given the nodes hidden states $\{\mathbf{h}_i^{(k)}\}_{v_i \in \mathcal{V}}$, DIFNET defines the diffusion operator together with the nodes integrated representations as follows:

$$\mathbf{Z}^{(k)} = \text{Diffusion}(\{\mathbf{h}_j^{(k)}\}_{v_j \in \mathcal{V}}) = \text{softmax}(\mathbf{M} \otimes \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}})\mathbf{V},$$

where $\mathbf{V} = \left[\mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}, \cdots, \mathbf{h}_{|\mathcal{V}|}^{(k)}\right]^\top \in \mathbb{R}^{|\mathcal{V}| \times d_h}$ covers the hidden states of all the nodes in the input graph. Matrices $\mathbf{Q}$ and $\mathbf{K}$ are the identical copies of $\mathbf{V}$. The output matrix $\mathbf{Z}^{(k)} = \left[\mathbf{z}_1^{(k)}, \mathbf{z}_2^{(k)}, \cdots, \mathbf{z}_{|\mathcal{V}|}^{(k)}\right]^\top \in \mathbb{R}^{|\mathcal{V}| \times d_z}$ denotes the aggregated neighbor representations of all the nodes. Term $\mathbf{M} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ denotes a mask matrix, where entry $\mathbf{M}(i,j) = 1$ iff $(v_i, v_j) \in \mathcal{E} \vee (v_j, v_i) \in \mathcal{E} \vee v_i = v_j$. Matrix $\mathbf{M}$ can effectively help prune the influences among nodes which are not connected in the original input graph.

2.3. Graph Diffusive Neural Network Learning

Based on both the GDU neurons and the attentive diffusion operator, we can construct the DIFNET model, whose detailed architecture is provided in Figure 4. Based on the input graph, DIFNET involves multiple layers of GDU neurons, which accept inputs from the previous layer, diffusion operators and the graph residual terms. In DIFNET, at each layer of the model, one GDU neuron is constructed to represent the state of each node. Based on the learned representations of each node at the last layer, one fully-connected layer is adopted to project the node representations to their labels. Formally, $\forall v_i \in \mathcal{V}$, according to Figure 4, its learning process of DIFNET can be denoted as the following equations:

$$\begin{cases}\text{/* Initialization: */}\\ \mathbf{x}_i = \text{Embedding}(\mathbf{x}_i), \text{ and } \mathbf{h}_i^{(0)} = \text{Linear}(\mathbf{x}_i, \mathbf{W}_x);\\ \tilde{\mathbf{x}}_i = \text{Graph-Residual}\left(\{\mathbf{x}_j\}_{v_j \in \mathcal{V}}, \left\{\mathbf{h}_j^{(k)}\right\}_{v_j \in \mathcal{V}}; G\right);\\ \text{/* Iterative Updating: */}\\ \begin{cases}\mathbf{z}_i^{(k)} = \text{Diffusion}\left(\{\mathbf{h}_j^{(k)}\}_{v_j \in \Gamma(v_i) \cup \{v_i\}}\right), \forall k \in \{0, \cdots, K\};\\ \mathbf{h}_i^{(k+1)} = \text{GDU}\left(\mathbf{z}_i, \mathbf{h}_i, \tilde{\mathbf{x}}_i; G\right), \forall k \in \{0, \cdots, K\};\end{cases}\\ \text{/* Output: */}\\ \hat{\mathbf{y}}_i = \text{softmax}\left(\mathbf{W}_{fc}\mathbf{h}_i^{(K+1)}\right).\end{cases}$$

In the above equation, notation Embedding$(\cdot)$ will embed the nodes' raw input features (usually in a large dimension) into a latent representation. Depending on the raw features, different models can be used to define the Embedding$(\cdot)$ function. For instance, CNN can be used if $\mathbf{x}_i$ is an image; LSTM/RNN can be adopted if $\mathbf{x}_i$ is text; and fully connected layers can be used if $\mathbf{x}_i$ is just a long feature vector. The nodes' hidden states $\mathbf{h}_i^{(0)}$ is also initialized based on the embedded raw feature vectors with a linear layer parameterized by $\mathbf{W}_x$. The learning process involves several diffusion and GDU layers, which will update nodes' representations iteratively. The final outputted latent state vectors will be projected to the nodes' labels with a fully-connected layer (also normalized by the softmax function).

