Enhancing Robustness of Graph Convolutional Networks via Dropping Graph Connections

Lingwei Chen, Xiaoting Li, and Dinghao Wu(✉)

College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA 16802, USA

Abstract. Graph convolutional networks (GCNs) have emerged as one of the most popular neural networks for a variety of tasks over graphs. Despite their remarkable learning and inference ability, GCNs are still vulnerable to adversarial attacks that imperceptibly perturb graph structures and node features to degrade the performance of GCNs, which poses serious threats to the real-world applications. Inspired by the observations from recent studies suggesting that edge manipulations play a key role in graph adversarial attacks, in this paper, we take those attack behaviors into consideration and design a biased graph-sampling scheme to drop graph connections such that random, sparse and deformed subgraphs are constructed for training and inference. This method yields a significant regularization on graph learning, alleviates the sensitivity to edge manipulations, and thus enhances the robustness of GCNs. We evaluate the performance of our proposed method, while the experimental results validate its effectiveness against adversarial attacks.

Keywords: Graph convolutional networks · Adversarial attacks · Graph sampling · Robustness.

1 Introduction

Graph structured data plays an important role in many real-world applications, which represents natural yet complex relationships between objects in different domains [27, 33], such as social networks, biological networks, and citation networks. Inspired by the great success of applying convolutional neural networks (CNNs) to computer vision tasks [2, 7, 14, 18], many research efforts have been devoted to a paradigm shift in graph learning that generalizes convolutions to the graph domain to address such a challenge [13, 24, 28]. While recent works in this direction are making progress on spatial and spectral approaches respectively, graph convolutional networks (GCNs) have been emerging as one of the most popular and significant graph neural networks [1, 17, 30, 32, 19]. These GCNs take the connectivity structure of the graphs as the filter to perform neighborhood information aggregation so as to extract high-level features from the nodes and their neighborhoods [6], which have thus boosted the state-of-the-arts for a variety of tasks (e.g., node classification, clustering, and matching) over graphs.

Despite their remarkable graph representation learning and inference ability, GCNs are still faced with the same inherent learning-security challenge of

lacking adversarial robustness existing in other deep neural networks (DNNs) [4, 12, 22]. Recent studies [9, 26, 27, 34, 35] have shown that GCNs remain vulnerable to adversarial attacks that carefully design imperceptible perturbations to graph structures and/or node features; these attacks may taint the node neighborhoods, and thus drastically degrade the node representations and the corresponding performance of GCNs. This poses serious threats to real-world applications, especially for those security-critical scenarios such as traffic, financial and healthcare systems [33]. However, compared to the regular DNNs targeting individual objects, GCNs with "message-passing" framework have to cope with correlated objects; therefore, the question of how to improve the robustness of GCNs against adversarial attacks is more complicated and less studied.

Since GCNs deal with graph structured data where node representations are learned by propagating node features through neighborhoods, their vulnerabilities may naturally come from feature perturbations and edge manipulations. However, some recent works [26, 9, 34] suggested that node feature perturbations may not directly impact on the learning performance compromise, while edge manipulations are more effective to formulate successful graph adversarial attacks due to the facts that (1) graph connectivity plays a more crucial role in graph representation learning, and (2) small perturbations over high-dimensional feature space may not induce significant output variations for the aggregation layers. In other words, to enhance the robustness of GCNs against adversarial attacks, a feasible way is to alleviate the impacts of graph connections such that the adversarial edge manipulations will be penalized. When we revisit typical defensive methods over shallow or deep learning models where they are either using weight evenness to restrict weight values to a very narrow range [10], applying ensemble over random feature subspaces [3, 8], or adopting adversarial training to bound the abilities of other adversarial feature perturbations [12], we can observe that these strategies yield the regularization of different degrees to enforce the learning models less sensitive to changes in features and thus improve their robustness. This means that a well-formulated regularization technique over target space can be leveraged to penalize the corresponding space perturbations. Generally, dropout [21] is devised into GCN layers to regularize feature learning, but it gives insufficient penalty to feature perturbations and invalid regularization for edge manipulations. As such, it calls for a more effective regularization method over graph connectivity to confer a significant reduction in GCNs' vulnerability to adversarial attacks.

To address this challenge, in this paper, we propose to exploit graph sampling to drop graph connections. More specifically, instead of constructing GCNs on the complete graph that yields a high variance, we drop out part of the input graph edges to form a set of random and sparse subgraphs and train the GCNs over them. Different from the current graph sampling techniques [6, 20, 31, 11, 5], we take the behaviors of graph adversarial attacks into consideration, and elaborate a biased graph-sampling scheme over well-defined edge sampling probabilities with respect to nodes' degrees and feature similarity. In this regard, the sampled subgraphs may provide more randomness, sparseness and deforma-

tion to alleviate the reliance on the graph connections, and hence penalize the adversarial edge manipulations. Moreover, in order to generate an additional regularization benefit, in the inference stage, we further employ an ensemble approach to aggregate the individual predictions over the subgraphs sampled in the same graph-sampling scheme that are rooted by the test nodes. We summarize our main contributions as follows:

