Enhancing Multi-hop Connectivity for Graph Convolutional Networks

Anonymous Authors[1]

Abstract

Graph Convolutional Network and many of its variants are known to suffer from the dilemma between model depth and over-smoothing issues. Stacking layers of GCN usually lead to the exponential expansion of the receptive field (i.e., high-order neighbors). In order to smooth the local neighborhood into informative representation without drastically increasing the number of graph convolution layers, we propose a simple and effective pre-processing technique to increase graph connectivity. Our approach selectively inserts connections between center nodes and informative high-order neighbors, with learnable weights to control the information flow through the connection. Experiments show that our approach improves the performance of GCN, and reduce the depth of GCNII without sacrificing its performance. Besides, our proposed homophily-based weight assignment can mitigate the effect of graph structural attacks.

1. Introduction

Graph Neural Networks (GNNs) have achieved great success in various tasks conducting graph data, including recommendation systems (Ying et al., 2018), drug discovery (Shi et al., 2020), and traffic prediction (Guo et al., 2019). One important factor behind the success is that that GNNs exploit message passing mechanism to aggregate information from nodes as well as their neighborhoods. For example, Graph Convolutional Network (GCN) (Kipf & Welling, 2017), one of the most popular GNN variants, defines spectral graph convolution to aggregate the information from multi-hop neighborhoods. However, deep GNNs suffer from over-smoothing (Li et al., 2018), indicating that the node representation converges to a particular vector as the number of layers goes large. Besides, (Kipf & Welling, 2017) pointed out that overfitting can also become an issue as the number of parameters increases with model depth.

Recently, GCNII (Chen et al., 2020) constructs a residual connection from the input layer to preserve the locality of node representation, thus addressing over-smoothing. However, GCNII requires great depth to achieve competitive performance, which significantly increases computational complexity. Considering the improvement and the price, they aggregate information from high-order neighbors in a lazy manner, which is not efficient. Specifically, 32/64-layer GCNII aggregates the information from almost the entire graph without filtering, which includes unnecessary information from high-order neighbors for the center node.

In this work, instead of using spectral graph convolution defined by GCN, we leverage node features in addition to graph structure for the convolutional filter formulation. We propose an approach to sample high-order neighbors and assign high weights to informative high-order neighbors to form a new filter. Specifically, we leverage random walk to efficiently sample potential high-order neighbors. Among the sampled high-order neighbors, weights are assigned using our proposed homophily-based loss function, which evaluates the importance of selected neighbors. By introducing high-order neighbors, GCN enhanced with our proposed approach shows considerable improvements in performance on the semi-supervised node classification task and we can can reduce the required depth of GCNII. In addition, Our proposed homophily-based loss function controls the information flow, which can serve as a potential defense technique against graph structure attacks. Adversarial attacks (Zügner & Günnemann, 2019a) on graph structure usually rewire the graph topology and aim to change the information flow, thus degrading the GNN's performance. By reassigning weights to the edges with our method, we can guide the flow of information back to informative neighbors and mitigate the adverse effect of graph structure attacks. We compare the performance with other state-of-the-art GNN defense models. Although our method does not target GNN structure attacks, our method still shows comparable or better performance, which demonstrates that graph homophily can be a key factor in designing GNN defense models.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022. Do not distribute.

2. Preliminary

2.1. Notations

Let $G = (V, E)$ denote the graph, where V is the set of nodes $\{v_1, \cdots, v_N\}$ with $|V| = N$ and E is the set of edges. The adjacency matrix is defined as $A \in \{0,1\}^{N\times N}$, and $A_{ij} = 1$ if and only if $(v_i, v_j) \in E$. Let $\mathcal{N}_i = \{v_j | A_{ij} = 1\}$ denotes the neighborhood of node v_i and D denote the diagonal degree matrix, where $D_{ii} = \sum_{j=1}^{n} A_{ij}$. The feature matrix is denoted as $X \in \mathbb{R}^{N\times F}$ where each node v is associated with a F-dimensional feature vector X_v. $Y \in \{0,1\}^{N\times C}$ denote the one-hot label matrix, where $Y_i \in \{0,1\}^C$ is a one-hot vector and $\sum_{j=1}^{C} Y_{ij} = 1$ for any $v_i \in V$.

