Bayesian Graph Neural Networks with Adaptive Connection Sampling: Supplementary Materials

Arman Hasanzadeh [*1] **Ehsan Hajiramezanali** [*1] **Shahin Boluki** [1] **Mingyuan Zhou** [2] **Nick Duffield** [1]
Krishna Narayanan [1] **Xiaoning Qian** [1]

In this supplement, we first provide an ablation study on local GDC. Dataset statistics, and further implementation details are also presented. Finally, schematics of different stochastic regularization techniques for GCNs are provided.

1. Ablation Study: Global versus Local

We further investigate our learnable GDC, in which for each edge at each layer a different connection sampling distribution is learned. We refer to this scenario as the *local* learnable GDC. This, indeed, is a more general case than learning a single distribution for all edges in a layer. Expanding the variational beta-Bernoulli GDC to local learnable GDC is straightforward. Note that the KL term in the loss function can be derived in the same manner as in the global learnable GDC – as described in Section 4 of the paper – except that it will include the sum of $\mathrm{num\_layers} \times \mathrm{num\_edges}$ terms as opposed to the $\mathrm{num\_layers}$ terms in the global GDC.

By training the aforementioned model on the citation datasets, we find that the accuracy degrades and the KL divergence reduces to zero for every choice of prior. This phenomenon, which is known as *posterior collapse* or *KL vanishing*, is a common problem in variational auto-encoders for language modeling (Bowman et al., 2015; Goyal et al., 2017; Liu et al., 2019). It is often due to over-parametrization in the model, which is indeed the case in the local learnable GDC. A solution to this issue could be making the parameters of the distribution dependent on the graph topology and/or node attributes. We leave this for future studies.

[*]Equal contribution [1]Electrical and Computer Engineering Department, Texas A&M University, College Station, Texas, USA [2]McCombs School of Business, The University of Texas at Austin, Austin, Texas, USA. Correspondence to: Arman Hasanzadeh <armanihm@tamu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Table 1. Graph dataset statistics.

Dataset	# Classes	# Nodes	# Edges	# Features
Cora	7	2,708	5,429	1,433
Citeseer	3	3,327	4,732	3,703
Cora-ML	7	2,995	8,416	2,879

2. Datasets and Implementation Details

All of the models are implemented in PyTorch (Paszke et al., 2017). All of the simulations are conducted on a single NVIDIA GeForce RTX 2080 GPU node. We evaluate our proposed methods, GCN-BBDE and GCN-BBGDC, and baselines on three standard citation network benchmark datasets. We preprocess and split the dataset as done in (Kipf & Welling, 2017) and (Bojchevski & Gunnemann, 2018). For Cora and Cora-ML, we use 140 nodes for training, 500 nodes for validation and 1000 nodes for testing. For Citeseer, we use 120 nodes for training and the same number of nodes as Cora for validation and testing. Table 1 provides the detailed statistics of the graph datasets used in our experiments. The warm-up factor used in GCN-BBGDC with more than 2 layers for Cora and Cora-ML is $\min(\{1, \mathrm{epoch}/20\})$, and for Citeseer is $\min(\{1, \mathrm{epoch}/40\})$. We have deployed Adam optimizer (Kingma & Ba, 2014) in all of our experiments.

3. GDC versus Other Stochastic Regularization Techniques

To further clarify the differences of our proposed GDC from existing stochastic regularization techniques, we draw the schematics of a GCN layer to which DropOut, DropEdge, Node Sampling, and our GDC are applied; shown in figures below. The input graph topology for the GCN layer is depicted in 1. The number of input and output features are both two in this toy example.