- Graph-sampling mechanisms have been recently proposed to address GCN training over-fitting and deeper-layer computation issues, but never been discussed in the adversarial setting. We leverage this idea and design a biased graph sampling to regularize the graph connections, alleviate the sensitivity to the edge manipulations and hence enhance the robustness of GCNs.
- Instead of using the full graph to infer the test data, we further utilize bagging-like ensemble over sampled subgraphs as is done in parameter learning to approximate the final inference output and provide an additional regularization beyond that provided by model training.
- We conduct comprehensive experimental studies on a number of datasets, which demonstrate that our proposed method can effectively improve the GCN performance against different kinds of adversarial attacks.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries. Section 3 presents our proposed method. Section 4 evaluates the effectiveness of our method. Section 5 discusses related work. Section 6 concludes.

2 Preliminaries

Graph Convolutional Networks. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ be a graph, where $\mathcal{V}$ is the set of nodes $\{v_1, \cdots, v_n\}$ with $|\mathcal{V}| = n$, $\mathcal{E}$ is the set of edges, and $X = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^{n \times c}$ is feature matrix; for each node $v \in \mathcal{V}$, its feature $x_v \in \mathbb{R}^c$ is a c-dimensional row vector. Given $\mathcal{V}$ and $\mathcal{E}$, the adjacency matrix A can be accordingly formulated, where $A \in \mathbb{R}^{n \times n}$ and $A_{ij} = \{0, 1\}$, *i.e.*, if $(v_i, v_j) \in \mathcal{E}$, then $A_{ij} = 1$; otherwise, $A_{ij} = 0$. Based on the adjacency matrix A, the diagonal degree matrix D is defined as $D_{ii} = \sum_{j=1}^{n} A_{ij}$. In this work, the graph convolution network (GCN) proposed by Kipf and Welling [17] is exploited as a base model to facilitate the analysis and understanding of our further proposed approach; therefore, we briefly present its architecture here, where the graph convolutional layer is defined as:

$$H^{(l)} = \sigma\left(\tilde{A} H^{(l-1)} W^{(l)}\right) \tag{1}$$

where $H^{(l-1)}$ and $H^{(l)}$ are the input and output for layer l ($l \geq 1$), $W^{(l)} \in \mathbb{R}^{c_{l-1} \times c_l}$ is the learnable weight matrix for layer l, σ is the non-linear activation function (e.g., ReLU), and $\tilde{A} = \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$. $\tilde{A}$ is a symmetrically normalized adjacency matrix where $\hat{A} = A + I$ is the adjacency matrix of the graph $\mathcal{G}$ with self connections added, and $\hat{D}$ is the diagonal degree matrix of $\hat{A}$. Given

$H^{(0)} = X$, the GCN model with l layers can be thus defined as:

$$Z = f(A, X) = \text{softmax}\left(\tilde{A} \cdots \sigma\left(\tilde{A}XW^{(1)}\right) \cdots W^{(l)}\right). \tag{2}$$

GCNs can be applied under inductive and transductive settings. In this paper, we focus on transductive classification where all node connections and features are accessible during training. To train a GCN model on such a task with labels $\mathcal{C} = \{1, 2, \cdots, c_l\}$, the softmax function normalizes the final output matrix $Z \in \mathbb{R}^{n \times c_l}$, where each row represents the probability of c_l labels for a node. The cross-entropy loss $\mathcal{L} = -\sum_{v \in \mathcal{V}_{tr}} \log Z_{v,c_v}$ can be accordingly evaluated between the output and the corresponding nodes' true labels, while the weights are updated using some gradient descent optimization algorithms (e.g., Adam [16]).

Graph Adversarial Attacks. The adversarial attacks over graph data and GCNs aim to perturb graph structures and/or node features in an imperceptible way to enforce GCNs to incorrectly classify certain nodes. Specifically, given a graph $\mathcal{G} = (A, X)$, it is often possible to find a similar graph $\mathcal{G}' = (A', X')$ such that for a set of nodes $v \in \mathcal{V}' \subset \mathcal{V}$, $y_v = \arg\max_{i \in \mathcal{C}} Z_{v,i}$, $y'_v = \arg\max_{i \in \mathcal{C}} Z'_{v,i}$, and $y_v \neq y'_v$, while $\mathcal{G}$ and $\mathcal{G}'$ are close according to specific measure metric. Such adversarial attacks may be implemented through directly manipulating the edges or features of the target nodes, or indirectly manipulating other nodes to influence the target nodes' classification results [9, 34, 35]. In this work, our goal is to build a robust GCN model that can improve its classification performance on the attacked $\mathcal{G}'$.

3 Proposed Method

In this section, we present the detailed method of how we drop graph connections to formulate subgraphs using biased graph sampling and how we leverage such subgraphs to enhance GCNs' robustness against adversarial attacks.

