Probabilistic Graph Residual

Karthik Sanka*
Department of Computer Science
North Carolina State University
Raleigh, USA
ksanka@ncsu.edu

Fin Amin*
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, USA
samin2@ncsu.edu, ORCID ID: 0000-0003-1212-4951

***Abstract*—Graph Neural Networks (GNNs) have been used for a plethora of applications ranging from image processing to recommendation/page rank algorithms. One of the key struggles of GNNs have been their fragility to abnormal features and noise. Recently, there has been intense interesting in exploring how the message passing interface of GNNs deals with abnormal features and adversarial attacks. In this work, we introduce a methodology which is allows us to more precisely detect and adapt residual connections in GNNs to aberrant node features.**

***Index Terms*—adaptive residual, adversarial graphs, abnormal features, noisy features, robust graph neural networks**

I. Introduction

Although graphs are mathematical objects and humans are organisms, they share a lot in common. As the old saying goes, *tell me 5 of your best friends, and I know everything I need to know about you.* Graph neural networks–the message passing interface specifically–depend on the homophilic assumption. What is remarkable is that the assumption that "like attracts like" has a semantic interpretation to how a GNN is effected by abnormal features. Namely, nodes in GNNs are very heavily influenced by their neighbors–in the same way humans are heavily influenced by those close to us.

The contribution of our work is a new kind of GNN residual connection which more precisely captures differences in between normal and abnormal features, *ProRes*. Specifically, if given some adversarial input embedding for node $p_n = x_n + e$, where e is some noise, how do we prevent this embedding from being propagated throughout a GNN? Conversely, suppose p_n was not adversarially generated–it is an abnormal node, how do we allow this genuine node to propagate as usual? We show that our work combines the best of existing work and ameliorates their issues. Furthermore, we show that our methodology works well for non-abnormal/non-adversarial GNN tasks. The convention of our paper is to call adversarially generated input embeddings as "adversarial nodes;" abnormal but genuine input embeddings as "abnormal/noisy nodes;" and we refer to either of the aforementioned as "aberrant" features. Lastly, "normal/clean features," refer to node embeddings which are not aberrant.

II. Prior Work and Observations

In this section, we detail the message passing interface of GNNs and explain how the feature aggregation step can be interpreted as a lowpass/smoothing operation. Furthermore, we give an overview of prior work on GNN residual connections and how these connections deal with aberrant features.

A. The Message Passing Interface of GNNs

Suppose that we have some graph, G with V nodes and E edges. Additionally, let us declare the adjacency matrix of G as A. As shown in [9], the message passing framework of GNNs is as follows:

$$x_n(t+1) = F_\theta(x_n(t), l_{neigh(n)}(t)). \tag{1}$$

Where $x_n(t)$ is the *embedding* of node n after messaging passing step, t. $l_{neigh(n)}(t)$, is the *message* received from the neighbors of node n. The embedding of a node is how the GNN represents the semantic properties of a node in terms of d features. Therefore, $x_n \in \mathbb{R}^{1\times d}$; we may stack all the node embeddings of the graph to produce $X(t) \in \mathbb{R}^{V\times d}$. The message is some function of the labels/embeddings, of the neighboring nodes/edges of n. F is some aggregation function which combines the current embedding of x_n with $l_{neigh(n)}$ to update x_n.

Our work is concerned with $P \in \mathbb{R}^{V\times d}$. Where P represents the input embeddings of the nodes, ie $x_n(0) = P(n)$ for $\forall n$. Namely, what do we do if some of the input embeddings of in P are aberrant?

B. The Laplacian Smoothing Operation

The graph convolution operation which aggregates features can be thought of as a special form of Laplacian smoothing operation [3] [4]. This smoothing operation is essentially a weighted average of the features from the neighbouring nodes. Due to this phenomenon, graph convolution can act as a low pass filter and can smooth abnormal nodes which are usually high frequency signals [6]. Stacking many such layers can be problematic and lead to oversmoothing which can produce undesirable results. This motivates residual connections as they preserve original node features.

C. Approximate Personalized Propagation of Neural Predictions (APPNP)

APPNP [2] approaches the over smoothing problem caused by GNNs by introducing a residual connection. They define their message passing framework as:

$$X(t+1) = (1-\alpha)AX(t) + \alpha F_\theta(P) \tag{2}$$

* denotes equal contribution.

