Residual or Gate? Towards Deeper Graph Neural Networks for Inductive Graph Representation Learning

Binxuan Huang
binxuanh@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Kathleen M. Carley
kathleen.carley@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Abstract

In this paper, we study the problem of node representation learning with graph neural networks. Previous research has shown that graph neural networks (GNNs) are an effective framework for representation learning of graphs. However, training of deeper versions of GNNs becomes difficult. We show that using recurrent units to capture the long-term dependency across layers can successfully identify important information during recursive neighborhood expansion. In our experiments, we show that this model class achieves state-of-the-art results on three benchmarks: the Pubmed, Reddit, and PPI network datasets. Our in-depth analyses also demonstrate that incorporating recurrent units is a simple yet effective method to prevent noisy information in graphs, which enables a deeper graph neural network.

1 Introduction

Graphs are universal models of objects and their pairwise relationships. We can view many data in the form of graphs, including social networks, protein interactions, paper citations. But unlike sequence data or grid data, it is hard to express and exploit graph information in many machine learning tasks. Recently substantial efforts have been made to learn expressive structure information in graphs [21, 7, 28].

Graph neural networks are deep learning-based methods that operate on graphs. At each layer, GNNs aggregate information from neighbourhoods and generate hidden states for each node. Because GNNs do not require a fixed graph, we can easily apply them to new graphs on the fly, which is suitable for the inductive setting. In recent proposed GNNs, there is a common drawback that training becomes extremely difficult when models become deeper [12]. This is partially due to more layers would also propagate noisy information from expanded neighborhood [12]. Though researchers try to use residual connection [9] to overcome this issue [12, 30], the performance still gets worse with deeper models. In this paper, we will show that using residual connection is not the best option for a deep graph neural network. Rather, incorporating recurrent units in graph neural networks can effectively capture the neighbourhood information while keeping local features unvarnished. In this work, we present a deep graph neural network class named recurrent graph neural networks (RGNN). It uses recurrent units to compress previous neighborhood information into hidden states and successfully captures useful information during recursive neighborhood expansion.

In our experiments, we systematically evaluate the performance of RGNNs under supervised learning setting as well as unsupervised setting. In our comparative evaluation, we show RGNNs can consistently improve the performance of base GNN models. This model class achieves state-of-the-art results on three commonly used benchmark datasets. We further compare this neural network class with GNNs with residual connections. Experiments show that RGNNs have better learning capability with the same number of layers and can effectively avoid noisy neighbourhood information.

2 Recurrent Graph Neural Network

In a GNN model, each layer l can potentially capture information from neighbours with l-hops distance. Such deep GNNs could propagate noisy information from the expanded neighbourhood. An intuitive thought would be can we use recurrent units to model long-term dependency across layers. If we take hidden states across layers as a sequence of observations, a recurrent unit with good sequence modeling capability can ideally compress previous graph history into node states and control how much information should be added to new hidden states.

With this intuition, we present the general recurrent graph neural network framework as follows:

$$H^{l+1} = RNN(GNN(H^l, A; \Theta^l), H^l), \quad l \geq 0 \tag{1}$$

$$H^0 = RNN(W_i X + b_i, 0) \tag{2}$$

where at each layer $GNN(H^l, A; \Theta^l)$ generates new input for an RNN unit, and this RNN unit decides how much information should be added into the next layer. The initial hidden state H^0 is generated by feeding node local features into the RNN unit. W_i, b_i are a projection matrix and a bias vector that maps input features into the dimension of hidden states.

An intuitive view of this RGNN model is that at layer 0 the hidden state h^0 is only dependent on the node's local features, and at each layer l information from l-hop neighbourhood is compressed into the hidden state by a recurrent unit.

Take a graph convolutional neural network [12] with long short-term memory [10] for example, it updates node representations at each layer as follows:

$$\hat{X}^{l+1} = \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^l \Theta^l \qquad I^{l+1} = \sigma(X^{l+1} W_i + H^l U_i + [b_i]_N) \tag{3}$$

$$F^{l+1} = \sigma(X^{l+1} W_f + H^l U_f + [b_f]_N) \qquad O^{l+1} = \sigma(X^{l+1} W_o + H^l U_o + [b_o]_N) \tag{4}$$

$$\hat{C}^{l+1} = tanh(X^{l+1} W_c + H^l U_c + [b_c]_N) \qquad C^{l+1} = F^{l+1} \circ C^l + I^{l+1} \circ \hat{C}^{l+1} \tag{5}$$

$$H^{l+1} = O^{l+1} \circ tanh(C^{l+1}) \tag{6}$$

where $[b]_N$ represents stacking bias vector $b \in R^C$ N times and forms a bias matrix with dimension $R^{N \times C}$. N is the number of nodes, C is the dimension of hidden states. Similar update rules can be writen for other GNNs like graph attention neural networks (GAT) [30].