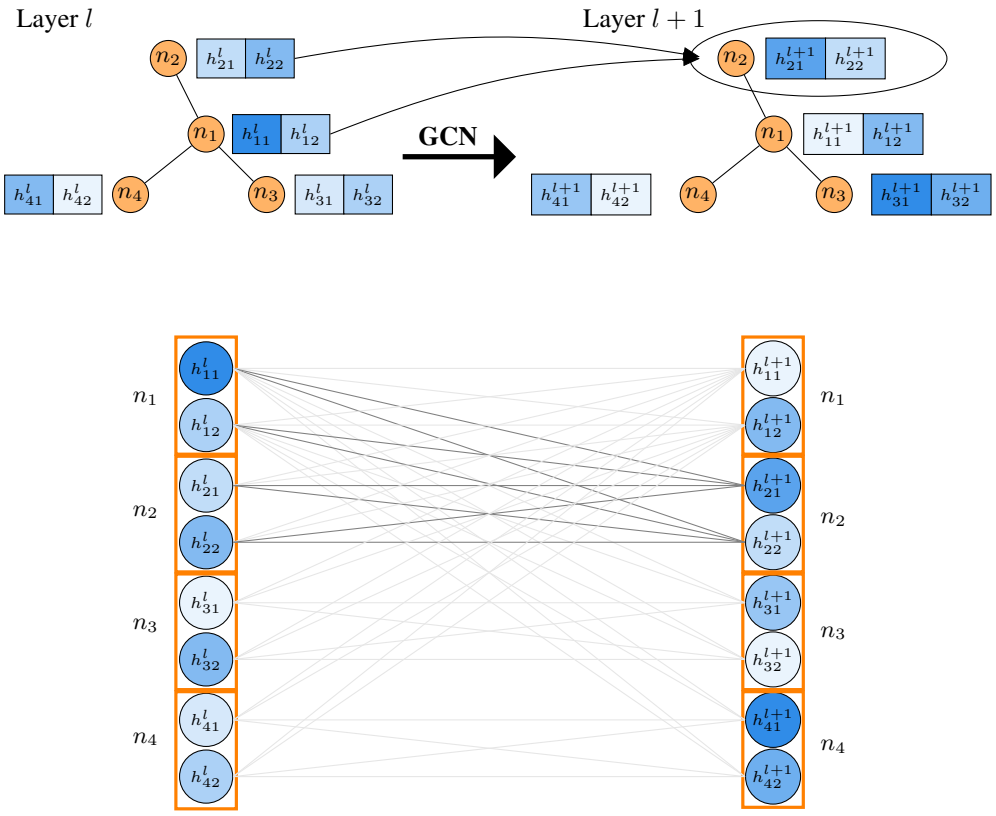

Figure 1. **Top**: Schematic of a GCN layer on a graph with 4 nodes. Number of both input and output features are two. The connections are localized as explicitly depicted for node 2. **Bottom**: The same GCN layer shown in a more conventional way, i.e. each layer is a vector of neurons or features. Each circle is a feature and each square represents a node. The connections are sparse and the sparsity is based on the input graph topology. The connections for node 2 in layer $l+1$ are highlighted.

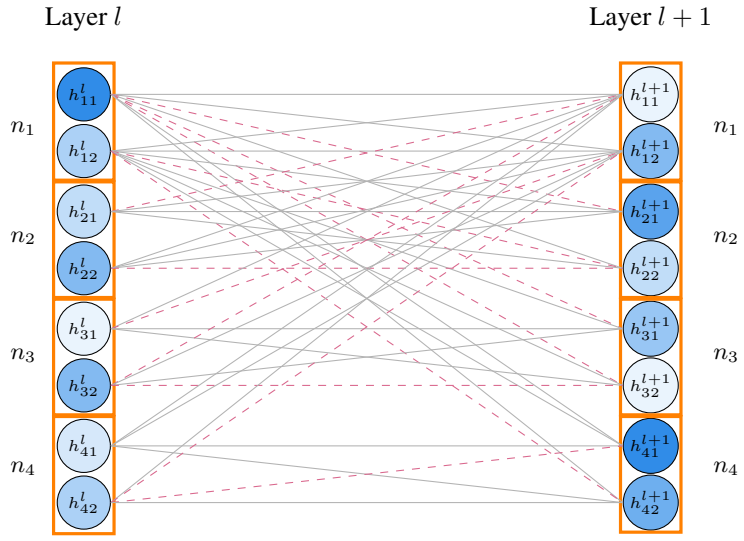

Figure 2. Schematic of our proposed GDC. Each circle is a feature and each square represents a node. GDC drops connections independently across layers. The dashed lines show dropped connections and the gray ones show the kept connections.

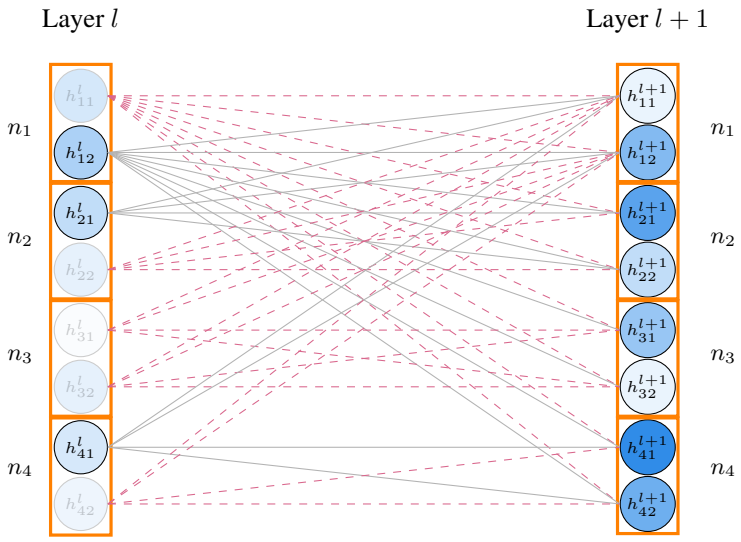

Figure 3. Schematic of DropOut (Srivastava et al., 2014). Each circle is a feature and each square represents a node. DropOut drops features at each layer. The faded circles represent dropped features while the other ones are kept. The dashed lines show dropped connections and the gray ones show the kept ones.

