DEFENDING GRAPH CONVOLUTIONAL NETWORKS AGAINST ADVERSARIAL ATTACKS

Vassilis N. Ioannidis *Georgios B. Giannakis*

Digital Technology Center and Dept. of ECE, University of Minnesota, Minneapolis, USA

ABSTRACT

The interconnection of social, email, and media platforms enables adversaries to manipulate networked data and promote their malicious intents. This paper introduces graph neural network architectures that are robust to perturbed networked data. The novel network utilizes a randomization layer that performs link-dithering (LD) by adding or removing links with probabilities selected to boost robustness. The resultant link-dithered auxiliary graphs are leveraged by an adaptive (A)GCN that performs SSL. The proposed robust LD-AGCN achieves performance gains relative to GCNs under perturbed network data.

Index Terms— Deep neural networks, graph convolutional networks, graph signals, dithering, robust learning.

1. INTRODUCTION

Deep learning with graph data achieved remarkable results in a variety of network science tasks. Semi-supervised learning (SSL) aims at predicting nodal attributes across all nodes given the values of those attributes at a subset of nodes and the network connections among nodes. A relevant example is protein-to-protein interaction networks, where the proteins (nodes) are associated with specific biological functions, thereby facilitating the understanding of pathogenic and physiological mechanisms.

Graph-based SSL methods typically utilize a variety of assumptions such as 'smoothness' that implies that connecting nodes have similar labels. Hence, leveraging the topology of the network as a regularizer for learning is highly motivated and typically increases classification performance. Graph-induced smoothness may be captured by kernels on graphs [1, 2]; or via bandlimited models as in graph signal processing, where the signal of interest lies in a low rank space of the Laplacian or adjacancy matrix [3, 4].

Recently, a number of deep learning based approaches incorporate the graph in the neural network architectures and achieved state-of-the-art results [5, 6, 7, 8, 9, 10]. These contemporary approaches learn features in a data-driven fashion and reveal properties of the graph data. Graph-attention mechanism further capitalize on the available data and give attention to 'important' edges [11].

Despite the success of these contemporary graph convolutional networks (GCN)s, recent results indicate that perturbations of the graph topology or nodal features may severely deteriorate their classification performance [12, 13, 14, 15]. The ever-expanding interconnection of social, email, and media service platforms presents an opportunity for adversaries manipulating networked data to launch malicious attacks [16, 17, 12]. Structural attacks target a subset of nodes and modify their links to promote miss-classification of targeted nodes [12].

Contributions. This paper develops a robust deep learning framework for SSL. Given the perturbed unweighted graph, multiple auxiliary graphs are drawn by dithering (adding or removing) links with probabilities selected to boost robustness. The novel link-dithering (LD) approach reconstructs the original neighborhood structure with high probability (w.h.p.) as the number of sampled graphs increases. LD can be applied even in the absence of nodal features. Towards robust SSL, an adaptive GCN architecture is designed to process the multiple graphs and unveil features originating from unperturbed neighborhoods. Experiments with real-data show the gains of the proposed LD-AGCN compared to GCN when the original graph is perturbed.

2. MODELING AND PROBLEM FORMULATION

A graph is denoted as $\mathcal{G} := (\mathcal{V}, \mathbf{S})$ of N nodes, with $\mathcal{V} := \{v_1, \ldots, v_N\}$ the vertex set, and $\mathbf{S}$ the $N \times N$ adjacency matrix capturing edge connectivity. Specifically, if an edge connects v_n and v'_n then $S_{nn'} = 1$, and 0 otherwise. The graphs considered here do not have self-loops, which means that $S_{nn} = 1, \forall n$. The neighborhood of v_n is

$$\mathcal{N}_n := \{n' : S_{nn'} \neq 0, \;\; v'_n \in \mathcal{V}\}. \tag{1}$$

The perturbed graph is $\tilde{\mathcal{G}} := (\mathcal{V}, \tilde{\mathbf{S}})$ with corresponding adjacency $\tilde{\mathbf{S}} := \mathbf{S} + \check{\mathbf{S}}$. The entries of the perturbation matrix

The work in this paper has been supported by the Doctoral Dissertation Fellowship of the Univ. of Minnesota, the USA NSF grants 171141, 1500713, and 1442686.