Where α is a user defined parameter that controls how much of the original input embedding to retain. Note that α is a scalar broadcasted to $\mathbb{R}^{V\times 1}$. As observed in [6], a key drawback of this approach is that all nodes have the same α value; in other words, APPNP is not node-wise adaptive. This is an issue because if we know that a certain node has an adversarial input embedding, then it is not possible to weigh that specific node with less weight using APPNP's approach. The residual connection might prevent aberrant nodes from getting smoothed out. One final remark is that APPNP trains on learning the input embedding from P, rather than learning a function for the overall aggregation step.

D. Adaptive Residual GNN (AirGNN)

AirGNN [6] aims to overcome this issue by using a node-wise adaptive residual connection. Their message passing frame work is:

$$Y(t) = (1 - 2\gamma(1-\lambda))X(t) + 2\gamma(1-\lambda))AX(t) \quad (3)$$

$$\beta_n = max(1 - \frac{\gamma\lambda}{||y(t)_n - p_n||_2}, 0) \quad (4)$$

$$x(t+1)_n = (1-\beta_n)p_n + \beta_n y_n(t), \forall n \in V \quad (5)$$

$$\gamma = \frac{1}{2(1-\lambda)} \quad (6)$$

The key improvement AirGNN makes over APPNP is adding node-wise granularity at the aggregation function. The β vector trains itself to weigh aberrant nodes with less importance at the aggregation step (5). Note that there is a user defined parameter λ which controls how accepting AirGNN is of aberrant features. A lower value of λ makes AirGNN more aggressively reject aberrant features.

III. Probabilistic Graph Residual (ProRes)

Our work makes two observations, APPNP performs better than AirGNN in the normal setting, this is likely due to APPNP's more generalized message passing framework. APPNP also uses an additional hyperparameter α to balance the strength of the aggregated features and the residual connection in the message passing step. The second observation is on the aggregation step and computation of β within AirGNN. Namely, their β proportional to the inverse of the standard deviation of our input feature and the aggregate features[1]. In our work, we surmise that there is a better way to quantify differences between node embeddings for the purpose of aberrant embedding detection. In this section, we present Probabilistic Graph Residual (ProRes).

Our work predicts the nature of data, i.e clean or aberrant by computing the KL divergence between the node features and the aggregated features. Afterwards, we use the best of APPNP's and AirGNN's message passing frameworks. ProRes' message passing frame work is as follows:

$$w_{agg} = \begin{cases} 0.1 & \text{if } scaled\_KL < \epsilon \\ 0.01 & \text{otherwise} \end{cases} \quad (7)$$

$$x(t+1)_n = w_{agg_n} p_n + (1 - w_{agg_n}) y(t)_n \quad (8)$$

Where $scaled\_KL$ is defined as:

$$u = SoftMax(x_n), v = SoftMax(y_n) \quad (9)$$

$$kl = KLDiv(u, v) \quad (10)$$

$$scaled\_KL = (1 - \exp(-kl)) \quad (11)$$

And Y being defined identically to AirGNN.

w_{agg_n} is used as a parameter to control the strength of residual connection and the aggregation (equivalent to Laplacian smoothing). ϵ is a user defined parameter; a lower value of ϵ makes ProRes more aggressively reject aberrant features. We found $\epsilon = 0.1$ using empirical techniques and we use this value for all experiments. Note that w_{agg_n} is node wise adaptive as compared to α in APPNP [2]. In the overall message passing scheme, the KL divergence replaces the l_{21} norm used in AirGNN [6].

The hemophiliac assumption made by GNNs make them increasingly sensitive to outliers. For this reason, being able to quantify distributional differences between nodes is critical. This is why the KL divergence is a more suitable choice for computing the difference between the input features and aggregated features as opposed to vector norms.

IV. Experiments

In this section, we give our experimental setup and explain our experiments. We conducted all experiments on an RTX 3080 16GB laptop GPU with 32GB of system memory and an Intel i7 11800H. We run three experiments on three citation datasets:

The datasets used are:

1) The Cora [7] dataset contains 2708 publications (nodes), 5429 citation links (edges). Each publication has a multi-hot bag of words vector of length 1433[2] (features). Each publication belongs to one of seven categories (labels).
2) The CiteSeer [8] dataset consists of 3312 publications (nodes) with 4732 citation links (edges) and, like Cora, has a multi-hot bag of words vector of length 3703 (features). Each publication is classified into one of six classes (labels)
3) The PubMed [1] dataset contains 19717 publications (nodes), 44338 citation links (edges). Each publication has an TF/IDF weighted word vector (features) of length 500 which corresponds to a dictionary of the same length. Each publication is classified in one of three classes (labels).