3.1 Regularization on Graph Connections

As observed in recent works [9, 26, 34], those effective graph adversarial attacks tend to manipulate edges rather than features, where the edge manipulations particularly fall into edge additions (e.g., connecting the target node to the nodes with very different features and labels). The reason behind this could be that GCNs with "message-passing" framework collectively aggregate feature information from the neighborhood of each node at each layer, where adding such an edge may more essentially impact on the target node's full feature dimensions than perturbing individual feature values [26]. As such, a potential defense against adversarial attacks over GCNs should regularize the graph connections to prevent the model learning from over relying on some specific edges and thus cause the adversarial attacks crafted using edge manipulations over the vanilla GCNs less effective.

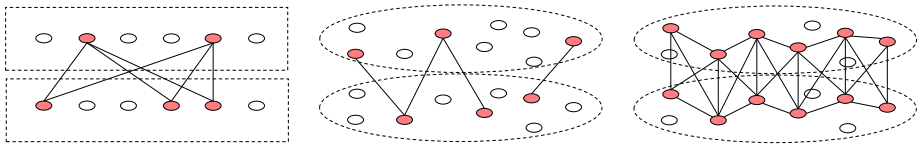

Fig. 1. Dropout examples: (a) a standard dropout on DNNs, (b) a potential way to drop out nodes on GCNs, and (c) another potential way to drop out edges on GCNs.

Dropout is a regularization technique widely used in different neural networks to prevent over-fitting through dropping out units in the hidden and visible layers, which is shown in Figure 1(a). Srivastava et al. [21] explained that dropout may enforce each unit not to rely on a large set of units but to learn to work with a small random set of other units such that the complex co-adaptations would be reduced and each unit is more robust as well. Similarly, if we drop out the graph connections (e.g., dropping out the nodes in Figure 1(b) or dropping out the edges in Figure 1(c)) at each training time, feature propagation may not be able to rely on the full neighborhood while driving each edge to learn to work with a randomly chosen sample of edges and create useful features on its own. In this respect, those manipulated edges would become less effective and the aggregated features could be more robust against adversarial attacks. Although this dropout is shown promising to prevent over-fitting for DNN training, later work pointed out that it doesn't provide sufficient regularization to confer a significant reduction in model's vulnerability to adversarial examples [12]. One possible explanation for this vulnerability could be that each unit is dropped out with the uniform probability; over the long-term training, the hidden units within a layer may still manage to learn the mix-ability of all units to approximate the full feature information including the perturbations. According to the empirical analysis and observations on adversarial attacks presented in [26, 34] that (1) nodes with higher degrees are more difficult to be manipulated than those with lower degrees and (2) adversarial attacks more likely connect the target node to the nodes that are dissimilar in terms of features, this implies that an edge should be dropped out in a different probability with respect to its associated nodes' degrees and feature similarity. That is to say, to better regularize the graph connections, the rationale is to reduce the possibility of those adversarial edges being selected for constructing each training graph batch, while those edges that connect nodes with extremely low similarity may never be selected; in this way, the node neighborhoods will be intuitively smoothed, which makes the aggregated features from nodes more robust against the adversarial attacks.

3.2 Training through Dropping Graph Connections

A severe challenge for GCNs is that recursive feature information aggregation from neighborhoods across layers enforces exponential growth of computations

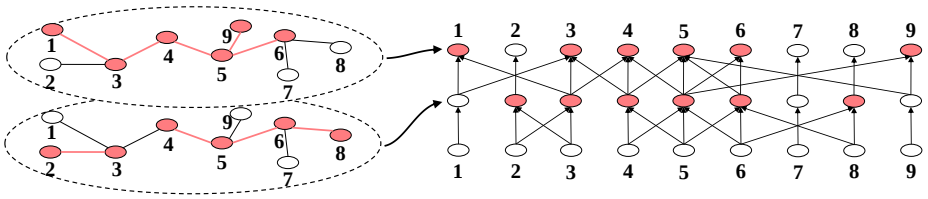

Fig. 2. Layer-wise graph sampling.

in training batches, and thus makes GCNs difficult to be trained efficiently. To address such neighbor explosion, different state-of-the-art graph-sampling methods, either neighbor sampling [13, 5], layer-wise sampling [15, 6], or edge sampling [31, 20], have been proposed to not only speed up the training process, but also reduce the model variance. In this respect, we leverage graph sampling idea to perform dropping graph connections. Clearly, when we sample a certain number of edges from the input graph, those excluded will be naturally dropped out. To make the defense more effective, we take the behaviors of adversarial attacks into consideration and design a biased graph-sampling scheme to construct the subgraphs at each training time with well-defined edge sampling probabilities.