$\check{\mathbf{S}}$ are defined as follows

$$\check{S}_{nn'} = \begin{cases} 1, & \text{if } S_{nn'} = 0 \text{ and } \tilde{S}_{nn'} = 1 \\ -1, & \text{if } S_{nn'} = 1 \text{ and } \tilde{S}_{nn'} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Evidently, an link insertion corresponds to $\check{S}_{nn'} = 1$, an link deletion to to $\check{S}_{nn'} = -1$, and when no perturbation is observed to $\check{S}_{nn'} = 0$. These perturbed links have a significant impact on the SSL performance, since these modify the neighborhood of the nodes. The number of perturbed edges is usually constrained, since either the adversary aims at *unnoticeable* changes or the noise during graph construction is limited. Hence, the number of perturbed edges (nonzero elements in $\check{\mathbf{S}}$) is small relative to the original number of links (nonzero elements in $\mathbf{S}$).

Associated with the n-th node is an $F \times 1$ vector $\mathbf{x}_n$ holding the features of node n. The matrix $\mathbf{X} := [\mathbf{x}_1^\top, \ldots, \mathbf{x}_N^\top]^\top$ holds these feature vectors. A label $y_n \in \{0, 1, \ldots, K-1\}$ is also considered for node n, which may capture the political beliefs of a person. The "one-hot" representation of the labels is the $N \times K$ matrix $\mathbf{Y}$.

Given the perturbed topology $\tilde{\mathbf{S}}$, $\mathbf{X}$, and $\{y_n\}_{n\in\mathcal{L}}$ with $\mathcal{L} \subset \mathcal{V}$, where $\mathcal{L}$ is a subset of labeled nodes, the goal of this work is to design robust GCN architectures.

3. LINK DITHERING

Perturbed links modify the neighborhood structures of nodes, which leads to significant degradation in the performance of GCNs [12]. This section develops a link-dithering (LD), where auxiliary graphs are created with probabilities designed to enhance robustness. The goal of the LD layer is to restore a node's initial graph neighborhood. We permeate ideas from visual and audio applications, where dithering refers to intentional injection of noise with the goal of converting the quantization error to random noise. Subsequently, techniques as whitening are employed to alleviate the effects of random noise [18].

Permeating the benefits of dithering towards robustifying GCNs, we generate LD graphs $\{\mathcal{G}_i\}_{i=1}^I$. Each auxiliary graph $\mathcal{G}_i := (\mathcal{V}, \mathbf{S}_i)$ is a dithered version of the perturbed graph $\tilde{\mathcal{G}}$, where the links in $\mathbf{S}_i$ are selected in a probabilistic fashion as follows

$$S_{nn'i} = \begin{cases} 1 & \text{wp. } q_1^{\delta(\tilde{S}_{nn'}=1)}(1-q_2)^{\delta(\tilde{S}_{nn'}=0)} \\ 0 & \text{wp. } q_2^{\delta(\tilde{S}_{nn'}=0)}(1-q_1)^{\delta(\tilde{S}_{nn'}=1)} \end{cases} \tag{3}$$

where $\delta(\cdot)$ is the indicator function, $q_1 := \Pr(S_{nn'i} = 1 | \tilde{S}_{nn'} = 1)$ and $q_2 := \Pr(S_{nn'i} = 0 | \tilde{S}_{nn'} = 0)$. If n and n' are connected in the perturbed graph $\tilde{\mathcal{G}}$, the link connecting n with n' is deleted with probability $1 - q_1$. Otherwise, if $(\tilde{S}_{nn'} = 0)$, a link between n and n' is created with probability $1 - q_2$.

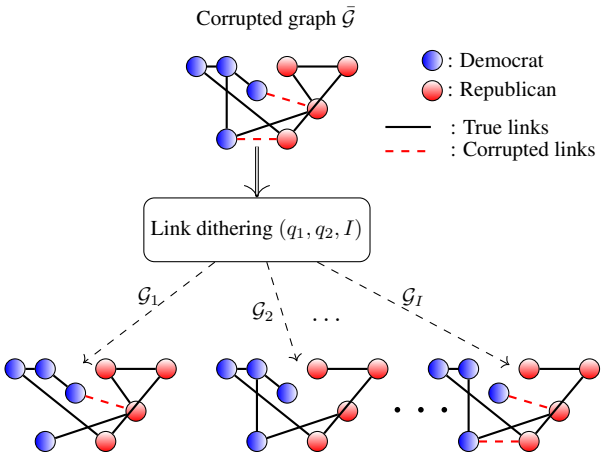

Fig. 1: LD in operation on a perturbed social network among voters. Black solid links are the true links and dashed red links represent perturbed links.