[1]When analyzing the denominator of (4), they use the l_{21}. This can be interpreted a standard deviation measure

[2]Each index of this vector represents a word from the dictionary. If the word is present in the publication, the corresponding index is set to 1, otherwise that index is set to 0.

To test our methodology[3], we needed a way to simulate adversarial attacks on graphs. To produce an "apples to apples" comparison, we use the DeepRobust [5] library–the same library used by the authors of AirGNN. Moreover, we use the suggested architectures of AirGNN and APPNP in their respective papers during our comparisons[4]. To simulate abnormal features, we add multi-variate Gaussian noise with unit variance to the a percentage of the node features. All experiments are repeated 10 times, the averages are shown.

To summarize, we run the following experiments:

1) We test the performance on adversarial nodes by using the DeepRobust library. We use this library to poison $\{0, 1, 2, 5, 10, 20, 50, 80\}$ nodes. Unless otherwise stated, each implementation was trained for 1000 epochs–as the authors do in AirGNN [6]. For consistency, we select 40 nodes which are unpoisoned and compute the accuracy on them. We use the same 40 nodes for all methodologies to test adversarial performance. We set $\lambda = 0.1$ for AirGNN.
2) We test abnormal features by randomly sampling a percentage of the nodes and adding Gaussian noise to those features. Accuracy is computed on the entire dataset after training for 1000 epochs. We set $\lambda = 0.9$ for AirGNN.
3) We give our accuracy on the three datasets without any perturbations or etc after training on it for 1000 epochs. We refer to this as the clean setting. We set $\lambda = 0.5$ for AirGNN.

When we originally ran our experiments using the value of λ described in the AirGNN's paper ($\lambda = 0.5$), AirGNN performed very poorly. We experimented with various values of λ to show AirGNN in the best possible light. The following results are with the best values we found for λ for each experiment. For APPNP, we used $\alpha = 0.2$ as suggested by their paper and for ProRes, we set $\epsilon = 0.1$.

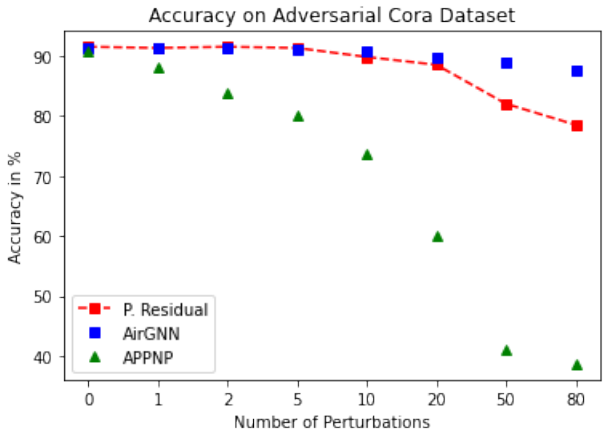

Fig. 1. The accuracy on 40 nodes of the Cora dataset with a number of nodes being adversarially perturbed. Here, ProRes outperforms all other implementations when the number of perturbations is ≤ 5. Our performance degrades at roughly 50 perturbations.

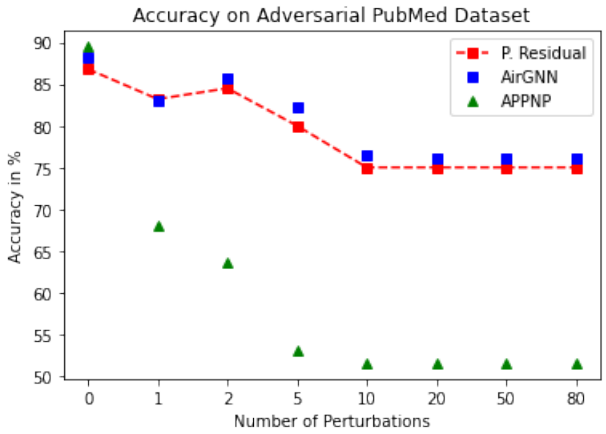

Fig. 2. The accuracy on 40 nodes of the PubMed dataset with a number of nodes being adversarially perturbed. We perform nearly identically to AirGNN.

V. Results and Conclusion

A. Adversarial Features

According to figures 1, 2 3, our implementation performs competitively. In fact, in Cora, our implementation outperforms all others as long as there are less than or equal to 5 perturbations. Our performance degrades significantly at 50 or greater perturbations. We surmise that this is due to our model beginning to erroneously believe that these adversarial nodes are part of the distribution of embeddings. Therefore, during our aggregation step, they are not given low weight.

