Edge Dithering for Robust Adaptive Graph Convolutional Networks

Vassilis N. Ioannidis, and Georgios B. Giannakis
Digital Technology Center and Dept. of ECE, University of Minnesota, Minneapolis, USA*

Abstract

Graph convolutional networks (GCNs) are vulnerable to perturbations of the graph structure that are either random, or, adversarially designed. The perturbed links modify the graph neighborhoods, which critically affects the performance of GCNs in semi-supervised learning (SSL) tasks. Aiming at robustifying GCNs conditioned on the perturbed graph, the present paper generates multiple auxiliary graphs, each having its binary $0-1$ edge weights flip values with probabilities designed to enhance robustness. The resultant edge-dithered auxiliary graphs are leveraged by an adaptive (A)GCN that performs SSL. Robustness is enabled through learnable graph-combining weights along with suitable regularizers. Relative to GCN, the novel AGCN achieves markedly improved performance in tests with noisy inputs, graph perturbations, and state-of-the-art adversarial attacks. Further experiments with protein interaction networks showcase the competitive performance of AGCN for SSL over multiple graphs.

1 Introduction

A task of major importance at the cross-roads of machine learning and network science is semi-supervised learning (SSL) over graphs. SSL aims at predicting nodal labels given: i) the graph connections; ii) feature vectors at all nodes; and iii) labels only at a subset of nodes. This partial label availability may be attributed to privacy concerns (e.g., with medical data); energy considerations (e.g., with wireless sensor networks); or unrated items (e.g., with recommender systems).

Standard SSL schemes typically assume that the available labels and graph connections have certain properties such as smoothness, which asserts that connected nodes have similar attributes (Smola and Kondor 2003). In various scenarios however, robustness issues arise. Powerful adversaries manipulate nodal attributes and connections to bias learning, and promote their malicious goals (Zügner, Akbarnejad, and Günnemann 2018). Further, human annotators or noisy data introduce errors during the graph construction that leads to perturbed edge weights (Akoglu, Tong, and Koutra 2015). Adversarially perturbed or simply anomalous graph data may degrade the performance of SSL algorithms with severe consequences. The recent era of misinformation and "fake" news calls for robust machine learning algorithms for network science (Goodfellow, Shlens, and Szegedy 2014; Aggarwal 2015; Yan et al. 2016). In this context, a novel robust GCN framework is introduced here that utilizes edge-dithered auxiliary graphs, which are combined using learnable weights.

1.1 Related work

Graph-based SSL methods typically assume that the true labels are "smooth" with respect to the underlying network structure, which naturally motivates leveraging the topology of the network to propagate the labels and increase classification performance. Graph-induced smoothness may be captured by kernels on graphs (Belkin, Niyogi, and Sindhwani 2006; Smola and Kondor 2003); or Gaussian random fields (Zhu, Ghahramani, and Lafferty 2003). Graph convolutional networks (GCN)s incorporate the graph structure to achieve state-of-the-art results in SSL tasks (Kipf and Welling 2017; Bronstein et al. 2017; Veličković et al. 2018; Xu et al. 2019b).

With the success of GCNs on graph learning tasks granted, recent results indicate that perturbations of the graph topology or nodal features can severely deteriorate their classification performance (Zügner, Akbarnejad, and Günnemann 2018; Xu et al. 2019a; Dai et al. 2018). Structural attacks target a subset of nodes and modify their links to promote miss-classification of targeted nodes (Wu et al. 2019). The designed graph perturbations are "unnoticeable", which is feasible so long as the degree distribution of the perturbed graphs are similar to the initial distribution (Zügner, Akbarnejad, and Günnemann 2018). GCNs learn nodal representations by extracting information within local neighborhoods. Adversaries poison the learned features by perturbing the node's neighborhood. Hence, the vulnerability of GCNs challenges their deployment to critical applications dealing with security or healthcare, where robust learning is of major importance. Defending against adversaries may

*The work in this paper has been supported by the Doctoral Dissertation Fellowship of the Univ. of Minnesota, the USA NSF grants 1901134, 171141, and 1500713.

unleash the potential of GCNs and broaden the scope of machine learning applications altogether. Recent works robustify GCNs against structural perturbations by utilizing the nodal features (Wu et al. 2019; Zhu et al. 2019). Gaussian regularizers are employed in (Zhu et al. 2019) to protect the network from adversarial attacks. Jaccard similarity among features is utilized in (Wu et al. 2019) to prune perturbed edges. However, these methods are challenged in the absence of nodal feature vectors.

1.2 Contributions

The present paper develops a framework for robust deep learning over perturbed graphs. Specifically, the contribution of this work is twofold.

C1. Given the perturbed unweighted graph and aiming at robust SSL, multiple auxiliary graphs are drawn by dithering (adding or removing) edges with probabilities selected to boost robustness. The novel edge-dithering (ED) approach reconstructs the original neighborhood structure with high probability (whp) as the number of sampled graphs increases. ED can be applied even in the absence of nodal features.

C2. A weighted combination of the auxiliary ED graphs is employed across GCN layers. Per layer weights are adapted to promote those ED graphs that maximally avoid the adversarially perturbed edges. Further, a residual feed of the data is utilized to facilitate diffusion of the features across the graph. Robust graph-based regularizers are also included to prevent overfitting, and further account for the underlying graph topology.