Biased Graph Sampling. Given an input graph $\mathcal{G} = (A, X)$, our goal is to construct a set of subgraphs as training batches, each of which extracts a certain number of edges to formulate a new adjacency matrix. Formally, we introduce a Boolean matrix $Q \in \{0, 1\}^{n \times n}$ to encode whether or not an edge in $\mathcal{G}$ is selected, *i.e.*, if the edge connecting v_i and v_j is selected, then $Q_{ij} = 1$; otherwise, $Q_{ij} = 0$. We also define an edge sampling probability matrix $P \in \mathbb{R}^{n \times n}$ ($P_{ij} \in [0, 1]$) to specify the probability of an edge being selected. If the graph $\mathcal{G}$ is indirect, then both matrices Q and P are symmetric. For each training batch, we utilize a pseudo random function $r(\cdot) \in (0, 1)$ to activate the edge sampling probabilities and decide which edges will be sampled, where

$$Q_{ij} = \begin{cases} 1 & r(\cdot) \leq P_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

If we denote the resulting adjacency matrix for training time t as A_t, then the subgraph can be represented as

$$A_t = Q \circ A \tag{4}$$

where $\circ$ denotes the element-wise product. Following the processing steps in Section 2, we always perform the symmetrical normalization on A_t to obtain $\tilde{A}_t$. Considering that the layer-wise graph sampling provides more randomness and deformation of the input graph [20], which may yield more desirable regularization for the model training, here, we further sample subgraph $A_t^{(l)}$ for each layer l at training time t. Afterwards, training proceeds by iterative weight

Algorithm 1: GCN training through dropping graph connections.

Input: Graph $\mathcal{G} = (A, X)$, epochs T, GCN layers L, training nodes $\mathcal{V}_{tr}$.
Output: GCN model with trained weights W.

Pre-calculate edge sampling probability matrix P;
for *each training epoch* $t \leq T$ **do**
  Pseudorandomly sample L subgraphs $A_t^{(l)}$ ($l \in [1, L]$) using Eq. (3) and (4);
  $Z = \text{softmax}\left(A_t^{\tilde{(L)}} \cdots \sigma\left(A_t^{\tilde{(1)}} X W^{(1)}\right) \cdots W^{(L)}\right)$;
  Back propagation from cross-entropy loss $\mathcal{L} = -\sum_{v \in \mathcal{V}_{tr}} \log Z_{v,c_v}$;
  Update weights W;
end

updates through Eq. (2), where each iteration includes independently sampled subgraphs $A_t^{(l)}$ with respect to the number of GCN layers. Figure 2 and Algorithm 1 illustrate the training algorithm. This graph sampling scheme may inevitably introduce bias into training batches, while we consider such a bias as an advantage to penalize the adversarial edge manipulations.

Edge Sampling Probability. The biased graph sampling method is performed independently on each edge, while the only determinant for an edge being selected for subgraph construction is its sampling probability. Since each edge is associated with two nodes, edge sampling probability should consider the joint information from both nodes in terms of node degrees and features. Based on the variance reduction analysis in [33], the feature aggregation at each training batch can be computed as:

$$\zeta = \sum_{l} \sum_{i,j} \frac{\tilde{A}_{ij} X_j^{(l-1)} W^{(l-1)} + \tilde{A}_{ji} X_i^{(l-1)} W^{(l-1)}}{P_{ij}} \delta_{ij}^{(l)} \tag{5}$$

where $\delta_{ij}^{(l)} = 1$ if $(v_i, v_j) \in \mathcal{E}_t^{(l)}$; otherwise, $\delta_{ij}^{(l)} = 0$. As proved in [33], we can accordingly summarize a theorem that under independent edge sampling with a specified budget, the optimal edge probabilities P_{ij} to minimize the sum of variance of feature aggregation ζ can be approximately given by $P_{ij} \propto \tilde{A}_{ij} + \tilde{A}_{ji}$. It suggests that the edge sampling probability for (v_i, v_j) should be higher if two nodes v_i and v_j have fewer neighbors and are more likely to be influenced by each other. As aforementioned, in the adversarial setting, nodes with more neighbors are more difficult to be attacked than those with fewer neighbors and thus have higher classification accuracy in both the clean and attacked graphs [26]. We can interpret it in the way that (1) if either one of two connected nodes v_i and v_j has few neighbors, this edge is more likely to be manipulated by the attacks, and the edge sampling probability should hence be low, while (2) if both v_i and v_j have many neighbors, the edge tends to already exist in the original input graph. Therefore, in order to gain relatively low variance as well as high

adversarial robustness, we define our edge sampling probability with respect to node degrees as

$$P_{ij} \propto \frac{1}{\tilde{A}_{ij} + \tilde{A}_{ji}} = \frac{\deg(v_i)\deg(v_j)}{\deg(v_i) + \deg(v_j)} = P_{ij}^{\deg} \tag{6}$$

We utilize a probability mass function for all the edges in the given graph:

$$P_{ij}^{\deg} = P_{ij}^{\deg} / \sum_{(v_a, v_b) \in \mathcal{E}} P_{ab}^{\deg}, \ (v_i, v_j) \in \mathcal{E} \tag{7}$$

As for node features, considering that the attackers more likely connect the target node to the nodes with very different features, we further calculate feature similarity score S_{ij} between v_i and v_j to restrict the edge sampling probability. Based on different node feature spaces, the feature similarity measures may vary. For example, Jaccard similarity can be employed to measure binary feature space, while cosine similarity can be used for numeric feature space. Accordingly, the edge sampling probability can be updated as

$$P_{ij} \propto P_{ij}^{\deg} \cdot S_{ij} \tag{8}$$

which implies that the edges that connect nodes with more neighbors and larger feature similarity score have higher sampling probability; if $S_{ij} = 0$, then the corresponding edges would never be selected. Wu et al. [26] indicated that even though there may be a few legitimate edges with very low similarity score in the clean graph, it has little impact on the predictions of the target nodes to remove these edges. We take P as our final edge sampling probability matrix (note that, $P_{ii} = 0$, $\forall v_i \in \mathcal{V}$ and $P_{ij} = 0$, $\forall (v_i, v_j) \notin \mathcal{E}$), which leads to better regularization and robustness since it enables the GCNs to explore the random and deformed graph connections and node features. As stated in Algorithm 1, the probability matrix P can be pre-calculated, which would not significantly increase the computational complexity for the GCN training, except for some extra computations on adjacency matrix formulation and normalization.