Note that for a large-scale graph with millions of nodes, training for the whole graph becomes unfeasible because of the memory limitation. We use the sampling method proposed in GraphSAGE [8] for batched training. At each training iteration, we first sample a small batch of nodes B_0 and then recursively expand B_l to B_{l+1} by sampling S_l neighbourhood nodes of B_l. With a GNN of M layers, we get a hierarchy of nodes: $B_0, B_1, ..., B_M$. Representations of target nodes B_0 are updated by aggregating node states from the bottom layer B_M to the upper layer B_0.

Supervised Learning: Given a final representation h_i for node v_i, we first project h_i into the classification space and get an output $o_i = W_o h_i + b_o$ where $W_o \in R^{|Y| \times C}, b_o \in R^{|Y|}$ are a projection matrix and a bias vector. $|Y|$ is the number of target classes. In a multi-label classification case, where labels are not mutually exclusive, the loss function for node v_i is written as $loss = \frac{1}{|Y|} \sum_{j=1}^{|Y|} [y_{ij} log(\sigma(o_{ij})) + (1 - y_{ij}) log(1 - \sigma(o_{ij}))]$, where $y_{ij} \in \{0, 1\}$ is the label for class j. In a multi-class classification setting, where labels are mutually exclusive, the loss is the cross-entropy loss after softmax, which is $-y_{ij} log(\frac{exp(o_{ij})}{\sum_j exp(o_{ij})})$.

Unsupervised Learning: Following previous work [28, 8], in the unsupervised setting, we learn node representations by network modeling. Specifically, given a node v_i with representation h_i, the goal is to optimize the probability of observing a context node v_j as $p(v_j | v_i) = \frac{exp(h_j^T h_i)}{\sum_{k=1}^N exp(h_k^T h_i)}$, where context node v_j is generated by a random walk starting from node v_i. We use negative sampling [19] to approximate it and the objective becomes $log\sigma(h_j^T h_i) + \sum_{k=1}^K E_{v_k \sim P_N(v)} [log\sigma(-h_k^T h_i)]$. The task turns into distinguishing the context node v_j from K randomly sampled negative nodes. We use uniform distribution $P_N(v)$ here. To further reduce memory consumption in our batched training, nodes in one batch share the same set of negative nodes, which works well in practise.

3 Experiments

3.1 Datasets

We adopt three commonly used benchmark datasets in our experiments – Pubmed [24], Reddit [8], PPI [32]. A summary of these datasets and the experiments setup are shown in the Appendix A.

3.2 Baseline Comparisons

Results of comparative evaluation experiments are shown in Table 1. We evaluate GCN and GAT with LSTM/GRU units. In the supervised setting, we compare RGNN based models with various baselines — GCN, FastFCN [2], GAT, and GraphSAGE models. In the unsupervised setting, we use GraphSAGEs as baselines.

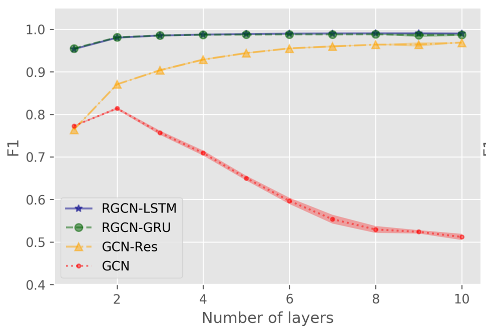
Table 1: Comparative evaluation results for three datasets. We report micro-averaged F1 scores. "-" signifies no results are published for the given setting.

Methods	Pubmed	Reddit		PPI	
	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
GCN	0.875	-	0.930	-	0.865
FastGCN	0.880	-	0.937	-	0.607
GAT	0.883	-	0.950	-	0.973
GraphSAGE-GCN	0.849	0.908	0.930	0.465	0.500
GraphSAGE-mean	0.888	0.897	0.950	0.486	0.598
RGCN-LSTM	**0.908**	0.919	0.963	0.791	0.992
RGCN-GRU	0.900	0.915	**0.964**	0.765	0.991
RGAT-LSTM	0.905	**0.921**	**0.964**	**0.806**	**0.994**
RGAT-GRU	0.902	0.913	**0.964**	0.791	0.994