2 Modeling and problem formulation

Consider a graph $\mathcal{G} := (\mathcal{V}, \mathbf{A})$ of N nodes, with $\mathcal{V} := \{v_1, \ldots, v_N\}$ denoting the vertex set, and $\mathbf{A}$ the $N \times N$ adjacency matrix capturing edge connectivity through $A_{n,n'}$ that is 1 if an edge connects v_n and v'_n, and 0 otherwise. The neighborhood of v_n is

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

The perturbed graph is $\bar{\mathcal{G}} := (\mathcal{V}, \bar{\mathbf{A}})$ with corresponding adjacency $\bar{\mathbf{A}} := \mathbf{A} + \check{\mathbf{A}}$ having entries perturbed by

$$\check{A}_{n,n'} = \begin{cases} 1, & \text{if } A_{n,n'} = 0 \text{ and } \bar{A}_{n,n'} = 1 \\ -1, & \text{if } A_{n,n'} = 1 \text{ and } \bar{A}_{n,n'} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $+1$ corresponds to edge insertion, -1 to edge deletion, and 0 to no perturbation. Evidently, these links may drastically degrade the performance of SSL methods since the neighborhood is either adversarially or randomly modified (Zügner, Akbarnejad, and Günnemann 2018). The adversarial attacks aim at *unnoticeable* changes, a constraint that limits the number of perturbations. Hence, the number of perturbed links (nonzero elements in $\check{\mathbf{A}}$) is small relative to the original number of edges in $\mathcal{G}$.

Associated with the n-th node can be an $F \times 1$ feature vector $\mathbf{x}_n$. These vectors are collected in the $N \times F$ feature matrix $\mathbf{X} := [\mathbf{x}_1^\top, \ldots, \mathbf{x}_N^\top]^\top$, where X_{nf} may denote, for example, the salary of the n-th individual in the LinkedIn social network. Let also $y_n \in \{0, 1, \ldots, K-1\}$ denote the label of node n, which may represent, for example, the education level of a person. The $N \times K$ matrix $\mathbf{Y}$ is the "one-hot" representation of the nodal labels belonging to K classes, that is, if $y_n = k$ then $Y_{n,k} = 1$ and $Y_{n,k'} = 0, \forall k' \neq k$.

Goal. Given the perturbed topology $\bar{\mathbf{A}}$, the features in $\mathbf{X}$, and labels only at a subset $\mathcal{L}$ of nodes $\{y_n\}_{n \in \mathcal{L}}$ with $\mathcal{L} \subset \mathcal{V}$, the goal of this paper is to design robust GCN architectures that are minimally affected by the perturbed edges.

3 Edge dithering

The ever-expanding interconnection of social, email, and media service platforms presents an opportunity for adversaries manipulating networked data to launch malicious attacks (Goodfellow, Shlens, and Szegedy 2014; Aggarwal 2015; Zügner, Akbarnejad, and Günnemann 2018). Perturbed edges modify the graph neighborhoods, which leads to significant degradation in the performance of GCNs. Aiming to restore a node's initial graph neighborhood an edge-dithering (ED) module is developed in this section, where auxiliary graphs are created with probabilities designed to enhance robustness. Dithering in visual and audio applications, refers to intentional injection of noise so that the quantization error is converted to random noise, which can be easily handled (Ulichney 1988).

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

$$A_{n,n',i} = \begin{cases} 1 & \text{wp. } q_1^{\delta(\bar{A}_{n,n'}=1)}(1-q_2)^{\delta(\bar{A}_{n,n'}=0)} \\ 0 & \text{wp. } q_2^{\delta(\bar{A}_{n,n'}=0)}(1-q_1)^{\delta(\bar{A}_{n,n'}=1)} \end{cases} \tag{3}$$

where $\delta(\cdot)$ is the indicator function, $q_1 := \Pr(A_{n,n',i} = 1 | \bar{A}_{n,n'} = 1)$ and $q_2 := \Pr(A_{n,n',i} = 0 | \bar{A}_{n,n'} = 0)$. If n and n' are connected in $\bar{\mathcal{G}}$, the edge connecting n with n' is deleted with probability $1 - q_1$. Otherwise, if n and n' are not connected in $\bar{\mathcal{G}}$ i.e. $(\bar{A}_{n,n'} = 0)$, an edge between n and n' is inserted with probability $1 - q_2$.

Hence, the ith ED graph neighborhood of v_n is

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

The ED graphs give raise to different neighborhoods $\mathcal{N}_n^{(i)}$, and the role of ED is to ensure that the unperturbed neighborhood of each node will be present with high probability (whp) in at least one of the I graphs. The remarks asserting formally these claims are in the supplementary material.

4 Adaptive GCN with edge dithering

The ED module generates $\{\mathcal{G}_i\}_{i=1}^I$, which along with the perturbed graph $\bar{\mathcal{G}}$ will be judiciously combined to obtain a robust learning architecture. Typically, deep or shallow learning over graphs considers that the relation among the nodal variables is represented by a single graph. This may

be inadequate in several contemporary applications, where nodes may engage on multiple relations (Kivelä et al. 2014), motivating the generalization of SSL approaches for *single* graphs to *multiple* graphs[1]. In social networks for example, each graph may capture a specific form of social interaction, such as friendship, family bonds, or coworker-ties (Wasserman and Faust 1994). Aiming at a weighted combination of the auxiliary ED graphs, this section develops a novel GCN that adapts to multiple relations and enhances robustness.

Deep learning architectures typically process the input information using L hidden layers. Each layer implements a conveniently parametrized linear transformation, a scalar nonlinear transformation, and oftentimes a dimensionality reduction (pooling) operator. Through nonlinear mappings of linearly combined local features the idea is to progressively extract useful information (Goodfellow et al. 2016). GCNs tailor these operations to the graph that supports the data (Bronstein et al. 2017; Kipf and Welling 2017; Cao, Lu, and Xu 2016; Li et al. 2018). Next, our AGCN architecture and training are presented.