Based on the set of labeled nodes, i.e., the training set $\mathcal{T}$, DIFNET computes the introduced loss by comparing the learned label against the ground-truth, which can be denoted as the following equation:

Table 1: Learning result by models with deeper architectures.

Method (Dataset)	Model Depth (Accuracy)				
	10	20	30	40	50
GCN (Cora)	0.215	0.057	0.057	0.057	0.057
DIFNET (Cora)	**0.851**	0.833	0.834	0.825	0.825
GCN (Citeseer)	0.064	0.064	0.064	0.064	0.064
DIFNET (Citeseer)	0.650	0.665	0.663	0.657	0.647
GCN (Pubmed)	0.366	0.366	0.366	0.366	0.366
DIFNET (Pubmed)	0.790	0.788	0.785	0.787	0.779

$$\ell(\mathbf{\Theta}) = \sum_{v_i \in \mathcal{T}} \sum_{d=1}^{d_y} -\mathbf{y}_i(d) \log \hat{\mathbf{y}}_i(d).$$

By minimizing the above loss term, DIFNET can be effectively learned with the error back-propagation algorithm.

3. Experiments

To test the effectiveness of the proposed DIFNET model, extensive experiments have been done on several graph benchmark datasets in this paper, including Cora, Citeseer, and Pubmed. Cora, Citeseer and Pubmed are the benchmark datasets used for semi-supervised node classification problem used in almost all the state-of-the-art GNN papers (Kipf & Welling, 2016; Veličković et al., 2018; Zhang & Meng, 2019). In the experiments, for fair comparison, we will follow the identical experimental settings (e.g., train/validation/test partition) as (Kipf & Welling, 2016).

Reproducibility: Both the datasets and source code used can be accessed via link[1]. Detailed information about the server used to run the model can be found at the footnote[2].

Experimental Settings: By default, the experimental settings of the DIFNET model is as follows on all these datasets: learning rate: 0.01 (0.005 on pubmed), max-epoch: 1000, dropout rate: 0.5, weight-decay: 5e-4, hidden size: 16, graph residual term: graph-raw.

3.1. Model Depth and Suspended Animation

As illustrated by Figure 2 provided in the introduction section, we have shown that DIFNET model can converge very fast even with less epochs than GCN. What's more, as the model depth increases, we didn't observe the suspended animation problem with DIFNET at all, which can perform well even with very depth architectures, e.g., 50 layers.

Furthermore, in Table 1, we also provide the learning accuracy of DIFNET (with different depths) on Cora, Citeseer and Pubmed, respectively. To illustrate its performance advantages, we also add the results of GCN (with different depths) on these three datasets in the table as well. The results demonstrate that DIFNET can also work well in handling the suspended animation problem for the other datasets from different sources as well.

[1]https://github.com/jwzhanggy/DifNet

[2]GPU Server: ASUS X99-E WS motherboard, Intel Core i7 CPU 6850K@3.6GHz (6 cores), 3 Nvidia GeForce GTX 1080 Ti GPU (11 GB buffer each), 128 GB DDR4 memory and 128 GB SSD swap. For the deep models which cannot fit in the GPU memory, we run them with CPU instead.

Table 2: Learning result accuracy of node classification methods. In the table, '-' denotes the results of the methods on these datasets are not reported in the existing works. We have run DIFNET with depth values from $\{2, 3, \cdots, 10, 20, \cdots, 50\}$. Results reported of DIFNET in this paper is the best one among all these depths.

Methods	Datasets (Accuracy)		
	Cora	**Citeseer**	**Pubmed**
LP	0.680	0.453	0.630
ICA ((Lu & Getoor, 2003))	0.751	0.691	0.739
ManiReg ((Belkin et al., 2006))	0.595	0.601	0.707
SemiEmb ((Weston et al., 2008))	0.590	0.596	0.711
DeepWalk ((Perozzi et al., 2014))	0.672	0.432	0.653
Planetoid ((Yang et al., 2016))	0.757	0.647	0.772
MoNet ((Monti et al., 2016))	0.817	-	0.788
GCN ((Kipf & Welling, 2016))	0.815	0.703	0.790
GAT ((Veličković et al., 2018))	0.830	0.725	0.790
LOOPYNET ((Zhang, 2018))	0.826	0.716	0.792
SF-GCN ((Lin et al., 2019))	0.833	**0.734**	**0.793**
DIFNET	**0.851**	**0.727**	0.783
	Time Cost (seconds)		
DIFNET	47.55	34.40	215.40

3.2. Comparison with State-of-the-Art

Besides the results showing that DIFNET can perform well in addressing the suspended animation problem, to make the experimental results more complete, we also compare DIFNET with both classic and state-of-the-art baseline models, whose results are provided in Table 2. According to the results, compared against these baseline methods, DIFNET can outperform them with great advantages.