When analyzing performance on CiteSeer in the adversarial setting (figure 3), our model performs marginally worse than AirGNN. We suspect that this is due to overfitting on the dataset due to training for 1000 epochs. More precisely, we presume that our model erroneously learns that the adversarial features are genuine parts of the distribution of neighborhood features. Interestingly, this phenomenon does not occur with PubMed (figure 2). We surmise that this is because the PubMed dataset is *considerably* larger than CiteSeer.

To summarize, in all of the tested datasets, our methodology performs well compared to AirGNN and significantly outperforms APPNP.

B. Abnormal Features

This section compares the performance on abnormal features. As stated in the experimental set up section, we test this by adding multi-variate Gaussian noise with unit variance to the node features of a percentage of nodes.

In this experiment, our model significantly outperforms APPNP once more. When compared to AirGNN, it performs comparably on Cora (figure 4 and very competitively on CiteSeer (figure 5). When analyzing results on PubMed (figure 6), our model performs considerably worse once the percent-

[3]Our work can be found on GitHub. The repository can be found on https://github.com/FinAminToastCrunch

[4]We set our message passing steps to $K = 10$, a drop out of 0.8, a learning rate of 0.01 and weight decay of 0.0005. [6] [2]

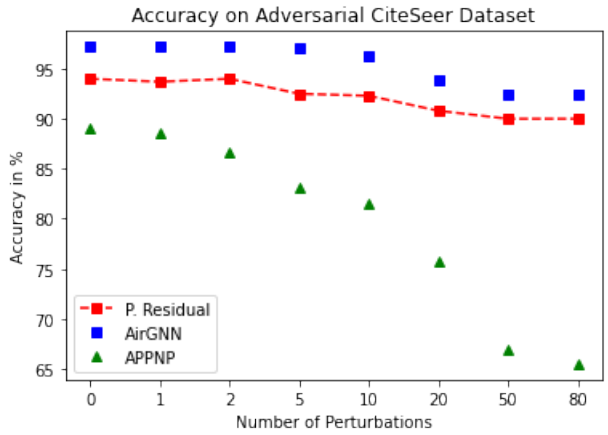

Fig. 3. The accuracy on 40 nodes of the CiteSeer dataset with a number of nodes being adversarially perturbed.

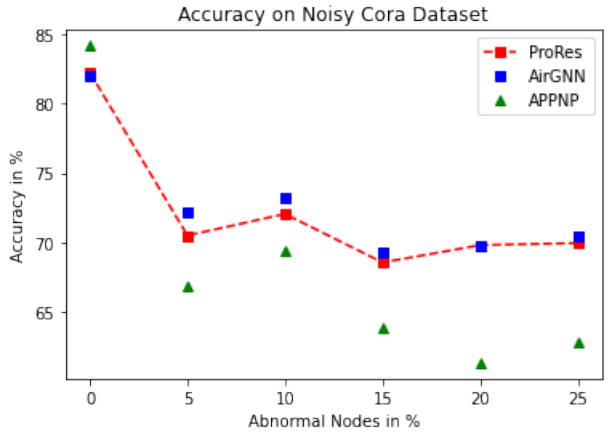

Fig. 4. Accuracy on all of Cora with a % of the nodes abnormal. Note that the performance of AirGNN and ProRes *increases* once $20\% \geq$ of the nodes are abnormal.

age of abnormal nodes $\geq 5\%$. *Why isn't our performance better?* We surmise that this is more due to how we generated abnormal nodes rather than our methodology being genuinely poor performing. AirGNN's l_{21} loss in equation (4) makes it particularly well suited to dealing with nodes with additive Gaussian noise. The λ hyperparameter can be tuned to accommodate this specific level of noise. Because ProRes takes a probabilistic approach, it has trouble deciphering the added noise as being aberrant rather than adversarial. Evidence of this phenomenon can be seen when noticing that the accuracy of Pro-Res *improves* once $20\% \geq$ of the nodes in PubMed (figure 6) and Cora (figure 4) are abnormal. In other words, we suspect that ProRes begins to believe the abnormal nodes are actually normal and part of the original distribution of the data. We suspect that a similar explanation can be given as to why AirGNN also increases performance $20\% \geq$ are abnormal in the aforementioned scenarios.