4.1 Per layer operation

Consider a hidden layer (say the lth one), whose output is the $N\times I\times P^{(l)}$ tensor $\check{\underline{\mathbf{Z}}}^{(l)}$ that holds the $P^{(l)}\times 1$ feature vectors $\check{\mathbf{z}}_{n,i}^{(l)}, \forall n,i$, with $P^{(l)}$ being the number of output features at l. Similarly, let $\check{\underline{\mathbf{Z}}}^{(l-1)}$ represent the input to this lth layer.

The mapping from $\check{\underline{\mathbf{Z}}}^{(l-1)}$ to $\check{\underline{\mathbf{Z}}}^{(l)}$ can be split into two steps. A linear one designed to map the tensor $\check{\underline{\mathbf{Z}}}^{(l-1)}$ to the tensor $\underline{\mathbf{Z}}^{(l)}$. The latter is then processed elementwise to obtain $\check{\underline{Z}}_{i,n,p}^{(l)} := \sigma(\underline{Z}_{i,n,p}^{(l)})$. A common choice for $\sigma(\cdot)$ is the rectified linear unit (ReLU), for which $\sigma(c)=\max(0,c)$.

Of critical importance is the design of the linear map from $\check{\underline{\mathbf{Z}}}^{(l-1)}$ to $\underline{\mathbf{Z}}^{(l)}$ that is tailored to our ED-based setup. Convolutional (C)NNs typically consider a small number of trainable weights, and then generate the linear output by convolving the input with these weights (Goodfellow et al. 2016). The convolution combines values of close-by inputs (consecutive time instants, or neighboring pixels), and thus extracts information of local neighborhoods.

GCNs generalize CNNs to operate on graph data by replacing the convolution with a graph filter whose parameters are also learned (Bronstein et al. 2017; Kipf and Welling 2017). This preserves locality, reduces the degrees of freedom of the map, and leverages the graph structure.

Neighborhood aggregation module. First, a neighborhood aggregation module is considered that combines linearly the nodal features within a graph neighborhood. Since the neighborhood depends on the particular ED graph (4), the combined nth feature of the i-th graph is

$$\mathbf{h}_{n,i}^{(l)} := \sum_{n'\in\mathcal{N}_n^{(i)}} A_{n,n',i}\check{\mathbf{z}}_{n',i}^{(l-1)}. \tag{5}$$

While the entries of $\mathbf{h}_{n,i}^{(l)}$ depend only on the one-hop neighbors of n (one-hop diffusion), successive application of this operation will increase the diffusion range, spreading the information across the network. Generalizing to neighborhoods with larger diameter, consider the kth power of the adjacency matrix $\mathbf{A}^k$. Indeed, the vector $\mathbf{A}^k\mathbf{x}$ holds the linear combinations of the values of $\mathbf{x}$ in the k-hop neighborhood (Kipf and Welling 2017). After defining the matrices $\mathbf{A}_i^k := \mathbf{A}_i^{(k)}$ for $k=1,\ldots,K,\ i=1,\ldots,I$, consider the following parametrized mapping

$$\mathbf{h}_{n,i}^{(l)} := \sum_{k=1}^{K}\sum_{n'\in\mathcal{N}_n^{(i)}} c_i^{(k)} A_{n,n',i}^{(k)}\check{\mathbf{z}}_{n',i}^{(l-1)},\ \ \forall n,i \tag{6}$$

where the learnable coefficients $\{c_i^{(k)}\}_{k=1}^K$ weight the effect of the corresponding k-th hop neighbors for relation i. At the l-th layer, the coefficients $\{\{c_i^{(k)}\}_{k=1}^K\}_{i=1}^I$ are collected in the $K\times I$ matrix $\mathbf{C}^{(l)}$. The proposed map in (6) aggregates the diffused features in the K-hop neighborhoods per i.

Graph adaptive module. The extracted feature vector $\mathbf{h}_{n,i}^{(l)}$ captures the diffused features per ED graph i. Aiming at robustness, the learning algorithm should promote features originating from non-perturbed graph neighborhoods. Towards this end, a graph adaptive module is developed that combines $\mathbf{h}_{n,i'}^{(l)}$ across i' as

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

where $R_{i,i',n}^{(l)}$ mixes features across ED graphs. A key contribution of this paper is viewing $\{R_{i,i',n}^{(l)}\}_{i,i',n}$ as training parameters, which allows AGCN to learn how to combine the different relations encoded by the ED graphs. This characteristic enables the novel AGCN to navigate through the ED graphs $\{\mathcal{G}_i\}_{i=1}^I$, and assign larger weights to features originating from non-perturbed neighborhoods.

The graph adaptive module in (7) allows for different $R_{i,i',n}$ per n. Considering the same R for each n, that is $R_{i,i',n}^{(l)} = R_{i,i'}^{(l)}$, results in a design with less parameters at the expense of reduced flexibility. On the other hand, the flexible design in (7) allows large weights $R_{i,i',n}$ for neighborhoods without corrupted edges, even if $\mathcal{G}_i$ is perturbed.