The ith LD graph neighborhood of nodes n is as follows

$$\mathcal{N}_n^{(i)} := \{n' : S_{nn'i} \neq 0, \;\; v_n' \in \mathcal{V}\}. \tag{4}$$

The LD graphs give raise to different neighborhoods $\mathcal{N}_n^{(i)}$, and the role of LD is to ensure that the unperturbed neighborhood of each node will be present with high probability (w.h.p.) in at least one of the I graphs. For the following consider that there is the no-self links are created from the perturbation, i.e. $\tilde{S}_{nn} = 0, \forall n$.

Remark 1 *W.h.p., there exists one LD graph where a perturbed link will be restored to its initial value. Since, each $\mathcal{G}_i$ is independently drawn, it holds that*

$$\Pr\left(\bigcup_{i=1}^{I}(S_{nn'i} = 0)\Big|\tilde{S}_{nn'} = 1, S_{nn'} = 0\right) = 1 - q_1^I$$

$$\Pr\left(\bigcup_{i=1}^{I}(S_{nn'i} = 1)\Big|\tilde{S}_{nn'} = 0, S_{nn'} = 1\right) = 1 - q_2^I$$

Further, it can be shown that local neighborhood for each node will be recovered w.h.p. in at least one LD graph. The high probability claim holds as I increases. Nevertheless, experiments demonstrate that even with a small I the use of LD boosts classification performance. The operation of the LD module is detailed in Fig. 1. Note that the proposed LD does not require availability of nodal feature vectors. The generated graphs have to be processed by a dedicated architecture that promotes the learned features from unperturbed nodal neighborhoods.

4. LD-AGCN ARCHITECTURE

Aiming at robust SSL, the multiple LD graphs are processed by an adaptive GCN architecture. The input information is

processed by a succession of L hidden layers. Each of the layers is composed by a conveniently parametrized linear transformation, a scalar nonlinear transformation, and, oftentimes, a dimensionality reduction (pooling) operator. The intuition is to combine nonlinearly local features to progressively extract useful information [19]. GCNs tailor these operations to the graph that supports the data [5], including the linear [20], nonlinear [20] and pooling [7] operators.

First, the operation of the lth intermediate layer is presented. The $N \times I \times P^{(l-1)}$ tensor $\check{\underline{\mathbf{T}}}^{(l-1)}$ represents the input to the layer, whereas the $N \times I \times P^{(l)}$ tensor is the output of the layer $\check{\underline{\mathbf{T}}}^{(l)}$ that holds the $P^{(l)} \times 1$ feature vectors $\check{\mathbf{t}}_{ni}^{(l)}, \forall n, i$, with $P^{(l)}$ being the number of output features at l. The mapping from $\check{\underline{\mathbf{T}}}^{(l-1)}$ to $\check{\underline{\mathbf{T}}}^{(l)}$ is split into two steps: (i) A linear transformation that maps the $N \times I \times P^{(l)}$ tensor $\check{\underline{\mathbf{T}}}^{(l-1)}$ into the $N \times I \times P^{(l)}$ tensor $\underline{\mathbf{T}}^{(l)}$; (ii) and a scalar nonlinear transformation that is applied to each element of $\underline{\mathbf{T}}^{(l)}$ as $\check{\underline{t}}_{inp}^{(l)} := \sigma(\underline{t}_{inp}^{(l)})$. A common choice for $\sigma(\cdot)$ is the rectified linear unit (ReLU), i.e. $\sigma(c) = \max(0, c)$ [19]. Hence, the main task is to define a linear transformation that maps $\check{\underline{\mathbf{T}}}^{(l-1)}$ to $\underline{\mathbf{T}}^{(l)}$ and is tailored to process the multiple LD graphs.

