TiDL-S22 | Team-18: Data Augmentation for Graph Neural Networks

Aryan Agarwal and **Shivansh Rakesh** and **KV Aditya Srivatsa**
International Institute of Information Technology, Hyderabad
{aryan.agarwal, shivansh.rakesh}@students.iiit.ac.in
k.v.aditya@research.iiit.ac.in

Abstract

Data augmentation can improve generalizability, can improve performance and reduces cost of gathering and labelling new training data. But data augmentation for graphs is less studied, primarily due to complex and non-Euclidean structure. The structure of graphs is encoded by node connectivity which is irregular, hence making it difficult to augment.

The paper provides a novel solution to this problem called GAUG. Given a graph G, the paper aims at generating a graph G' with undesirable edges pruned and required edges maintained or added. The augmented graph G' can then be utilized for downstream tasks such as edge prediction and node classification.

1 Introduction

The question here is, how to augment the graph data. When dealing with graphs, we have two options. Either to add/remove nodes or to add/remove edges.

Adding nodes poses challenges in labelling, inputting features and connecting new nodes for node classification. Removing nodes would cause loss of available data.

Thus, we are left with adding/removing edges.

Now, the question is, which edges should be modified. Some of the recent approaches for altering edges are:

- DropEdge: Randomly removes a fraction of graph edges before each training epoch, in an approach reminiscent of dropout (Rong et al., 2019)
- AdaEdge: Iteratively add (remove) edges between nodes predicted to have the same (different) labels with high confidence in the modified graph. (Chen et al., 2019)
- BGCN: Predictions of GCN to produce multiple denoised graphs, and ensembles results from multiple GCNs. BGCN also bears the risk of error propagation. (Zhang et al., 2019)

*	GNN	Air-Eur		PolBlogs	
		R	C	R	C
M	GCN	62.96	62.5	44.82	43.97
	GSAGE	61.44	61.03	41.7	40.16
	GAT	61.78	60.29	43.92	43.88
O	GCN	61.4	52.5	43.66	43.73
	GSAGE	60.98	52.03	41.15	41.06
	GAT	61.02	50.29	42.8	42.2

Table 1: Micro-F1 scores on new datasets

Neural edge predictors like GAE are able to latently learn closs-homophilic tendencies in existent edges that are improbable and nonexistent edges that are probable. GAUG, the method proposed in this paper makes use of this idea. It does strategic edge manipulation to promote intra-class edges and demote inter-class edges makes class differentiation in training trivial with a GNN, when done with label omniscience.

The motivation for this is the inconsistency of data and the non-trivial solution for graphs.

One of the challenges we faced is the dependency of edge probabilities on class homophily. Also, the numbers from our implementation varied from those in the paper.

2 Contributions

1. Reproducing results (from author's code base)
2. Recreating the code base and generating & comparing results
3. Training and evaluation on external datasets
4. Testing out hypotheses and novel ideas (GAugV & GAugT)

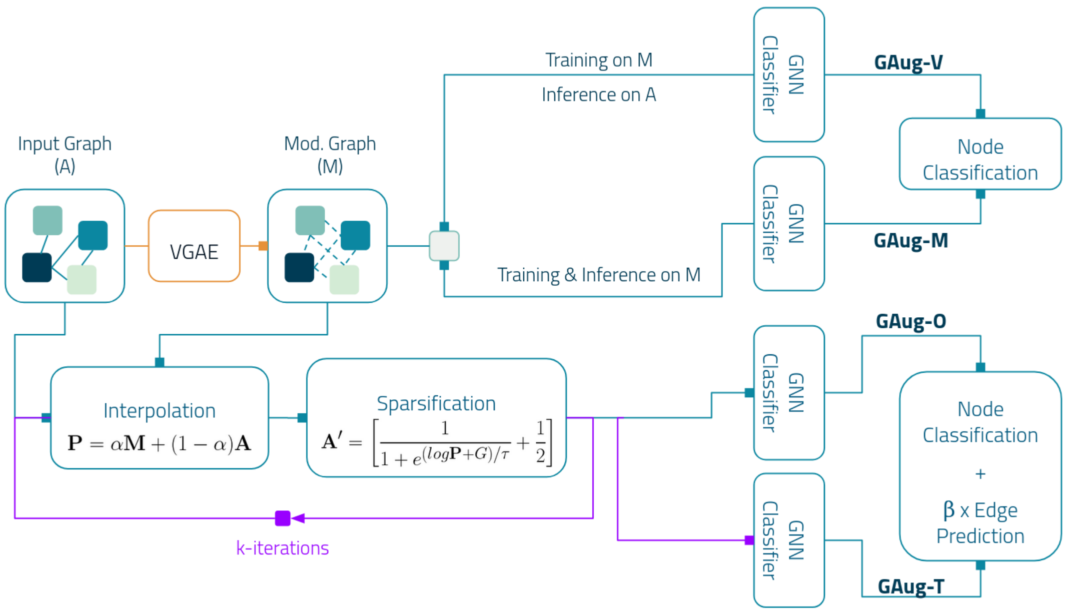
Input Graph (A)
VGAE
Mod. Graph (M)
Training on M
Inference on A
Training & Inference on M
GNN Classifier
GNN Classifier
GNN Classifier
GNN Classifier
GAug-V
GAug-M
GAug-O
GAug-T
Node Classification
Node Classification + β x Edge Prediction
Interpolation
$\mathbf{P} = \alpha\mathbf{M} + (1-\alpha)\mathbf{A}$
Sparsification
$\mathbf{A}' = \left\lfloor \frac{1}{1+e^{(log\mathbf{P}+G)/\tau}} + \frac{1}{2} \right\rfloor$
k-iterations

Figure 1: GAug network architecture. Original methods: GAugM GAugO. Novel methods: GAugV GAugT

3 Experiments

- The original paper's codebase was utilized to produce Micro-F1 scores for the various dataset-GNN pairs. Most scores were observed to be similar to the scores reported by the paper.