Feature aggregation module. Next, the graph adaptive features $\mathbf{g}_{n,i}^{(l)}$ are mixed using learnable vectors $\mathbf{w}_{n,i,p}^{(l)}$ to obtain

$$\underline{Z}_{i,n,p}^{(l)} := \mathbf{g}_{n,i}^{\top}\mathbf{w}_{n,i,p}^{(l)}, \tag{8}$$
$$i=1,\ldots,I,\ n=1,\ldots,N,\ p=1,\ldots,P^{(l)}\ .$$

The $P^{(l-1)}\times N\times I\times P^{(l)}$ tensor $\underline{\mathbf{W}}^{(l)}$ collects the feature mixing weights $\{\mathbf{w}_{n,i,p}^{(l)}\}$, while the $I\times I\times P^{(l)}$ tensor $\underline{\mathbf{R}}^{(l)}$ collects the graph mixing weights $\{R_{i,i',n}^{(l)}\}$. Upon collecting all the scalars $\{\underline{Z}_{i,n,p}^{(l)}\}$ in $\underline{\mathbf{Z}}^{(l)}$ (6)-(8) reduce to

$$\underline{\mathbf{Z}}^{(l)} := f(\check{\underline{\mathbf{Z}}}^{(l-1)};\boldsymbol{\theta}_z^{(l)}) \tag{9}$$
$$\boldsymbol{\theta}_z^{(l)} := [\mathrm{vec}(\underline{\mathbf{W}}^{(l)});\mathrm{vec}(\underline{\mathbf{R}}^{(l)});\mathrm{vec}(\mathbf{C}^{(l)})]^{\top}. \tag{10}$$

[1]Many works refer to these as multi-layer graphs (Kivelä et al. 2014).

Residual GCN layer. Concatenating L GCN layers diffuses the input $\mathbf{X}$ across the L-hop graph neighborhood, cf. (5). However, the exact size of the relevant neighborhood is not always known a priori. To endow our architecture with increased flexibility, a residual GCN layer is introduced that inputs $\mathbf{X}$ at each l, and thus captures multiple types of diffusion[2]. As a result, the linear operation in (9) is replaced by the residual linear tensor mapping

$$\underline{\mathbf{Z}}^{(l)} := f(\check{\underline{\mathbf{Z}}}^{(l-1)}; \boldsymbol{\theta}_z^{(l)}) + f(\underline{\mathbf{X}}; \boldsymbol{\theta}_x^{(l)}) \tag{11}$$

where $\boldsymbol{\theta}_x^{(l)}$ encodes trainable parameters, cf. (10). When viewed as a map from $\underline{\mathbf{X}}$ to $\underline{\mathbf{Z}}^{(l)}$, the operator in (11) implements a broader class of graph diffusions than the one in (9). If $l = 3$ and $K = 1$ for example, the first summand in (11) is a one-hop diffusion of the input that corresponds to a two-hop (nonlinear) diffused version of $\mathbf{X}$, while the second summand diffuses $\mathbf{X}$ in one-hop. At a more intuitive level, the second summand also guarantees that the impact of $\mathbf{X}$ in the output does not vanish as the number of layers grows.

To recap, aiming at a robust GCN architecture, we introduce a novel edge-dithering module that generates probabilistically auxiliary graphs. These graphs are processed by a robust AGCN architecture that: combines features within neighborhoods originating from the different graphs; adapts to each graph by aggregating the learned features with $\underline{\mathbf{R}}$; uses a simple but versatile residual tensor mapping (11); and employs smoothness and sparsity promoting graph-based regularizers. Additional details on the regularize and the objective are in the supplementary material.

5 Experiments

The AGCN is tested with $L = 3$, $P^{(1)} = 64$, $P^{(2)} = 8$, and $P^{(w)} = K$. The regularization parameters $\{\mu_1, \mu_2, \lambda\}$ are chosen based on the performance of the AGCN in the validation set for each experiment. For the training stage, an ADAM optimizer with learning rate 0.005 was employed (Kingma and Ba 2015), for 300 epochs with early stopping at 60 epochs. The goal here is to provide tangible answers to the following research questions. The supplementary material includes comparison of AGCN to state-of-the-art methods for SSL over multi-relational graphs.

RQ1. How robust is AGCN compared to GCN under noisy features, noisy edge weights, and random as well as adversarial edge perturbations?

RQ2. How sensitive is AGCN to the parameters of the ED module q_1, q_2 and I)?

5.1 Robustness to additive Gaussian input noise

This section reports the performance of the proposed ED-AGCN architecture under noisy graphs or features. For this experiment, the ionosphere dataset is considered, which contains $N = 351$ data points with $F = 34$ features that belong to $K = 2$ classes (Dheeru and Karra Taniskidou 2017). In this case, the graphs $\mathcal{G}_i$ are formed using κ-nearest neighbors graphs for different values of κ (i.e., different number of neighbors). This method computes the link between n and n' based on the Euclidean distance of their features $\|\mathbf{x}_n - \mathbf{x}'_n\|_2^2$.

[2]This is also known as a skip connection (He et al. 2016)

Table 1: Classification accuracy in percent for nodes in $\mathcal{T}$ for different numbers of attacked nodes.

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**

Oftentimes, the available topology and feature vectors might be corrupted with noise. Capturing this noise, $\underline{\mathbf{A}}$ and $\mathbf{X}$ are obtained as $\underline{\mathbf{A}} = \underline{\mathbf{A}}_{tr} + \underline{\mathbf{O}}_{\underline{\mathbf{A}}}, \mathbf{X} = \mathbf{X}_{tr} + \mathbf{O}_{\mathbf{X}}$, where $\underline{\mathbf{A}}_{tr}$ and $\mathbf{X}_{tr}$ represent the *true* multi-relational topology and features and $\underline{\mathbf{O}}_{\underline{\mathbf{A}}}$ and $\mathbf{O}_{\mathbf{X}}$ denote the corresponding additive perturbations. We draw $\underline{\mathbf{O}}_{\underline{\mathbf{A}}}$ and $\mathbf{O}_{\mathbf{X}}$ from a zero mean white Gaussian distribution with specified signal to noise ratio (SNR). Since random additive noise is considered here, the ED module is not employed.