2.2. Vanilla GCN and GCNII

The graph convolution layer of Graph Convolution Network (GCN) (Kipf & Welling, 2017) can be written as:

$$H^{(l)} = f\left(H^{(l-1)}, A\right) = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l-1)}\Theta^{(l)}\right),$$

where σ denotes the non-linear activation function and $\Theta^{(l)}$ is the parameter of l-th layer. Accordingly, the l-th layer of GCNII can be written as follows:

$$H^{(l+1)} = \sigma\left(\left((1-\alpha_l)\,\tilde{P}H^{(l)} + \alpha_l H^{(0)}\right)\left((1-\beta_l)\,I + \beta_l\Theta^{(l)}\right)\right), \tag{1}$$

where $\tilde{P} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, α_l and β_l control the extent that the input feature $H^{(0)}$ and $l-1$th layer's node representations $H^{(l-1)}$ mix with $H^{(l)}$ respectively.

3. Expanding High-order Neighbors for Spectral Graph Convolution Methods

The architecture of graph convolution networks dictates that only after k layers of graph convolution, the k-order neighbors' node representations can be embedded into the final representations for each node. This property leads to a dilemma in the choice of model depth: model with too few layers restricts the receptive field; model with too many layers dilute the influence of important low-order neighbors since the node representations could become indistinguishable after multiple graph convolution layers, and overfitting can become an issue as the number of parameters increases with model depth (Kipf & Welling, 2017).

To solve the above dilemma, we aim to introduce high-order neighbor information into the final node representation without increasing the depth of the model. Our method is to increase graph connectivity by adding edges to high-order informative neighbors for each node. Adding such edges can enlarge the receptive field. High-order neighbors will be sampled from random walk and reweighted by our proposed loss function. The sampling, reweighting, and training approaches will be introduced in Section 3.1, 3.2, and 3.3 respectively.

Formally, we define the formulation of the modified GCN's l-th layer as

$$H^{(l)} = f\left(H^{(l-1)}, A'\right) = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}'\tilde{D}^{-\frac{1}{2}}H^{(l-1)}\Theta^{(l)}\right), \tag{2}$$

where $A' = A + Q$ and Q contains our inserted edges connecting high-order neighbors.

3.1. Sampling High-order Neighbors with Random Walk

The most straightforward way of embedding high-order neighbor information into final node representation is to establish connectivity between them with induced edges. However, directly connecting all the high-order neighbors in the adjacency matrix will greatly increase the graph density, and thus increase the computation time of the graph convolution operation. To avoid significant changes in graph structure as well as high computational complexity, we exploit random walk to sample high-order neighbors for each node.

3.2. Reweighting High-order Neighbors

In our observation, the information of some nodes in the high-order neighborhood overlaps with the local neighborhood to a certain extent. Our aim is to select these high-order nodes to help smooth the local neighborhood into informative representation (Zeng et al., 2021). Accordingly, we present an unsupervised Lasso (Tibshirani, 1996) approach to catch these nodes and assign higher weights automatically through minimizing the feature loss between the first-order and high-order neighbors as follows:

$$\mathcal{L} = \sum_{i=1}^{n}\mathcal{L}_i = \sum_{i=1}^{n}\left(\left\|\sum_{j=1}^{n}\left(x_j - \left(\tilde{P}X\right)_i\right)Q_{ij}\right\|_2^2 + \lambda\sum_{j=1}^{n}|Q_{ij}|\right), \tag{3}$$