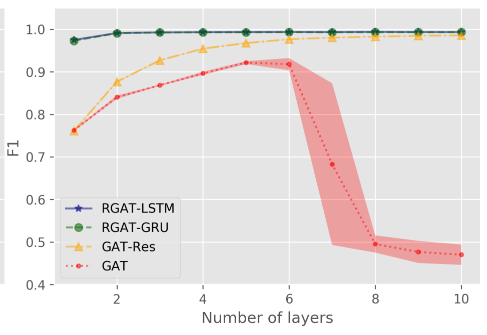
Our results demonstrate that unifying recurrent units in modern GNN models can effectively improve state-of-the-art performance across all three datasets. In the supervised learning setting, we are able to improve GCNs by an absolute increase of 12.7% on PPI. For Reddit and Pubmed, 2% - 4% improvement is achieved. Note that even Veličković et al. use residual connections for GAT on PPI dataset, their result is still worse than RGAT with recurrent units.

In the unsupervised setting, we observe similar improvement on Reddit and PPI datasets. Noticeably, RGNN based models perform much better on PPI than baselines under unsupervised learning. Our best model RGAT-LSTM achieves over 30% improvement over GraphSAGE, which is even better than some baseline models with supervised signals. Comparing RGCN-LSTM with GraphSAGE-GCN, we can find the LSTM unit provides a significant gain on this task.

3.3 Model Depth Analysis



(a) GCN based models on PPI



(b) GAT based models on PPI

Figure 1: Influence of model depth (number of layers) on performance. Markers denote averaged micro-F1 scores on test dataset in 5 runs. Shaded areas represent standard deviations. We show results for RGNN with RNN units, GNN with residual connections, and standard GNN models.

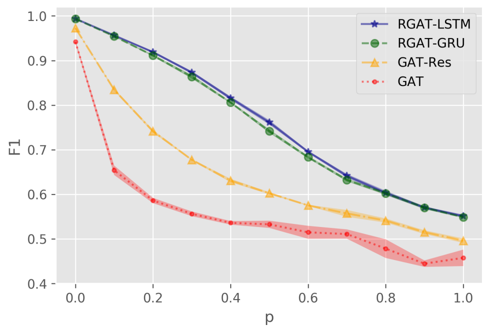
In this section, we investigate the influence of model depth (number of layers) on performance and compare the effects of adding recurrent units and residual connection. In this experiment, we use the same hyperparameter setting across all the base models. We run each method 5 times with various depths on PPI and Pubmed under supervised setting. Because of the GPU memory limitation, we change the dimension of hidden states from 1024 to 512 on PPI in these experiments.

As shown in Figure 1a and 1b, on PPI dataset, GNNs with recurrent units can be easily extended to deeper models and perform noticeably better than GNNs with residual connections (GNN-Res). A vanilla GCN degenerates quickly when the depth increases to 3 or higher. The GAT is better than the GCN, but it still fails when its depth goes beyond 6. Using residual connections does help GAT and GCN models to generalize to deeper models. However, performances of GNN models with residual connections are still worse than GNNs with recurrent units. RGCN models and RGAT models can quickly reach and maintain their optimal performances on PPI dataset. Although the difference between GAT-Res and RGAT models becomes less when depth gets larger, a 10-layer GAT-Res is still worse than 10-layer RGAT-LSTM/RGAT-GRU (0.985 versus 0.993/0.993). A similar analysis on Pubmed dataset is shown in the Appendix B.

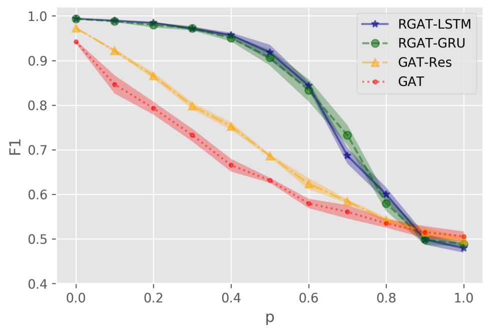
3.4 Perturbation Analysis

For many real-world problems, we do not have access to accurate information about the graph. Many times, there is noisy information in the graph structure and nodes' local features. In this section, we perform a perturbation study where we compare 3-layer RGNN models against 3-layer GNNs with imperfect information on PPI dataset under supervised learning.

In the first noisy graph scenario, we cut an edge with probability p and connect two randomly selected nodes. When p equals to 1, the reconstructed graph turns to be a random graph with the same graph density. We measure the performance of RGNNs, GNNs with residual connections, and GNNs under various probability p. In Figure 2a, we observe that the GAT models with LSTM and GRU units are more robust to noisy graph information, which shows that gates in these two RNN units are helpful for capturing important information and avoiding noisy graph information. RGNNs with LSTM units generally work better than RGNN with GRU units in this case.