Fig. 3 reports the SSL classification performance for the ionosphere dataset of AGCNs. The AGCN is tested with different values of κ, where $\kappa = 5, 10$ corresponds to a two relational graph, i.e. $I = 2$. We deduce that multiple κ-nearest neighbors graphs lead to learning more robust representations of the data, which testifies to the merits of proposed multi-relational architecture.

5.2 Robustness to Bernoulli noise on edges

This experiment tests our architecture with four network datasets (Sen et al. 2008): "Cora" ($N = 2708, K = 7, |\mathcal{L}| = 140$), "Citeseer" ($N = 3327, K = 6, |\mathcal{L}| = 120$) and "Pubmed" ($N = 19717, K = 3, |\mathcal{L}| = 30$) are citation graphs, while "Polblogs" ($N = 1224, K = 2, |\mathcal{L}| = 24$) is a political blog network. To facilitate comparison, we reproduce the same experimental setup than in (Kipf and Welling 2017), i.e., the same split of the data in train, validation, and test sets. To study the effect of graph perturbations on the neural network architecture, the feature vectors of the citation datasets are not used. Notice that our robust GCN architecture can be applied even in the absence of nodal features, whereas existing approaches are not directly applicable (Wu et al. 2019). For this experiment, the perturbed graph $\bar{\mathbf{A}}$ is generated by inserting new edges in the original graphs between a random pair of nodes n, n' that are not connected in $\mathbf{A}$, i.e. $A_{n,n'} = 0$. The added edges can be regarded as drawn from Bernoulli distribution. AGCN utilizes the multiple graphs generated via the ED module with $I = 10$ sam-

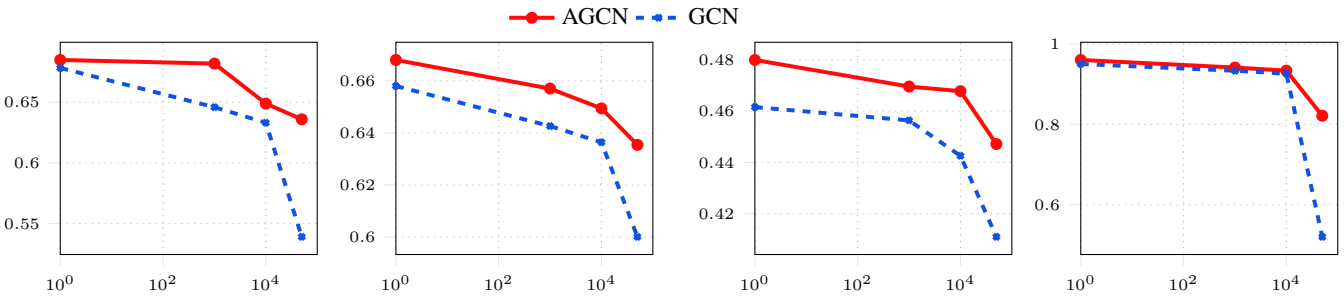

Figure 1: Classification accuracy for increasing number of perturbed edges. (**Left**) Cora, (**Middle left**) Pubmed, (**Middle right**) Citeseer, (**Right** Polblogs).

ples, $q_1 = 0.9$, and $q_2 = 1$ since no edge is deleted in $\bar{\mathbf{A}}$.

Fig. 1 demonstrates the classification accuracy of the GCN (Kipf and Welling 2017) compared to the proposed AGCN as the number of perturbed edges is increasing. Evidently, AGCN utilizes the novel ED module, and achieves robust SSL compared to GCN. Surprisingly, even when no edges 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 edges increases, which suggests that GCN is challenged even by "random attacks".

5.3 Robustness to adversarial attacks on edges

The original graphs in Cora, Citeseer, Pubmed, and Polblogs were perturbed using the adversarial setup in (Zügner, Akbarnejad, and Günnemann 2018), where structural attacks are effected on attributed graphs. These attacks perturb connections adjacent to $\mathcal{T}$ a set of targeted nodes by adding or deleting edges (Zügner, Akbarnejad, and Günnemann 2018). Our ED 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.[3]

Table 1 reports the classification accuracy of GCN and AGCN for different number of attacked nodes ($|\mathcal{T}|$). Different from Fig. 1 where the classification accuracy over the test set is reported, Table 1 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 (Zügner, Akbarnejad, and Günnemann 2018). This finding justifies the use of the novel ED in conjunction with the AGCN that judiciously selects extracted features originating from non-corrupted neighborhoods.

Parameter sensitivity analysis. Fig. 2 includes sensitivity of the AGCN to varying parameters of the ED module for the experiment in Table 1 with the Cora and $|\mathcal{T}| = 30$. It is observed that the AGCN's performance is relative smooth for certain ranges of the parameters. In accordance with Remark 2, notice that even for small I AGCN's performance is increased significantly.