In the loss function $\mathcal{L}$, for each target node i and its high-order neighbor j, $(\tilde{P}X)_i$ is the node representation of node i after a graph convolution layer. When minimizing $\mathcal{L}_i$ on weight vector Q, we aim to assign larger weights to the high-order neighbors whose feature vector has a small distance with target node representation $(\tilde{P}X)_i$. $\lambda\sum_{j=1}^{n}|Q_{ij}$ is the penalty factor to limit the magnitude of Q_{ij}, and λ is the parameter that controls the strength of the penalty, where the larger the penalty value of λ, the greater the amount of shrinkage. When λ goes to infinity, t becomes 0 and all coefficients shrink to 0. More specifically, the Lagrangian form $\mathcal{L}_i$ can be rewritten as follows (Breiman, 1995):

$$\underset{Q}{\text{minimize}} \left\| \sum_{j=1}^{n} \left(x_j - \left(\tilde{P}X \right)_i \right) Q_{ij} \right\|_2^2, \quad \text{subject to } Q_{ij} \geq 0, \sum_{j=1}^{n} Q_{ij} = \alpha_i \tag{4}$$

where α_i is the sum of the scale coefficients for high-order neighbors of node i in Q. Considering that the feature vector of high-order neighbors may be different from that of the first-order neighbors, it is necessary to perform scale transformation for the high-order neighbors. Given the scale of the aggregated feature vector of first-order neighbors, we can naturally leverage it to control the scale of the feature vector for high-order neighbors. To this end, $\mathcal{L}_i$ has the following form:

$$\begin{cases} \mathcal{L}_i = \left\| \sum_{j=1}^{n} \left(x_j - \left(\tilde{P}X \right)_i \right) Q_{ij} \right\|_2^2 \\ \sum_{j=1}^{n} Q_{ij} = \sum_{j=1}^{n} (D_\alpha Q)_{ij} \\ Q_{ij} \geq 0 \end{cases} \tag{5}$$

Note that we can optimize all $\mathcal{L}_i$ in parallel.

3.3. Training Framework

In the process of random walk sampling, our method generates different random graph structures from the input data and can serve as a topology-level augmentation skill for graphs. Inspired by the application of consistency training on unsupervised data augmentation, we naturally exploit consistency regularization loss function (Sajjadi et al., 2016) for our supervised learning. Given K augmented graph structure matrix for each training epoch, we minimize the loss function as follows:

$$\mathcal{L} = \mathcal{L}_s + \beta \mathcal{L}_c, \tag{6}$$

where

$$\mathcal{L}_s = -\frac{1}{K} \sum_{k=1}^{K} \sum_{i \in \mathbf{T}} Y_i \log Z_i^{(k)}, \tag{7}$$

and

$$\mathcal{L}_c = \frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{N} \left\| \overline{Z}_i' - Z_i^{(k)} \right\|_2^2. \tag{8}$$

Here, $Z_i^{(k)}$ is the output of GCNs on the k-th augmented structure matrix. Following the sharpening trick (Berthelot et al., 2019), we compute $\overline{Z}_i = \frac{1}{K} \sum_{k=1}^{K} Z_i^{(k)}$ and $\overline{Z}_i' = \overline{Z}_i^{\frac{1}{T}} / \sum_{c=1}^{C} Z_{ic}^{\frac{1}{T}}$.

Table 1: Classification results on fixed split (%)

Method	**Cora**	**Citeseer**	**Pubmed**
GCN	81.5	70.3	79.0
GAT	83.0	72.5	79.0
APPNP	83.8	71.6	79.7
JKNet	81.1	69.8	78.1
DropEdge-GCN	82.8	72.3	79.6
Ours (GCN)	**83.9**	**74.0**	**79.9**

Table 2: Summary of classification accuracy (%) results with various depths.