4. Conclusion

In this paper, we focus on studying the graph representation learning problem and apply the learned graph representations for node semi-supervised classification. To address the common "suspended animation problem" with the existing graph neural networks, we introduce a new graph neural network model, namely DIFNET, in this paper. DIFNET is constructed based on both the GDU neurons and the attentive diffusion operator. To handle the suspended animation problem, GDU includes both neural gate learning and graph residual learning into its architecture, which can also handle the conventional gradient vanishing/exploding problem as well. Extensive experiments have been done on several benchmark graph datasets for node classification, whose results also demonstrate the effectiveness of both DIFNET and GDU in deep graph representation learning.

References

Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, December 2006. ISSN 1532-4435.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016. URL http://arxiv.org/abs/1606.09375.

Gao, Y., Yang, H., Zhang, P., Zhou, C., and Hu, Y. Graphnas: Graph neural architecture search with reinforcement learning. *CoRR*, abs/1904.09981, 2019. URL http://arxiv.org/abs/1904.09981.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, Mar 2011. ISSN 1063-5203. doi: 10.1016/j.acha.2010.04.005. URL http://dx.doi.org/10.1016/j.acha.2010.04.005.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

Klicpera, J., Bojchevski, A., and Günnemann, S. Personalized embedding propagation: Combining neural networks on graphs with personalized pagerank. *CoRR*, abs/1810.05997, 2018. URL http://arxiv.org/abs/1810.05997.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. *CoRR*, abs/1810.10659, 2018. URL http://arxiv.org/abs/1810.10659.

Lin, G., Wang, J., Liao, K., Zhao, F., and Chen, W. Structure fusion based on graph convolutional networks for semi-supervised classification. *CoRR*, abs/1907.02586, 2019. URL http://arxiv.org/abs/1907.02586.

Lu, Q. and Getoor, L. Link-based classification. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 496–503. AAAI Press, 2003. ISBN 1577351894.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. *CoRR*, abs/1611.08402, 2016. URL http://arxiv.org/abs/1611.08402.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014. URL http://arxiv.org/abs/1403.6652.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

Weston, J., Ratle, F., and Collobert, R. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning*, ICML'08, pp. 1168–1175, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390303. URL https://doi.org/10.1145/1390156.1390303.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. 2019.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. *CoRR*, abs/1603.08861, 2016. URL http://arxiv.org/abs/1603.08861.

Zhang, J. Deep loopy neural network model for graph structured data representation learning. *CoRR*, abs/1805.07504, 2018. URL http://arxiv.org/abs/1805.07504.

Zhang, J. Graph neural networks for small graph and giant network representation learning: An overview. *CoRR*, abs/1908.00187, 2019.

Zhang, J. and Meng, L. Gresnet: Graph residual network for reviving deep gnns from suspended animation. *CoRR*, abs/1909.05729, 2019.

Zhang, J., Dong, B., and Yu, P. S. Fake news detection with deep diffusive network model. *CoRR*, abs/1805.08751, 2018. URL http://arxiv.org/abs/1805.08751.

Zhang, J., Zhang, H., Xia, C., and Sun, L. Graph-bert: Only attention is needed for learning graph representations. *CoRR*, abs/2001.05140, 2020.

5. Appendix: More Discussions on GDU

Here, we would like to provide more discussions about the GDU neuron, and also introduce a simplified version of GDU to lower down the learning cost of DIFNET.