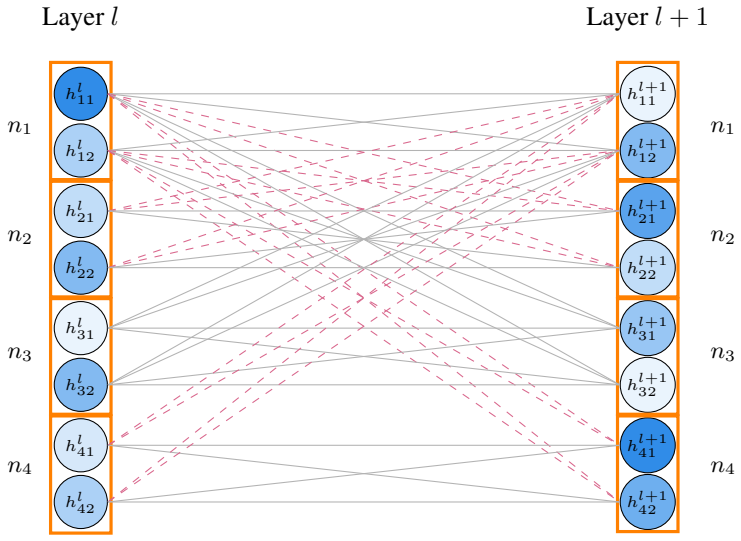

Figure 4. Schematic of DropEdge (Rong et al., 2019). Each circle is a feature and each square represents a node. DropEdge drops edges between nodes hence all of the connections between their corresponding channels are dropped. Note that the mask in DropEdge is symmetric. In this example, the edge between nodes 1 and 2 as well as the edge between nodes 1 and 4 are dropped. The dashed lines show dropped connections and the gray ones show the kept ones.

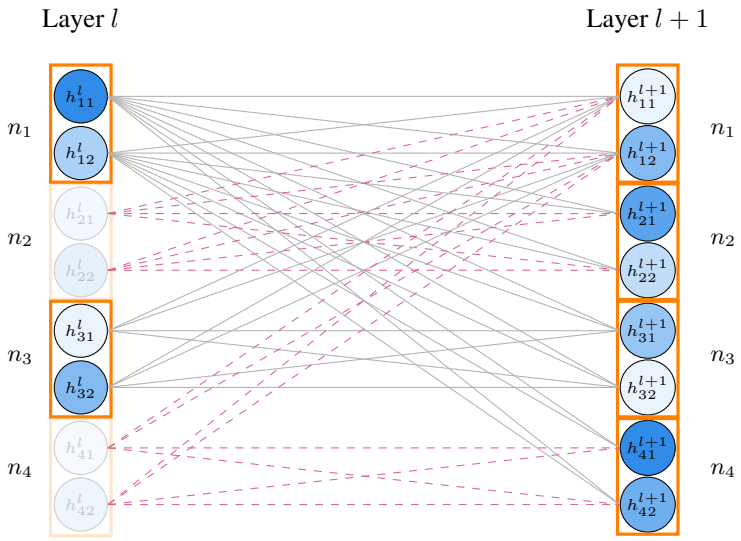

Figure 5. Schematic of the node sampling strategy in FastGCN (Chen et al., 2018). Each circle is a feature and each square represents a node. FastGCN drops nodes hence all of the connections to that node are dropped. The faded nodes represents the dropped nodes. The dashed lines show dropped connections and the gray ones show the kept ones.

References

Bojchevski, A. and Gunnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

Goyal, A. G. A. P., Sordoni, A., Côté, M.-A., Ke, N. R., and Bengio, Y. Z-forcing: Training stochastic recurrent networks. In *Advances in neural information processing systems*, pp. 6713–6723, 2017.

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

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Liu, X., Gao, J., Celikyilmaz, A., Carin, L., et al. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. *arXiv preprint arXiv:1903.10145*, 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Rong, Y., Huang, W., Xu, T., and Huang, J. DropEdge: Towards the very deep graph convolutional networks for node classification, 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.