Dataset	Method	Layers 2	4	8	16	32	64
Cora	GCN	81.1	80.4	69.5	64.9	60.3	28.7
	GCN(Drop)	82.8	82.0	75.8	75.7	62.5	49.5
	JKNet	-	80.2	80.7	80.2	81.1	71.5
	JKNet(Drop)	-	83.3	82.6	83.0	82.5	83.2
	Incep	-	77.6	76.5	81.7	81.7	80.0
	Incep(Drop)	-	82.9	82.5	83.1	83.1	83.5
	GCNII	82.2	82.6	84.2	84.6	**85.4**	**85.5**
	GCNII*	80.2	82.3	82.8	83.5	84.9	85.3
	Ours (GCNII)	**84.0**	**84.0**	**85.2**	**85.3**	85.1	83.9
Citeseer	GCN	70.8	67.6	30.2	18.3	25.0	20.0
	GCN(Drop)	**72.3**	70.6	61.4	57.2	41.6	34.4
	JKNet	-	68.7	67.7	69.8	68.2	63.4
	JKNet(Drop)	-	72.6	**71.8**	72.6	70.8	72.2
	Incep	-	69.3	68.4	70.2	68.0	67.5
	Incep(Drop)	-	**72.7**	71.4	72.5	72.6	71.0
	GCNII	68.2	68.9	70.6	72.9	73.4	**73.4**
	GCNII*	66.1	67.9	70.6	72.0	73.2	73.1
	Ours (GCNII)	70.7	70.4	71.5	**73.9**	**73.7**	**73.4**
Pubmed	GCN	79.0	76.5	61.2	40.9	22.4	35.3
	GCN(Drop)	**79.6**	79.4	78.1	78.5	77.0	61.5
	JKNet	-	78.0	78.1	72.6	72.4	74.5
	JKNet(Drop)	-	78.7	78.7	79.1	**79.2**	78.9
	Incep	-	77.7	77.9	74.9	OOM	OOM
	Incep(Drop)	-	**79.5**	78.6	79.0	OOM	OOM
	GCNII	78.2	78.8	**79.3**	80.2	79.8	79.7
	GCNII*	77.7	78.2	78.8	**80.3**	79.8	**80.1**
	Ours (GCNII)	78.0	77.8	78.9	80.0	OOM	OOM

4. Experiments

In this section, we evaluate the performance of our proposed method on semi-supervised node classification tasks. To show the robustness of our method, we also compare our method with other state-of-the-art defense methods against graph adversarial attacks.

Dataset We exploit three citation network datasets Cora, Citeseer, and Pubmed (Sen et al., 2008) for semi-supervised node classification and graph defense evaluation. For graph defense evaluation, we follow (Zügner & Günnemann, 2019a; Zügner et al., 2018) and only consider the largest connected component (LCC).

4.1. Semi-supervised Node Classification

Settings, Baselines, and Results We exploit the standard fixed splits (Yang et al., 2016) on these datasets, with 20 nodes per class for training, 500 nodes per class for validation, and 1000 nodes for testing. For baselines, we include GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), APPNP (Klicpera et al., 2019), DropEdge (Rong et al., 2019), and JKNet (Xu et al., 2018). Table 1 reports the mean classification accuracy on the test nodes after 100 runs. We reuse the metrics reported in the corresponding papers. Our method improves GCN by a margin of 2.4%, 3.7%, and 0.6% on Cora, Citeseer, and Pubmed respectively. And our method achieves the best on these datasets, outperforming multihop baselines such as APPNP and JKnet, which suggests we can use high-order neighbors to obtain better performance without increasing the depth of the graph neural networks.

4.2. Model Depth Analysis for GCN

In order to prove that our method can achieve good performance without increasing the number of layers, we have done some experiments on GCNII in terms of model depth. Table 2 reports the results for various models with different depth and we reuse the results from DropEdge and GCNII (Chen et al., 2020). We can observe that when our method is combined with shallow-depth GCNII, it can get better results on cora and citeseer. Specifically, our method gets 85.2% accuracy on cora with only 8 layers, and GCNII needs 32 layers to get above 85% accuracy. Besides, our model achieves 73.9% accuracy on citeseer with 16 layers, which is better than all results of GCNII. Although our model does not perform well on pubmed, we can still achieve similar performance to GCNII at 16 layers. These results demonstrate that our method can indeed achieve very good performance in the case of a low number of layers, which can reduce the complexity of the model. In addition, our method can effectively reduce the number of layers of the deep model, which suggests we need to design a better graph convolutional layer to effectively aggregate information from high-order neighbors and we just provide a strategy.