(a) GAT based models with noisy graphs



(b) GAT based models with noisy features

Figure 2: Perturbation analysis. Markers denote averaged micro-F1 scores on test graphs of PPI in 5 runs. Shaded areas represent standard deviations. We show results for RGNN with LSTM and GRU units, GNN with residual connections, and standard GNN models.

In the second noisy feature scenario, for each node, we randomly mutate its local features with probability p, where we replace its features with Gaussian noises draw from $N(0, 1)$. As shown in Figure 2b, RGNN models have a better capability of distinguishing noisy features than GNNs with residual connections. RGAT-LSTM and RGAT-GRU work similarly and they both outperform GAT-Res and GAT in a large margin. Same analyses for GCN are shown in the Appendix C.

4 Conclusion

In this paper, we systematically evaluate the effect of adding recurrent units. Our results demonstrate that GNN models with recurrent units are much easier to extend to deeper models than GNN models with residual connections. Compared to previous methods, the presented RGNN models establish new state-of-the-art results on three benchmark datasets under both the supervised setting and the unsupervised setting. In our further analyses, we show RGNN models are more robust to noisy information from graph structure as well as local features, which enables a deeper graph neural network.

References

[1] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[2] J. Chen, T. Ma, and C. Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[6] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[7] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[8] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[13] T. Lei, Y. Zhang, and Y. Artzi. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755*, 2017.

[14] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5457–5466, 2018.

[15] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[16] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[17] Z. Liu, C. Chen, L. Li, J. Zhou, X. Li, L. Song, and Y. Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4424–4431, 2019.

[18] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[20] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[21] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[22] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 459–467. ACM, 2018.

[23] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[24] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[25] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[27] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[31] 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 preprint arXiv:1803.07294*, 2018.

[32] M. Zitnik and J. Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

Appendix

A. Datasets and Experimental Setup

We adopt three commonly used benchmark datasets in our experiments. A summary of these datasets is shown in Table 2.

Pubmed is a citation dataset introduced by [24]. Nodes represent academic papers within the Pubmed database and links are citations between papers. Node features are sparse bag-of-words representations of papers. Labels are the categories of these papers. Following [2], we use all labeled training examples for training per the supervised learning scenario. Because of the sparsity of this graph, we only test models with the supervised setting on Pubmed.

Reddit is a social network dataset compiled in [8]. It contains 232K Reddit posts as nodes. If the same user comments on two posts then there is a link between these two posts. Node features are generated from Glove word embeddings [20]. The node label in this case is the "subreddit" a post belongs to. Because of the graph size, we apply batched training on this dataset.

PPI contains 24 protein-protein interaction graphs, with each graph corresponding to a different human tissue [32]. Node features include positional gene sets, motif gene sets, and immunological signatures. Node labels are protein roles in terms of their cellular functions. Following [8], we train all models on 20 graphs, validate and test on 2 graphs each. It validates the generalizing performance across graphs.

Table 2: Statistics of three datasets

Data	# Nodes	# Edges	# Features	# Classes
Pubmed	19, 717 (1 graph)	44, 338	500	3
Reddit	232,965 (1 graph)	11, 606, 919	602	41
PPI	56,944 (24 graph)	818,716	50	121

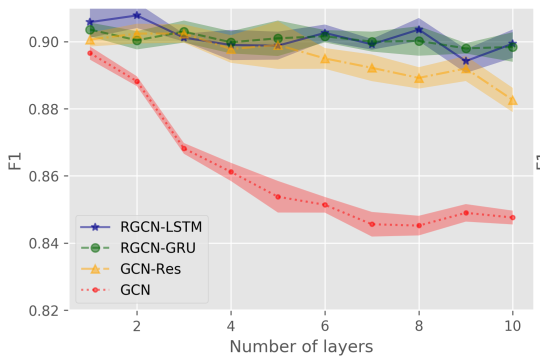
Supervised learning: We set dimensions of hidden states as 64, 600, and 1024 for Pubmed, Reddit, and PPI respectively. For GAT based models, we use 8 heads for Pubmed, 5 heads for Reddit, 4 heads for PPI. We apply dropout [26] on the input features for Pubmed, PPI with dropout rate 0.2. We apply two-layer GNN models for Pubmed and Reddit, and three-layer ones for PPI. Each layer in GNNs is followed by an exponential linear unit (ELU) nonlinearity [5]. Models are trained with Adam optimizer [11] with an initial learning rate of 0.01 for Pubmed, and 0.001 for other datasets. We apply batched training on the Reddit dataset with neighborhood sample sizes $S_1 = 25$ and $S_2 = 10$. The batch size is 128. Because of the dataset size, we run models 5 times and report an average performance for Pubmed and PPI.