- To better understand the granular functioning of the original paper's approach as well as to have an easily editable code structure, the paper's augmentation approach was replicated using our own code base. The scores for all dataset-GNN pairs were generated and have been reported in Table 2.

- Both the original and the reconstructed codebases were expanded to accommodate additional datasets. We have reported scores for two datasets, namely: (1) Airport-Europe (Ribeiro et al., 2017), and (2) PolBlogs (Adamic and Glance, 2005) in Table 1.

4 Novel Approaches

4.1 GAugV

According to the authors, GAugM is best suitable for contexts wherein the graph is not subject to dynamic edge-manipulation. This is because the GAugM pipeline requires re-computation of the M-edge-matrix for each inference. For more dynamic setups, they prescribe the GAugO method. This leads to the hypothesis: *"GAugM must furnish poorer performance when run inference on by the raw graph itself instead of M"*. This suggests that an alternate setup **GAugV**, which is architecturally identical to GAugM and has been trained similarly, would perform subpar to GAugM & GAugO, when provided with the original graph A instead of the modified graph M during inference.

The results of this experimental setup have been covered in Table D. Most dataset-GNN pairs show that GAugV performs at par or worse off than GAugM as well as GAugO. This confirms our hypothesis and validates the need for GAugO for contexts with dynamic edge-manipulation which requires low-latency inference.

4.2 GAugT

GAugO supports a learnable, stochastic augmentation procedure. It uses the variational graph auto encoder (VGAE) to generate the modified adjacency matrix M, but unlike GAugM, interpolates it with the original graph A to prevent it from deviating arbitrarily. The resultant graph P is then sparsified using Bernoulli sampling (Maddison et al., 2016) (Jang et al., 2016). This procedure allows for the subsequent network to learn to actively denoise the input graphs, thus allowing us to pass the original (assumed to be

Method	**GNN Arch.**	**Cora**		**Citeseer**		**PPI**		**Flickr**		**Air-USA**	
		R	C	R	C	R	C	R	C	R	C
GAugM	GCN	82.7	82.38	73.27	72.28	46.56	46.57	68.24	67.98	61.4	60.02
	GSAGE	82.49	83.61	71.48	71.70	44.42	44.28	65.4	65.27	59.3	58.70
	GAT	82.21	82.9	69.59	69.52	45.9	45.36	61.45	61.62	58.59	57.53
GAugO	GCN	83.2	78.3	72.3	72.9	44.6	42.99	61.6	61.23	57.9	58.58
	GSAGE	81.6	77.8	71.3	65.7	45.1	43.73	44.3	50.91	55.2	52.1
	GAT	81.3	81.7	70.4	69.9	44.23	43.81	17.6	11.05	53.4	54.26

Table 2: Micro-F1 scores. R: Reproduced from author's code, C: Reconstructed code

Method	**GNN Arch.**	**Cora**		**Citeseer**		**PPI**		**Flickr**		**Air-USA**	
		C	T	C	T	C	T	C	T	C	T
GAugV	GCN	82.38	82.02	73.27	72.30	46.56	41.03	68.24	65.62	61.4	45.14
	GSAGE	83.61	81.20	71.48	71.40	44.42	41.22	65.4	63.29	59.3	46.58
	GAT	82.9	81.16	69.59	67.69	45.9	41.95	61.45	58.5	58.59	50.78
GAugT	GCN	78.3	73.6	72.9	65.9	42.99	**43.31**	61.23	58.37	58.58	57.9
	GSAGE	77.8	**79.3**	65.7	**67.2**	43.73	42.26	50.91	50.42	52.1	37.3
	GAT	81.7	80.6	69.9	68.9	43.81	**44.19**	11.05	**12.60**	54.26	51.02

Table 3: Micro-F1 scores. C: Old methods (M&O); T: New methods (V&T)

noisy) graph during inference to GaugO.

Our novel method, GAugT, attempts to strengthen this denoising step by passing the original graph A and the modified graph M through multiple nested layers of interpolation and sprasification. The results have been shown in Table E.

Several dataset-GNN pairs show a substantial improvement (emboldened in table), however many other pairs show a deterioration in performance. This non-monotonic behaviour suggetss that our method requires model specific hyperparameter tuning to increase the positive contribution of the nested denoising step.

5 Conclusion and Future Work

In this project, we explored the domain of graph data augmentation from the perspective of the methods offered by the paper: "Data Augmentation for Graph Neural Networks" (Zhao et al., 2020). Through the timeline of the project, we generated results from their codebase, created a codebase oof our own to reproduce their results, conducted a few experiments to expand their work as well as to test our hypotheses. We also proposed a novel variant of augmentation which outperformed the original paper's scores in several cases. Additionally, more experimentation can be performed at the autoencoding step as well as the denoising-sparsification steps.

References

Lada A. Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, LinkKDD '05, page 36–43, New York, NY, USA. Association for Computing Machinery.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2019. Measuring and relieving the oversmoothing problem for graph neural networks from the topological view. *CoRR*, abs/1909.03211.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables.

Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. 2017. istruc2vec/i. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. The truly deep graph convolutional networks for node classification. *CoRR*, abs/1907.10903.

Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. 2019. Bayesian graph convolutional neural networks for semi-supervised classification.

Proceedings of the AAAI Conference on Artificial Intelligence, 33(01):5829–5836.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2020. Data augmentation for graph neural networks.