3.3 Inference through Ensemble

In the previous section, a biased graph sampling method is presented for constructing subgraphs, which essentially drops the graph connections, and penalizes the adversarial edge manipulations. Based on the trained GCNs with updated weights using such subgraph training batches, we can easily proceed with inference for test data. Generally, a full neighborhood architecture is fed to the model for testing, which is simple and straightforward [6]. However, considering that the adversarial attacks may inevitably manipulate the edges over test data, in this work, instead, we first sample subgraphs from the full graph for test data as is done in GCN training, and then utilize a bagging-like ensemble to average the predictions from all the individual outputs Z over the subgraphs including

Algorithm 2: GCN inference through ensemble.

Input: Graph $\mathcal{G} = (A, X)$, epochs T, trained weights W, testing nodes $\mathcal{V}_{ts}$.
Output: Labels for testing nodes.

Pre-calculate edge sampling probability matrix P;
Initialize the output matrix $O \in \mathbb{R}^{n \times c_l}$ and counter array $N \in \mathbb{R}^n$;
for *each testing epoch* $t \leq T$ **do**
  Pseudorandomly sample a subgraph A_t using Eq. (3) and (4);
  $Z = \text{softmax}\left(\tilde{A}_t \cdots \sigma\left(\tilde{A}_t X W^{(1)}\right) \cdots W^{(L)}\right)$;
  for $v \in \mathcal{V}_{ts}$ *and* v *is sampled* **do**
    $O_v \leftarrow O_v + Z_v$;
    $N_v \leftarrow N_v + 1$;
  end
end
$O_v \leftarrow O_v / N_v, \forall v \in \mathcal{V}_{ts}$;
return *Labels for testing nodes* $\mathcal{V}_{ts}$ *using* O.

the connections to the corresponding test nodes, and hence approximate the final inference results. With the use of ensemble, the inference may provide an additional regularization beyond that provided by model training. The inference algorithm is illustrated in Algorithm 2, which is similar to Algorithm 1, with three differences: (1) the inference only applies a sampled subgraph cross all the layers in a testing epoch for easier test node tracking; (2) the trained weights W will not be updated anymore but directly used for forward aggregation and prediction; and (3) all the outputs Z with sampled target test nodes are further averaged to return the inferred labels for the corresponding test nodes.

4 Experimental Results and Analysis

In this section, we evaluate the performance of our proposed robust GCN model on a number of datasets to defend against adversarial attacks. We perform experiments on the graph-based benchmark node classification tasks with the random splits [29] of each dataset, and compare our model with state-of-the-art baselines.

4.1 Experimental Setup

Datasets. We test our model on two citation network benchmark datasets: *Cora* and *Citeseer* [29, 17] - in both of these datasets, nodes represent documents and edges denote citation links; node features correspond to elements of a bag-of-words representation of a document *i.e.*, 0/1 values indicating the absence/presence of a certain word, while each node has a class label [25]. The dataset statistics are summarized in Table 1. For dataset random split, we adjust Cora and Citeseer respectively to align with our experimental data setting: 20 instances for each class are randomly sampled as training data while another 500 and 1000 instances are selected as validation and test data.

Table 1. Statistics of the datasets used in our experiments.

Dataset	Nodes	Edges	Classes	Features
Cora	2,708	5,429	7	1,433
Citeseer	3,327	4,732	6	3,703

Baselines and Adversarial Attacks. We compare our proposed method (named DropCONN throughout the experiments) with some state-of-the-art baselines, including:

- GCN [17]: this is the original GCN model introduced by Kipf and Welling, which is described in Section 2.
- DropEdge [20]: this is a sampling-based GCN model, which drops out a certain rate of edges of the input graph by random and shares the same perturbed adjacency matrix across all the layers.
- GraphSAINT [31]: this is another sampling-based GCN model, which extracts a connected subgraph for each training batch with the edge sampling rate $P_{ij} \propto \tilde{A}_{ij} + \tilde{A}_{ji}$.
- RGCN [33]: this model defends against graph adversarial attacks by adopting Gaussian distributions as the hidden representations of nodes to absorb the effects of adversarial changes.

To assess the robustness of our proposed method against different attack methods, we choose three representative graph adversarial attacks as follows:

- Nettack [34]: this is a targeted attack method to enforce misclassification on the target nodes using edge and feature perturbations, which can handle both direct and influence attacks. We focus on direct poisoning attack here.
- Meta-gradient attack (Metattack) [35]: this is a non-targeted attack method to reduce the overall classification performance of GCNs using meta-gradients.
- Random attack [33]: it randomly selects some non-adjacent node pairs and add fake edges.