Unsupervised learning: In the unsupervised setting, we use two-layer GNN models. The negative sampling size K is 10. Random walk lengths for PPI and Reddit are 2 and 3 respectively. Other hyper-parameter settings are the same with supervised learning, except for that we have not applied dropout here. After we get the node representations with unsupervised learning, we use representations of training nodes to train a downstream linear classifier same as GraphSAGE [8].

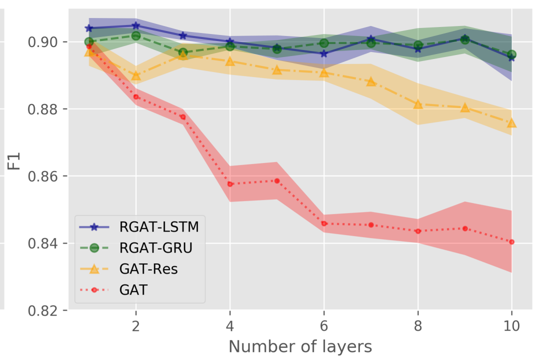
In both learning scenarios, we strictly follow the inductive learning setting where validation and test nodes are hidden during training. We ran our experiments on a linux machine with 4 NVIDIA Titan XP GPUs (12GB of RAM), one Intel Core i7-6850K CPU, 128GB of RAM.

B. Model Depth Analysis on Pubmed

In Figure 3a and 3b, for Pubmed dataset, best results are obtained with shallow models. Without residual connections or recurrent units, the performance of GNN models decreases with larger depth. GNN-Res models degenerate when the model is deeper than 5 layers. On the contrary, deep RGCN and RGAT still work similarly to shallow models, as LSTM and GRU successfully capture the long-term dependency.



(a) GCN based models on Pubmed

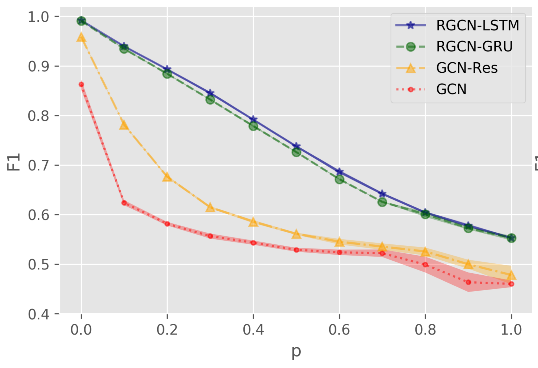


(b) GAT based models on Pubmed

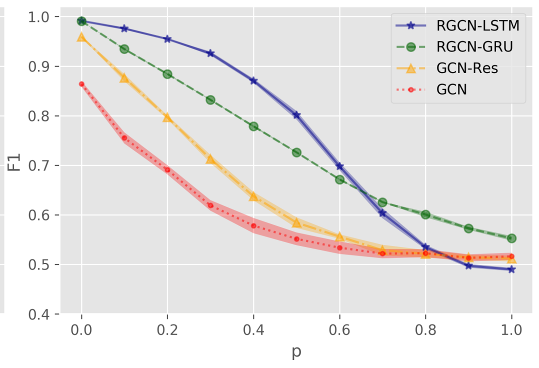
Figure 3: Influence of model depth (number of layers) on performance. Markers denote averaged micro-F1 scores on test dataset in 5 runs. Shaded areas represent standard deviations. We show results for RGNN with LSTM and GRU units, GNN with residual connections, and standard GNN models.

C. Perturbation Analysis for GCN

As shown in Figure 4a and 4b, GCN models with LSTM and GRU units can successfully avoid the noisy information from graph structures as well as nodes' local features. In the case of noisy feature, the performance of RGCN-LSTM is generally better than RGCN-GRU, but it decreases faster than RGCN-GRU in extreme cases ($p > 0.7$).



(a) GCN based models with noisy graphs



(b) GCN based models with noisy features

Figure 4: Perturbation analysis. Markers denote averaged micro-F1 scores on test graphs of PPI in 5 runs. Shaded areas represent standard deviations. We show results for RGNN with LSTM and GRU units, GNN with residual connections, and standard GNN models.