4.3. Defense Performance

In this section, we evaluate the defense performance of our selection algorithm in Section 3.2. Note that we exploit the homophily property of graphs to defend against the attack on graph structures.

Settings, Baselines, and Results The performance of our method is compared with other baseline methods against Meta-attack (Zügner & Günnemann, 2019a). Meta-attack treats the adjacency matrix as a hyperparameter and utilizes meta-learning to learn the best poisoning strategy. For the settings of our method, we set hidden units to 32 and keep other hyper-parameters unchanged compared with GCN. The defense performance is compared with many state-of-the-art defense mechanisms: RobustGCN (Zügner & Günnemann, 2019b), GCN-Jaccard (Wu et al., 2019), GCN-SVD (Entezari et al., 2020), and Pro-GNN series (Jin et al., 2020). Table 3 reports the results of our method and other baselines, and we use the metric reported in (Jin et al., 2020). We observe that our method outperforms other defense mechanisms on Cora and Citeseer in most cases, and achieves relatively close performance with the best defense mechanism on Pubmed.

Table 3: Node classification performance (Accuracy±Std) under non-targeted attack (meta-attack).

Dataset	Ptb Rate (%)	15	20	25
Cora	GCN	65.10	59.56	47.53
	GAT	69.78	59.94	54.78
	RobustGCN	66.82	59.27	50.51
	GCN-Jaccard	71.03	65.71	60.82
	GCN-SVD	66.69	58.94	52.06
	Pro-GNN-fs	76.01	68.78	56.54
	Pro-GNN	76.40	73.32	69.72
	Ours	**76.55**	**75.26**	**72.37**
Citeseer	GCN	64.52	62.03	56.94
	GAT	69.02	61.04	61.85
	RobustGCN	65.69	62.49	55.35
	GCN-Jaccard	65.95	59.30	59.89
	GCN-SVD	63.26	58.55	57.18
	Pro-GNN-fs	70.82	66.19	66.40
	Pro-GNN	72.03	**70.02**	68.95
	Ours	**73.69**	68.22	**69.57**
Pubmed	GCN	78.66	77.35	75.50
	GAT	71.13	68.21	65.41
	RobustGCN	73.91	71.18	67.95
	GCN-Jaccard	84.76	83.88	83.66
	GCN-SVD	83.10	83.01	82.72
	Pro-GNN-fs	**87.20**	87.09	86.71
	Pro-GNN	87.20	**87.15**	**86.76**
	Ours	85.85	85.83	85.57

5. Conclusion

We present a pre-processing approach to select high-order neighbors for GCNs. By assigning weights to high-order neighbors with our proposed loss function, we can control information flow to informative high-order neighbors. Our approach can effectively boost the performance of shallow-depth GCNII, thus reducing the amount of computation in GCNII. Moreover, our selection algorithm can also defend against graph structure attacks.

References

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2019.

Breiman, L. Better subset regression using the nonnegative garrote. *Technometrics*, 1995.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 2020.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

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

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems*, 2016.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 2008.

Shi, C., Xu, M., Guo, H., Zhang, M., and Tang, J. A graph to graphs framework for retrosynthesis prediction. In *International Conference on Machine Learning*, 2020.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1996.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, 2018.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, 2016.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

Zeng, H., Zhang, M., Xia, Y., Srivastava, A., Malevich, A., Kannan, R., Prasanna, V., Jin, L., and Chen, R. Decoupling the depth and scope of graph neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations*, 2019a.

Zügner, D. and Günnemann, S. Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019b.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.