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

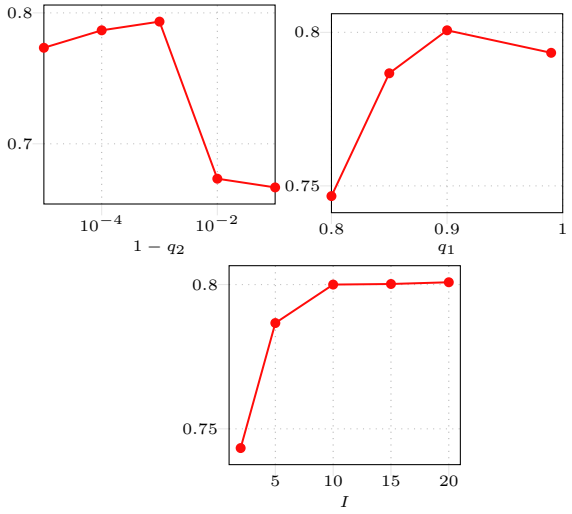

Figure 2: SSL classification accuracy of AGCN under varying edge creation prob. q_1, edge deletion prob. q_2, and number of samples I.

6 Conclusions

This work advocates a novel deep learning approach to robust SSL over perturbed graphs. It relies on random dithering applied to edges with probabilities selected to restore a node's original neighborhood with high probability. The auxiliary edge-dithered graphs are combined and jointly exploited by an adaptive GCN. The latter assigns larger combining weights to learned features extracted from graph neighborhoods without perturbed edges. Experiments demonstrate the performance gains of AGCN in the presence of noisy features, noisy edge weights, and random as well as adversarial edge perturbations.

References

Aggarwal, C. C. 2015. Outlier analysis. In *Data mining*, 237–263. Springer.

Akoglu, L.; Tong, H.; and Koutra, D. 2015. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery* 29(3):626–688.

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regu-

larization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.* 7:2399–2434.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Sig. Process. Mag.* 34(4):18–42.

Cao, S.; Lu, W.; and Xu, Q. 2016. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Dai, H.; Li, H.; Tian, T.; Huang, X.; Wang, L.; Zhu, J.; and Song, L. 2018. Adversarial attack on graph structured data. In *Proc. Intl. Conf. Mach. Learn. (ICML)*.

Dheeru, D., and Karra Taniskidou, E. 2017. UCI machine learning repository.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep Learning*, volume 1. MIT press Cambridge.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. In *Proc. Int. Conf. on Learn. Represantions (ICLR)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE Conf on Comp. Vision and Pat. Rec.*, 770–778.

Kingma, D. P., and Ba, J. L. 2015. Adam: A method for stochastic optimization. In *Proc. Int. Conf. on Learn. Represantions (ICLR)*.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. on Learn. Represantions (ICLR)*.

Kivelä, M.; Arenas, A.; Barthelemy, M.; Gleeson, J. P.; Moreno, Y.; and Porter, M. A. 2014. Multilayer networks. *Journal of Complex Networks* 2(3):203–271.

Li, R.; Wang, S.; Zhu, F.; and Huang, J. 2018. Adaptive graph convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* 29(3):93.

Smola, A. J., and Kondor, R. I. 2003. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines*. Springer. 144–158.

Ulichney, R. A. 1988. Dithering with blue noise. *Proceedings of the IEEE* 76(1):56–79.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *Proc. Int. Conf. on Learn. Represantions (ICLR)*.

Wasserman, S., and Faust, K. 1994. *Social Network Analysis: Methods and Applications*, volume 8. Cambridge university press.

Wu, H.; Wang, C.; Tyshetskiy, Y.; Docherty, A.; Lu, K.; and Zhu, L. 2019. Adversarial examples on graph data: Deep insights into attack and defense. In *IJCAI*.

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

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019b. How powerful are graph neural networks? In *Proc. Int. Conf. on Learn. Represantions (ICLR)*.

Yan, Y.; Xu, Z.; Tsang, I. W.; Long, G.; and Yang, Y. 2016. Robust semi-supervised learning through label aggregation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ye, J., and Akoglu, L. 2018. Robust semi-supervised learning on multiple networks with noise. In *Pacific-Asia Conf. on Knowl. Disc. and Data Mining (PKDD)*, 196–208. Springer.

Zhu, D.; Zhang, Z.; Cui, P.; and Zhu, W. 2019. Robust graph convolutional networks against adversarial attacks. In *Proc. Intl. Conf. on Knowledge Disc. and Data Mining (KDD)*.

Zhu, X.; Ghahramani, Z.; and Lafferty, J. D. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. Intl. Conf. Mach. Learn. (ICML)*, 912–919.

Zitnik, M., and Leskovec, J. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33(14):i190–i198.

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

Table 2: Protein-to-protein interaction datasets

Dataset	Nodes N	Features F	Relations I
Generic cells	4,487	502	144
Brain cells	2,702	81	9
Circulation cells	3,385	62	4

Edge Dithering for Robust Adaptive Graph Convolutional Networks Supplementary material

A Training with graph-smooth regularizers

The output of our graph architecture is

$$\hat{\mathbf{Y}} = g(\check{\underline{\mathbf{Z}}}^{(L)}; \boldsymbol{\theta}_g) \tag{12}$$

where $g(\cdot)$ is the normalized exponential function (softmax), $\hat{\mathbf{Y}}$ 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 AGCN weights are estimated by minimizing the discrepancy between estimated labels and the given ones as

$$\min_{\{\boldsymbol{\theta}_z^{(l)}\},\{\boldsymbol{\theta}_x^{(l)}\},\boldsymbol{\theta}_g} \mathcal{L}_{tr}(\hat{\mathbf{Y}}, \mathbf{Y}) + \mu_1 \sum_{i=1}^{I} \mathrm{Tr}(\hat{\mathbf{Y}}^\top \mathbf{A}_i \hat{\mathbf{Y}}) + \mu_2 \rho(\{\boldsymbol{\theta}_z^{(l)}\}, \{\boldsymbol{\theta}_x^{(l)}\}) + \lambda \sum_{l=1}^{L} \|\underline{\mathbf{R}}^{(l)}\|_1 \tag{13}$$

where $\mathcal{L}_{tr}(\hat{\mathbf{Y}}, \mathbf{Y}) := -\sum_{n\in\mathcal{L}} \sum_{k=1}^{K} Y_{nk} \ln \hat{Y}_{nk}$ is the cross-entropy loss function over the labeled examples.