Parameter Setting. Experiments are under the transductive, semi-supervised learning setting. All the GCNs are set as a two-layer structure with 16 hidden units, which are trained using Adam [16] with 0.01 initial learning rate, 5×10^{-4} L2 regularization on the weights, and 0.5 dropout rate for the input and hidden layers. For DropEdge, the edge sampling rate is 0.5. For GraphSAINT, we use edge sampler. The parameter settings of Nettack, Metattack, and Random attack are directly taken from [34], [35] and [33]: (1) Nettack performs the perturbations ranging in $\{1, 2, 3, 4, 5\}$, (2) the edge perturbation rate for Metattack varies in $\{1\%, 5\%, 10\%, 15\%, 20\%\}$, and (3) the perturbation rate for random attack is set as $\{20\%, 40\%, 60\%, 80\%, 100\%\}$. Since the data feature space is binary, we use Jaccard similarity for feature measure in our method. Besides, we report the mean classification accuracy of 10 runs on the test nodes for all the experiments.

Table 2. Classification results (accuracy %) on clean datasets.

Dataset	GCN	DropEdge	GraphSAINT	RGCN	DropCONN
Cora	81.5	82.8	81.1	82.8	80.5
Citeseer	70.3	72.3	71.0	71.2	69.8

4.2 Robustness Evaluation against Adversarial Attacks

Performance without Attack. We first compare our proposed method (i.e., DropCONN) with the original GCN and other variant models on the clean datasets to assess the classification performance when the datasets are not attacked. The experimental results are illustrated in Table 2. From the results, we can see that DropCONN slightly underperforms the baseline models on both datesets. The reason behind this is that the graph sampling using our formulated edge sampling probability may inevitably enforce some very sparse subgraph generations and some edge information loss for those that connect nodes with low feature similarity score as well, while averaging the testing output may further smooth the classification accuracy. Compared to the original GCN, DropCONN degrades the accuracy from 81.5% to 80.5% on Cora and from 70.3% to 69.8% on Citeseer; the performance degradation introduced by our model is not significant. Considering that DropCONN is designed to be resilient against different graph adversarial attacks, which will be thoroughly evaluated later, this comparable performance on clean data shows that our proposed robust solution can be applied in a realistic setting with or without adversarial attacks since we cannot really tell if perturbations happen or not in real-world graphs.

Robustness against Adversarial Attacks. In this section, we conduct experiments to evaluate the effectiveness of our defense strategy against three adversarial attacks (i.e., nettack, metattack, and random attack) to see (1) if the regularization on graph connections through biased graph sampling and ensemble inference contributes to robust graph learning and (2) if DropCONN can outperform other graph-sampling and defensive GCN models. More specifically, for non-targeted attacks (i.e., metattack and random attack), we perturb a certain rate of edges to reduce the overall classification of the model where this perturbation rate varies from 0% to 20% for metattack and from 0% to 100% for random attack. Note that, metattack is performed using meta-gradient approach with self-training [35]. For targeted attack, we set the target nodes as all nodes in the test data and perform nettack to generate different numbers of perturbations (ranging from 1 to 5) for the target nodes. After either non-targeted attacks or targeted attack, we train different GCN models on the modified graph and evaluate the classification accuracy on the test nodes.

We report the comparative evaluation results against three attacks in Figure 3(a)-Figure 3(f) respectively. From Figure 3(a) and Figure 3(b), we can observe that direct nettack attack on the target nodes is a strong attack method,