Convolutional NNs (CNNs) typically employ a small number of weights and then extract the learnable feature as a convolution of the input with these weights [19]. The convolution operation combines values of close-by inputs (consecutive time instants, or neighboring pixels) and thus extracts information of local neighborhoods. GCNs have generalized CNNs to operate on graph data by replacing the convolution with a graph convolution (or graph filter) [5]. The graph filter is usually a polynomial function of the Laplacian or adjacency matrix and captures certain properties of the input features. The graph convolution preserves locality, reduces the degrees of freedom of the transformation, and leverages the structure of the graph. In this work, the graph convolution operation has to be adapted to account for the multiple dithered graphs.

First, a neighborhood aggregation module is considered that combines linearly the information within a LD graph neighborhood. Since the neighborhood depends on the particular LD graph (4), we obtain for the i-th graph

$$\mathbf{h}_{ni}^{(l)} := \sum_{n' \in \mathcal{N}_n^{(i)}} S_{nn'i} \check{\mathbf{t}}_{n'i}^{(l-1)}. \tag{5}$$

While the entries of $\mathbf{h}_{ni}^{(l)}$ depend only on the one-hop neighbors of n (one-hop diffusion), the successive application of this operation across layers will increase the reach of the diffusion, spreading the information across the network. Specifically, consider the r power of the shift matrix $\mathbf{S}^r$. Indeed, the vector $\mathbf{S}^r\mathbf{x}$ holds the linear combination of the values of $\mathbf{x}$ in the r-hop neighborhood [4]. After defining the matrices $\mathbf{S}_i^r := \mathbf{S}_i^{(r)}$ for $r = 1, \ldots, R,\ i = 1, \ldots, I$, consider the following parametrized mapping

$$\mathbf{h}_{ni}^{(l)} := \sum_{r=1}^{R} \sum_{n' \in \mathcal{N}_n^{(i)}} c_i^{(r)} S_{nn'i}^{(r)} \check{\mathbf{t}}_{n'i}^{(l-1)},\ \ \forall n, i. \tag{6}$$

where the learnable coefficients $\{c_i^{(r)}\}_{r=1}^R$ weight the effect of the correspoding r-th hop neighbors for graph i.

The extracted feature $\mathbf{h}_{ni}^{(l)}$ captures the diffused input per edge dithered graph i. The importance of a particular feature will depend on the inference task at hand. The learning algorithm should adapt to and the prevalent features. Towards that end, a graph adaptive module is introduced to adapt the different graphs and combine $\mathbf{h}_{ni}^{(l)}$ across i as follows

$$\mathbf{g}_{ni}^{(l)} := \sum_{i'=1}^{I} R_{ii'n}^{(l)} \mathbf{h}_{ni'}^{(l)} \tag{7}$$

where $R_{ii'n}^{(l)}$ mixes the outputs at different graphs. The combining scalars $R_{ii'n}^{(l)}$ unveil the features originating from LD-graphs that best fit the data. Our LD-AGCN design aspires to give larger weights to features that correspond to minimally perturbed graphs. Clearly, if prior information on the dependence among relations exists, this can be used to constrain the structure $\underline{\mathbf{R}}$ (e.g., by imposing to be diagonal or sparse).

Finally, the LD adaptive features are mixed using a feature aggregation module as follows

$$\underline{t}_{inp}^{(l)} := \mathbf{g}_{ni}^{\top} \mathbf{w}_{ni'p}^{(l)}, \tag{8}$$

where the $P^{(l-1)} \times 1$ learnable vector $\mathbf{w}_{ni'p}^{(l)}$ mixes the features.

Regarding layer $l = 1$, the input $\check{\underline{\mathbf{T}}}^{(0)}$ is defined using $\underline{\mathbf{X}}$ as $\check{\mathbf{t}}_{ni}^{(0)} = \mathbf{x}_n$ for all n, i. On the other hand, the output of our LD-AGCN is obtained as follows

$$\hat{\mathbf{Y}} := g(\check{\underline{\mathbf{T}}}^{(L)}; \boldsymbol{\theta}_g), \tag{9}$$

where $g(\cdot)$ is a nonlinear function, $\hat{\mathbf{Y}}$ is an $N \times K$ matrix, $\hat{Y}_{n,k}$ represents the probability that $y_n = k$, and $\boldsymbol{\theta}_g$ are trainable parameters.