The first regularizer in (13) promotes smooth label estimates over the graphs (Smola and Kondor 2003), and $\rho(\cdot)$ is an $\mathcal{L}_2$ norm over the AGCN parameters that is used to avoid overfitting (Goodfellow et al. 2016). Finally, the $\mathcal{L}_1$ norm in the third regularizer encourages learning sparse mixing coefficients, and hence it promotes activating only a subset of edge-dithered graphs per l. ED graphs with a large number of perturbed edges will result to a higher cost function in (13). Hence, the learning algorithm will assign larger combining weights to non-perturbed topologies. The backpropagation algorithm is employed to minimize (13).

B Predicting multi-relational protein functions

This section assesses the performance of the proposed AGCN when predicting protein functions over multiple graphs. For this experiment, the given network is multi-relational and no perturbations were considered, hence the ED module is not used. Protein-to-protein interaction networks relate two proteins via multiple cell-dependent relations and protein classification seeks the unknown function of some proteins based on the functionality of a (small) subset of them (Zitnik and Leskovec 2017). Given a target function y_n that is known on a subset of proteins $n \in \mathcal{L}$, known functions on all proteins summarized in $\mathbf{X}$, and the multi-relational protein networks $\underline{\mathbf{A}}$, the goal is to predict whether proteins in the unlabeled set $n \in \{\mathcal{V} \setminus \mathcal{L}\}$ are associated with the target function or not. Hence, the number of target classes in this application is simply $K = 2$. In this setting, $\mathbf{A}_i$ represents the protein connectivity in the i-th cell type. Examples of such cells include cerebellum, midbrain, or frontal lobe. Table 2 summarizes the dimensions of the three datasets used in our experiments.

Next, AGCN is compared with the GCN (Kipf and Welling 2017) that is the single-relational alternative, and the Mune (Ye and Akoglu 2018) that is a state-of-the-art diffusion-based approach for SSL over multi-relational graphs. Since GCN only accounts for a single graph, GCN employs the graph i that achieves the best results in the validation set. Furthermore, Mune does not account for feature vectors in the nodes of the graph. Hence, to facilitate fair comparison, we also employ our AGCN without using the feature vectors, i.e. $\mathbf{X} = \mathbf{I}_N$. Finally, since the classes are heavily unbalanced, we evaluate the performance of the various approaches using the macro F1 score for predicting the protein functions.[4]

Table 3: Macro F1 for the brain cells dataset.

$\lvert\mathcal{L}\rvert$	440	220	110	55
AGCN	**0.86**	**0.79**	**0.71**	**0.69**
GCN	0.49	0.48	0.48	0.47
AGCN (No feat.)	0.41	0.43	0.41	0.35
Mune (No feat.)	0.27	0.27	0.32	0.14

Tables 3-5 report macro F1 values for the aforementioned approaches for varying number of observed (labeled) nodes $|\mathcal{L}|$. The results for all datasets demonstrate that: i) the macro F1 score improves as $|\mathcal{L}|$ increases; ii) AGCN, that judiciously combines the multiple-relations, outperforms the GCN by a large margin; and iii) for the case where nodal features are not used (bottom two rows of each table), AGCN outperforms the state-of-the-art Mune.

[4] Accurate classifiers achieve macro F1 values close to 1.

Table 4: Macro F1 for the circulation cells dataset.

$\lvert\mathcal{L}\rvert$	440	220	110	55
AGCN	**0.77**	**0.76**	**0.70**	**0.69**
GCN	0.48	0.48	0.48	0.47
AGCN (No feat.)	0.41	0.42	0.40	0.35
Mune (No feat.)	0.28	0.27	0.26	0.13

Table 5: Macro F1 for the generic cells dataset.

$\lvert\mathcal{L}\rvert$	440	220	110	55
AGCN	**0.70**	**0.66**	**0.60**	**0.58**
GCN	0.49	0.48	0.48	0.47
AGCN (No feat.)	0.40	0.44	0.41	0.43
Mune (No feat.)	0.28	0.25	0.24	0.13

C High probability remarks

The ensuing remarks assert that this will happen whp as I increases.

Remark 1 *With high probability, there exists $\mathcal{G}_i$ such that a perturbed edge will be restored to its initial value. This means that there exists an ED graph i such that $A_{n,n',i} = A_{n,n'}$. Since, each $\mathcal{G}_i$ is independently drawn, it holds that*

$$\Pr\Big(\bigcup_{i=1}^{I}(A_{n,n',i}=0)\Big|\bar{A}_{n,n'}=1, A_{n,n'}=0\Big) = 1-q_1^I$$
$$\Pr\Big(\bigcup_{i=1}^{I}(A_{n,n',i}=1)\Big|\bar{A}_{n,n'}=0, A_{n,n'}=1\Big) = 1-q_2^I$$

Remark 2 *Whp there exists $\mathcal{G}_i$ which will recover the original neighborhood structure of a node, i.e. $\mathcal{N}_n^{(i)} = \mathcal{N}_n$. The proof of this remark is included in the Appendix.*

The high probability claims asserted in Remarks 1 and 2 hold as I increases. Nevertheless, experiments with adversarial attacks demonstrate that even with a small I the use of ED significantly boosts classification performance. The operation of the ED module is detailed in Fig. 4. Note that the proposed ED 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.