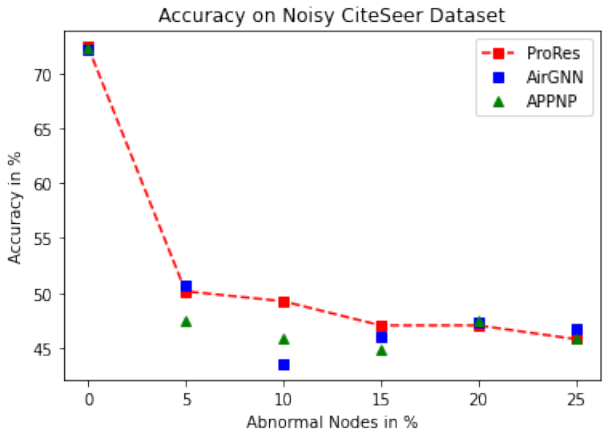

Fig. 5. Accuracy on all of CiteSeer with a % of the nodes abnormal.

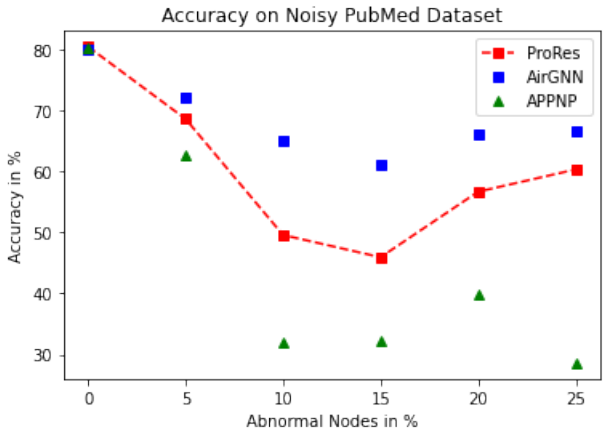

Fig. 6. Accuracy on all of PubMed with a % of the nodes abnormal. Note that the performance of AirGNN and ProRes *increases* once $20\% \geq$ of the nodes are abnormal.

C. Clean Features

To show that ProRes is effective in the clean setting, we give accuracy metrics on the entirety of the three datasets.

According to figure 7, we see that our implementation is useful for ordinary features. This is critical because in a real world use case, we would not always know if we expect abnormal nodes, so it is good to have an implementation which works well in both cases.

D. Conclusion

Future work can be done in improving how the difference in probability distributions are calculated and modeled. Future horizons may exist in using diffusion models [11] to model the distributions better. Discriminators [10] repurposed from a GAN trained on the dataset or learning discriminitive reconstruction procedures [12] can be used to detect out of distribution samples more effectively. In conclusion, we have introduced a new approach to adaptive residual connections in graph neural networks. Our methodology differs from existing work in that it has an adaptive residual which performs

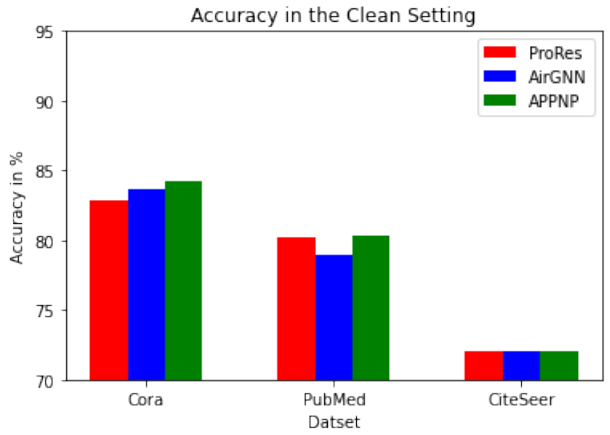

Fig. 7. The accuracy on all datasets without any perturbations or noise added to the features.

aggregation based on differences in probability distributions. We hope that our work will motivate future research in dealing with aberrant nodes.

REFERENCES

[1] Lise Getoor Galileo Mark Namata, Ben London and Bert Huang. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, Edinburgh, Scotland, 2012.

[2] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Personalized embedding propagation: Combining neural networks on graphs with personalized pagerank. *CoRR*, abs/1810.05997, 2018.

[3] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[4] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning, 2018.

[5] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

[6] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. Graph neural networks with adaptive residual. *Advances in Neural Information Processing Systems*, 34:9720–9733, 2021.

[7] Mustafa Bilgic Lise Getoor Brian Gallagher Prithviraj Sen, Galileo Mark Namata and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[8] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[9] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[10] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017.

[11] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015.

[12] Kai Tian, Shuigeng Zhou, Jianping Fan, and Jihong Guan. Learning competitive and discriminative reconstructions for anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5167–5174, 2019.