The proposed architecture depends on the aforementioned learnable weights, which are estimated by minimizing the discrepancy between the estimated labels and the given ones. The learning algorithm will assign larger combining weights to the least perturbed topologies. The back-propagation [21] is employed to minimize the classification error. The computational complexity of evaluating each back-propagation update scales linearly with the number of nonzero entries in $\underline{\mathbf{S}}$ (links).

5. NUMERICAL TESTS

We test the proposed AGCN with $L = 3$, $P^{(1)} = 64$, $P^{(2)} = 8$, and $P^{(3)} = K$. The following experiments test the robust-

Dataset	Nodes N	Classes K	$\lvert\mathcal{L}\rvert$
Cora	2,708	7	140
Citeseer	3,327	6	120
Pubmed	19,717	3	30
Polblogs	1,224	2	24

Table 1: Graph datasets

Dataset	Method	Number of attacked nodes $\lvert\mathcal{T}\rvert$				
		20	30	40	50	60
Citeseer	GCN	60.49	56.00	61.49	56.39	**58.99**
	AGCN	**70.99**	**56.00**	**61.49**	**61.20**	58.66
Cora	GCN	76.00	74.66	76.00	62.39	73.66
	AGCN	**78.00**	**82.00**	**84.00**	**73.59**	**74.99**
Pubmed	GCN	**74.00**	71.33	68.99	66.40	69.66
	AGCN	72.00	**75.36**	**71.44**	**68.50**	**74.43**
Polblogs	GCN	**85.03**	86.00	84.99	78.79	86.91
	AGCN	84.00	**88.00**	**91.99**	**78.79**	**92.00**

Table 2: Classification accuracy in percent for nodes in $\mathcal{T}$ for different numbers of attacked nodes.

ness of AGCN to random graph perturbations and adversarial attacks. AGCN utilizes our novel link dithering module to generate multiple LD graphs and alleviate for the perturbations on the graph links. Four network datasets [22] are employed; see also Table 1.

5.1. Robustness of AGCN to random graph perturbations

For this experiment, the same split of the data in train, validation, and test sets as in [8] is employed o facilitate comparison with GCN. The perturbed graph $\tilde{\mathbf{S}}$ is constructed by inserting new links in the original graphs between a random pair of nodes nn' that are not connected in $\mathbf{S}$, i.e. $S_{nn'} = 0$. The added links can be regarded as drawn from Bernoulli distribution. AGCN utilizes the multiple LD graphs generated via the LD module with $I = 10$, $q_1 = 0.9$, and $q_2 = 1$ since no link is deleted in $\tilde{\mathbf{S}}$.

Fig. 2 demonstrates the classification accuracy of the GCN [8] compared to the proposed AGCN as the number of perturbed links is increasing. Evidently, AGCN utilizes the novel LD module, and achieves robust SSL compared to GCN. Surprisingly, even when no links are perturbed, AGCN outperforms GCN. This observation may be attributed to noisy links in the original graphs, which hinder classification perfomance. Furthermore, SSL performance of GCN significantly degrades as the number of perturbed links increases, which suggests that GCN is challenged even by "random attacks".

5.2. Robustness to adversarial attacks on links

The original graphs in Cora, Citeseer, Pubmed, and Polblogs were perturbed using the adversarial setup in [12], where structural attacks are effected on attributed graphs. These attacks perturb connections adjacent to $\mathcal{T}$ a set of targeted