Remark 3 *With high probability there exists $\mathcal{G}_i$ such that a perturbed edge will be restored to its initial value. This means that there exist a graph i such that $A_{n,n',i} = A_{n,n'}$. Since, each $\mathcal{G}_i$ is independently drawn, it holds that* [5]

$$\Pr\Big(\bigcup_{i=1}^{I}(A_{n,n',i}=0)\Big|\bar{A}_{n,n'}=1, A_{n,n'}=0\Big) = 1-q_1^I \tag{14}$$
$$\Pr\Big(\bigcup_{i=1}^{I}(A_{n,n',i}=1)\Big|\bar{A}_{n,n'}=0, A_{n,n'}=1\Big) = 1-q_2^I \tag{15}$$

Remark 4 *With high probability there exists $\mathcal{G}_i$ which will recover the original neighborhood structure of a node, i.e. $\mathcal{N}_n^{(i)} = \mathcal{N}_n$.*

Proof. The two neighborhood structures will be the same $\mathcal{N}_n^{(i)} = \mathcal{N}_n$ if and only if $A_{n,n',i} = A_{n,n'}, \forall n'$. For any edge $A_{n,n',i}$ there are 4 scenarios

1) $\check{A}_{n,n'} = 1$ and $A_{n,n'} = 1$
2) $\check{A}_{n,n'} = 1$ and $A_{n,n'} = 0$
3) $\check{A}_{n,n'} = 0$ and $A_{n,n'} = 1$
4) $\check{A}_{n,n'} = 0$ and $A_{n,n'} = 0$

[5] Equations (k) with $k \leq 13$ correspond to the orginal manuscript while (k) with $k > 13$ correspond to the supplementary material).

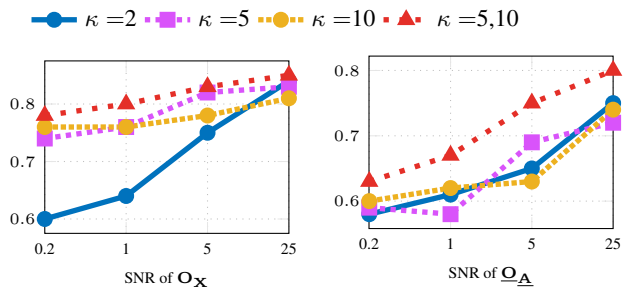

Figure 3: SSL classification accuracy of AGCN with $|\mathcal{L}| = 50$ for noisy features (left) and noisy graphs (right).

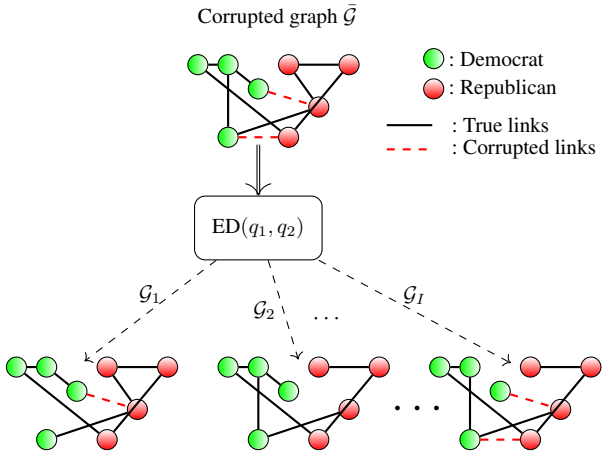

Figure 4: ED in operation on a perturbed social network among voters. Black solid edges are the true links and dashed red edges represent adversarially perturbed links.

It further holds that

$$\Pr\left(\bigcup_{i=1}^{I}(A_{n,n',i}=0)\Big|\bar{A}_{n,n'}=1, A_{n,n'}=0\right)=1-q_1^I \tag{16}$$

$$\Pr\left(\bigcup_{i=1}^{I}(A_{n,n',i}=1)\Big|\bar{A}_{n,n'}=0, A_{n,n'}=1\right)=1-q_2^I \tag{17}$$

$$\Pr\left(\bigcup_{i=1}^{I}(A_{n,n',i}=0)\Big|\bar{A}_{n,n'}=1, A_{n,n'}=1\right)=1-(1-q_1)^I \tag{18}$$

$$\Pr\left(\bigcup_{i=1}^{I}(A_{n,n',i}=1)\Big|\bar{A}_{n,n'}=0, A_{n,n'}=0\right)=1-(1-q_2)^I \tag{19}$$

Without loss of generality assume for the N connections $\{A_{n,n'}\}_{n'=1}^{N}$ the events 1)-4) appear with the following frequency $\kappa, \lambda, \mu, \nu$. Since sampling each edge of the graph is done independently across edges and across draws we arrive to the following

$$\Pr\left(\bigcup_{i=1}^{I}(\mathcal{N}_n^{(i)}=\mathcal{N}_n)\right)=\left(1-(1-q_1)^I\right)^{\kappa}\left(1-q_1^I\right)^{\lambda}\left(1-q_2^I\right)^{\mu}\left(1-(1-q_2)^I\right)^{\nu} \tag{20}$$