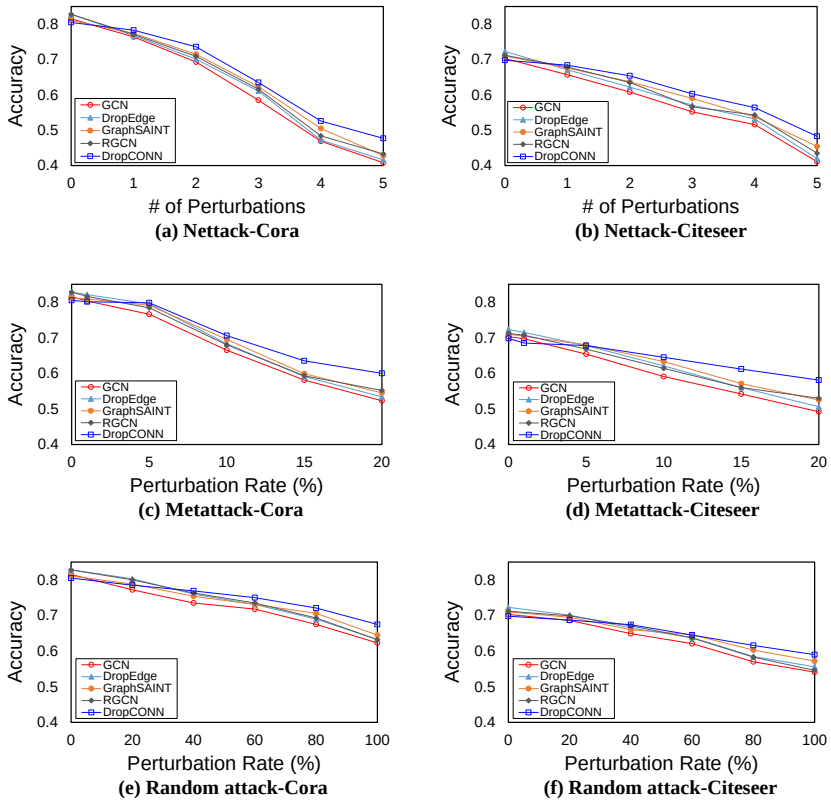

Fig. 3. Classification results (accuracy) under nettack, metattack, and random attack.

where five adversarial perturbations over either graph structure or node features have already drastically degraded the testing accuracy by around 40% on Cora and 30% on Citeseer for most of GCN models; clearly, the original GCN suffers from the biggest drop from 81.5% to 40.8% on Cora and from 70.3% to 41.1% on Citeseer. Despite its apparent attack effect, different GCN variants still show some different defensive potentials against nettack, in which our proposed method DropCONN successfully outperforms other state-of-the-art baselines on both datasets, especially when more perturbations are injected to the graph. Compared to GCN, DropCONN improves the accuracy by a margin of 1.9% to 6.9% and 2.7% to 7.2% respectively, while compared to the best performances from DropEdge, GraphSAINT and RGCN, DropCONN still surpasses them by 0.9% to 4.4% and 0.5% to 2.9% on the corresponding datasets. This is because that DropCONN follows the behaviors and impact of adversarial attacks on node connections and features to design the biased graph sampling and yields a higher regularization on graph connections than other graph sampling methods like DropEdge and GraphSAINT to better penalize the adversarial edge manipulations and thus achieves higher performance against the adversarial attacks.

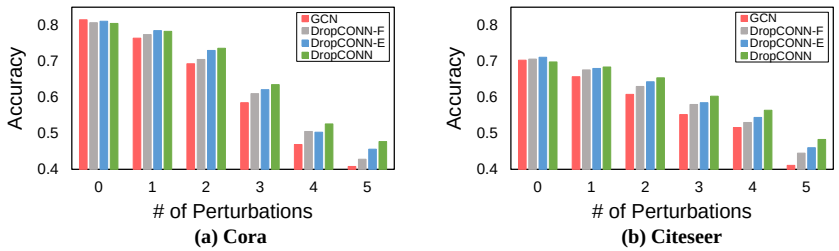

Fig. 4. Evaluations on importance of feature similarity and ensemble.

The same observations can be learned from Figure 3(c)-(d) and Figure 3(e)-(f). Though metattack shows less destructiveness than nettack, it still manages to enforce the GCN models to decay rapidly as the perturbation rate increases, which is stronger than random attack. The experimental results illustrate that DropCONN outperforms all the compared methods under metattack and random attack by a significant margin of performance improvements, which demonstrates its better robustness. Resting on these experimental results and observations, we conclude that DropCONN does not utilize any information of the attack methods and can hence be used in practice to improve the overall robustness of the GCN model against different adversarial attacks.

4.3 Ablation Study

In this section, we conduct ablation studies to further investigate how different components affect the robustness of our proposed method DropCONN. While designing the biased graph sampling scheme, we claim that both node degrees and feature similarity need to be considered for edge sampling probability formulation to adhere to adversarial perturbation behaviors and thus better penalize the edge manipulations. Based on the trained GCN model, we further introduce ensemble to average the outputs over the subgraphs rooted by test nodes to provide an additional regularization. To verify their contributions, we analyze the effect of the proposed method from two aspects: (1) DropCONN-F: since graph sampling methods generally consider the node degrees [6, 31, 13] but ignore the feature similarity, we remove the feature similarity measure from edge sampling probability; and (2) DropCONN-E: we omit ensemble and directly feed the full graph architecture for inference. We validate DropCONN-F and DropCONN-E on both datasets and their performances against nettack are reported in Figure 4. As we can see, feature similarity plays a crucial role in graph sampling to limit negative effects from perturbed edges, while its absence enforces a performance drop by 2.7% of accuracy on Cora and 2.2% on Citeseer on average. DropCONN-E performs closely to DropCONN when the perturbation is small, but the gap between them widens as the perturbation number increases, which implies that ensemble yields an additional advantage for inference. These ob-

servations reaffirm the effectiveness of our design to enhance the robustness of GCNs against adversarial attacks.