5.1. Design of GDU on Suspended Animation

The design of the GDU architecture can help DIFNET solve the "suspended animation problem" when the model goes deep in three perspectives:

- *Anti-Suspended Animation*: Instead of integrating the neighborhood information with the approximated graph convolutional operator via either equal importance (as GCN) or with weighted summation (as GAT), GDU introduces (a) an attentive neighborhood information diffusion and integration mechanism (to be introduced in the following subsection), and (b) a node state adjustment mechanism via four neural gates defined above. GDU is not based on the approximated graph convolutional operator at all. Both the attentive diffusion mechanism and neural gates based adjustment mechanism are dynamic, which can compute the parameters automatically depending on the specific learning scenarios and can overcome the suspended animation side-effects on the model learning performance.
- *Residual Learning*: As introduced in (Zhang & Meng, 2019), the graph residual terms (especially the *raw residual term* and *graph-raw residual term*) can help the GNN models address the "suspended animation problem". Both theoretic proofs and empirical experiments have demonstrated its effectiveness. The architecture of GDU also effectively integrate the graph residual learning mechanism into the learning process, where the input portal $\tilde{\mathbf{x}}_i^{(k)}$ can accept various graph residual terms.
- *Gradient Exploding/Vanishing*: In addition, as introduced in (Hochreiter & Schmidhuber, 1997), for the neural network models in a deep architecture, the learning process may also suffer from the gradient exploding/vanishing problem. Similar problem has been observed for the GNN models, which is different from the "suspended animation problem" that we mention above. Meanwhile, similar to LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Chung et al., 2014), the neural gates introduced in GDU can help ease even solve the "gradient exploding/vanishing" problem effectively.

Viewed in such perspectives, GDU is very different from the neurons used in the existing GNN models. In the following part, we will focus on applying GDU to construct the DIFNET model for the graph data representation learning.

5.2. Simplified GDU

Meanwhile, we also observed that the GDU neuron involves a complex architecture involving multiple variables, which may lead to a relatively high time cost for learning the model. To lower down the learning time cost, we also introduce a simplified version of GDU in this paper here. The main difference of the simplified GDU versus the original one introduced before lie in Equations 2.1-2.1, which can be denoted as follows:

$$\begin{aligned}\mathbf{h}_i^{(k+1)} &= \text{GDU}\left(\mathbf{z}_i, \mathbf{h}_i, \tilde{\mathbf{x}}_i; G\right) \\ &= \tanh\Big(\mathbf{g}_i^{(k)} \otimes \mathbf{W}_u[\tilde{\mathbf{z}}_i^{(k)} \sqcup \tilde{\mathbf{h}}_i^{(k)}] \\ &+ (\mathbf{1} - \mathbf{g}_i^{(k)}) \otimes \mathbf{W}_u[\mathbf{z}_i^{(k)} \sqcup \mathbf{h}_i^{(k)}] + \mathbf{W}'_u\tilde{\mathbf{x}}_i\Big),\end{aligned}$$

where the selection gates $\mathbf{g}_i^{(k)} = \sigma\left(\mathbf{W}_g\left[\tilde{\mathbf{z}}_i^{(k)} \sqcup \tilde{\mathbf{h}}_i^{(k)}\right]\right)$. According to the equations, one of the selection gate $\mathbf{r}_i^{(k)}$ defined in Equation 2.1 is removed, and the definition of gate $\mathbf{g}_i^{(k)}$ is also simplified, which is determined by the adjusted representations $\tilde{\mathbf{z}}_i^{(k)}$ and $\tilde{\mathbf{h}}_i^{(k)}$ only. Also the architecture of GDU is also updated, where the node state updating equation will balance between $\mathbf{W}_u[\tilde{\mathbf{z}}_i^{(k)} \sqcup \tilde{\mathbf{h}}_i^{(k)}]$ and $\mathbf{W}_u[\mathbf{z}_i^{(k)} \sqcup \mathbf{h}_i^{(k)}]$ (controlled by the gate $\mathbf{g}_i^{(k)}$). The residual term $\mathbf{W}'_u\tilde{\mathbf{x}}_i$ is added to the representation at the end only, where $\mathbf{W}'_u$ is the variable for vector dimension adjustment, which can be shared across layers with the same hidden dimensions.

5.3. Experimental Tests of Simplified GDU

Table 3: Learning result comparison of DIFNET (original GDU) vs DIFNET (simplified GDU).

Methods	Datasets (Accuracy)		
	Cora	**Citeseer**	**Pubmed**
DIFNET	**0.851**	**0.727**	0.783
DIFNET (simplified GDU)	**0.834**	0.715	**0.795**
	Time Cost (seconds)		
DIFNET	47.55	34.40	215.40
DIFNET (simplified GDU)	23.34	24.96	113.77

For DIFNET with the simplified GDU, it can achieve comparable performance as DIFNET with the original GDU with the same learning epochs. However, the learning time cost with simplified GDU is greatly reduced according to the experiment results.