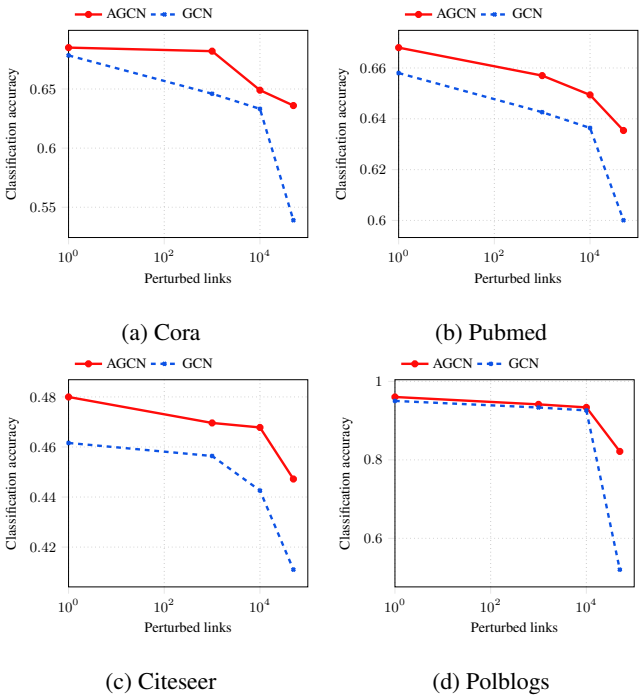

Fig. 2: Classification accuracy for increasing number of perturbed links.

nodes by adding or deleting links [12]. Our LD module uses $I = 10$ sampled graphs with $q_1 = 0.9$, and $q_2 = 0.999$. For this experiment, 30% of the nodes are used for training, 30% for validation and 40% for testing.[1]

Table 2 reports the classification accuracy of GCN and AGCN for different number of attacked nodes ($|\mathcal{T}|$). Different from Fig. 2 where the classification accuracy over the test set is reported, Table 2 reports the classification accuracy over the set of attacked nodes $\mathcal{T}$. It is observed that the proposed AGCN is more robust relative to GCN under adversarial attacks [12]. This finding justifies the use of the novel LD in conjunction with the AGCN that judiciously selects extracted features originating from non-corrupted neighborhoods.

6. CONCLUSIONS

This paper put forth a robust deep learning framework for SSL, which utilized a LD module that performs random dithering to links with probabilities selected to restore a node's original neighborhood with high probability. The auxiliary link-dithered graphs are combined and jointly exploited by the novel AGCN. Experiments demonstrate the performance gains of AGCN in the presence of random as well as adversarial link perturbations.

[1] The nodes in $\mathcal{T}$ are in the testing set.

7. REFERENCES

[1] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, 2006.

[2] V. N. Ioannidis, M. Ma, A. Nikolakopoulos, G. B. Giannakis, and D. Romero, "Kernel-based inference of functions on graphs," in *Adaptive Learning Methods for Nonlinear System Modeling*, D. Comminiello and J. Principe, Eds. Elsevier, 2018.

[3] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Sig. Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[4] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Sig. Process.*, vol. 64, no. 7, pp. 1832–1843, Apr. 2016.

[5] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Sig. Process. Mag.*, vol. 34, no. 4, pp. 18–42, 2017.

[6] V. N. Ioannidis, S. Chen, and G. B. Giannakis, "Pruned graph scattering transforms," in *Proc. Int. Conf. on Learn. Represantions*, 2020.

[7] F. Gama, G. Leus, A. G. Marques, and A. Ribeiro, "Convolutional neural networks via node-varying graph filters," in *IEEE Data Science Workshop*, Lausanne, Switzerland, Jun. 2018, pp. 1–5.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. on Learn. Represantions*, Toulon, France, Apr. 2017.

[9] V. N. Ioannidis, A. G. Marques, and G. B. Giannakis, "A recursive multi-layer graph neural network architecture for processing network data," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Process.*, London, England, May 2019.

[10] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. on Learn. Represantions*, 2019.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. on Learn. Represantions*, 2018.

[12] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. Intl. Conf. on Knowledge Disc. and Data Mining (KDD)*, 2018, pp. 2847–2856.

[13] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," in *IJCAI*, 2019.

[14] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *Proc. Int. Conf. Mach. Learn.*, 2018.

[15] V. N. Ioannidis, D. Berberidis, and G. B. Giannakis, "Graphsac: Detecting anomalies in large-scale graphs," *arXiv preprint arXiv::1910.09589*, 2019.

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. on Learn. Represantions*, 2014.

[17] C. C. Aggarwal, "Outlier analysis," in *Data mining*. Springer, 2015, pp. 237–263.

[18] R. A. Ulichney, "Dithering with blue noise," *Proceedings of the IEEE*, vol. 76, no. 1, pp. 56–79, 1988.

[19] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*. MIT press Cambridge, 2016, vol. 1.

[20] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Advances Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 3844–3852.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[22] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, p. 93, 2008.