5 Related Work

In recent years, graph neural networks (GNNs) on graph structured data have shown outstanding results in various applications [13, 24, 28]. Kipf and Welling [17] further introduced the GCNs via limiting the spectral filters to first-order neighbors for each node. Many GCN variants have been then proposed to leverage the graph structure to capture different properties [1, 30, 32, 19]. However, all of these GCN models lack adversarial robustness. To show the vulnerabilities of GCNs, different graph adversarial attack methods were proposed [9, 26, 27, 34, 35] to perturb graph structure and node features to enforce classification errors. Accordingly, a few attempts have been made to improve the robustness of GCNs against adversarial attacks [26, 27, 33, 23]. For example, Zhu et al. [33] proposed RGCN by adopting Gaussian distributions as the hidden representations of nodes to absorb the effects of adversarial changes; Wu et al. [26] removed the edges from graphs that connects the nodes with low feature similarity score; Tang et al. [23] utilized transfer learning over clean graphs to penalize the perturbations. Differently, in this paper, we leverage graph sampling idea [6, 20, 31, 11, 5] and design a biased graph sampling scheme to drop graph connections, which yields a strong regularization on the model and thus improves the robustness of GCNs.

6 Conclusion

In this paper, we propose to use graph sampling idea to formulate a secure learning solution that enhances the robustness of GCNs against adversarial attacks. By considering the properties of adversarial perturbations, we elaborate a biased graph sampling method over well-defined edge sampling probabilities with respect to nodes' degrees and feature similarity to drop graph connections and construct random, sparse and deformed subgraphs for training, which yields a significant penalty on edge manipulations. Based on the trained GCNs, we further use ensemble to average the outputs over sampled subgraphs rooted by test data to approximate the final inference results, which provides an additional regularization. We evaluate the performance of our proposed method on Cora and Citeseer datasets, while the experimental results validate its effectiveness against adversarial attacks.

Acknowledgments

We thank the reviewers for their valuable feedback. The work was supported in part by the National Science Foundation (NSF) under grant CNS-1652790 and a seed grant from Penn State Center for Security Research and Education (CSRE).

References

1. Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., Galstyan, A.: Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In: International Conference on Machine Learning. pp. 21–29 (2019)
2. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(12), 2481–2495 (2017)
3. Biggio, B., Fumera, G., Roli, F.: Multiple classifier systems for robust classifier design in adversarial environments. International Journal of Machine Learning and Cybernetics **1**(1-4), 27–41 (2010)
4. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. IEEE (2017)
5. Chen, J., Zhu, J., Song, L.: Stochastic training of graph convolutional networks with variance reduction. arXiv preprint arXiv:1710.10568 (2017)
6. Chen, J., Ma, T., Xiao, C.: Fastgcn: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247 (2018)
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(4), 834–848 (2017)
8. Chen, L., Hou, S., Ye, Y.: Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks. In: Proceedings of the 33rd Annual Computer Security Applications Conference. pp. 362–372 (2017)
9. Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L.: Adversarial attack on graph structured data. arXiv preprint arXiv:1806.02371 (2018)
10. Demontis, A., Melis, M., Biggio, B., Maiorca, D., Arp, D., Rieck, K., Corona, I., Giacinto, G., Roli, F.: Yes, machine learning can be more secure! a case study on android malware detection. IEEE Transactions on Dependable and Secure Computing (2017)
11. Gao, H., Wang, Z., Ji, S.: Large-scale learnable graph convolutional networks. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1416–1424 (2018)
12. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
13. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in neural information processing systems. pp. 1024–1034 (2017)
14. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4700–4708 (2017)
15. Huang, W., Zhang, T., Rong, Y., Huang, J.: Adaptive sampling towards fast graph representation learning. In: Advances in neural information processing systems. pp. 4558–4567 (2018)
16. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)

18. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2117–2125 (2017)
19. Liu, S., Chen, L., Dong, H., Wang, Z., Wu, D., Huang, Z.: Higher-order weighted graph convolutional networks. arXiv preprint arXiv:1911.04129 (2019)
20. Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. In: International Conference on Learning Representations (2020)
21. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
22. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
23. Tang, X., Li, Y., Sun, Y., Yao, H., Mitra, P., Wang, S.: Transferring robustness for graph neural network against poisoning attacks. In: Proceedings of the 13th International Conference on Web Search and Data Mining. pp. 600–608 (2020)
24. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
25. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. In: International Conference on Learning Representations (ICLR) (2018)
26. Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., Zhu, L.: Adversarial examples for graph data: Deep insights into attack and defense. In: International Joint Conference on Artificial Intelligence, IJCAI. pp. 4816–4823 (2019)
27. Xu, K., Chen, H., Liu, S., Chen, P.Y., Weng, T.W., Hong, M., Lin, X.: Topology attack and defense for graph neural networks: An optimization perspective. arXiv preprint arXiv:1906.04214 (2019)
28. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? ICLR (2019)
29. Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting semi-supervised learning with graph embeddings. In: International Conference on Machine Learning. pp. 40–48 (2016)
30. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., 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. pp. 974–983 (2018)
31. Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V.: Graphsaint: Graph sampling based inductive learning method. International Conference on Learning Representations (2020)
32. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
33. Zhu, D., Zhang, Z., Cui, P., Zhu, W.: Robust graph convolutional networks against adversarial attacks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1399–1407 (2019)
34. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2847–2856 (2018)
35. Zügner, D., Günnemann, S.: Adversarial attacks on graph neural networks via meta learning. arXiv preprint arXiv:1902.08412 (2019)