Robust Graph Convolutional Networks Against Adversarial Attacks

Dingyuan Zhu*
Tsinghua University
zhudy11@126.com

Ziwei Zhang
Tsinghua University
zw-zhang16@mails.tsinghua.edu.cn

Peng Cui
Tsinghua University
cuip@tsinghua.edu.cn

Wenwu Zhu
Tsinghua University
wwzhu@tsinghua.edu.cn

ABSTRACT

Graph Convolutional Networks (GCNs) are an emerging type of neural network model on graphs which have achieved state-of-the-art performance in the task of node classification. However, recent studies show that GCNs are vulnerable to adversarial attacks, i.e. small deliberate perturbations in graph structures and node attributes, which poses great challenges for applying GCNs to real world applications. How to enhance the robustness of GCNs remains a critical open problem.

To address this problem, we propose Robust GCN (RGCN), a novel model that "fortifies" GCNs against adversarial attacks. Specifically, instead of representing nodes as vectors, our method adopts Gaussian distributions as the hidden representations of nodes in each convolutional layer. In this way, when the graph is attacked, our model can automatically absorb the effects of adversarial changes in the variances of the Gaussian distributions. Moreover, to remedy the propagation of adversarial attacks in GCNs, we propose a variance-based attention mechanism, i.e. assigning different weights to node neighborhoods according to their variances when performing convolutions. Extensive experimental results demonstrate that our proposed method can effectively improve the robustness of GCNs. On three benchmark graphs, our RGCN consistently shows a substantial gain in node classification accuracy compared with state-of-the-art GCNs against various adversarial attack strategies.

KEYWORDS

Graph Convolutional Networks, Robustness, Adversarial Attacks, Deep Learning

ACM Reference Format:
Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330851

*Beijing National Research Center for Information Science and Technology(BNRist)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '19, August 4–8, 2019, Anchorage, AK, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330851

1 INTRODUCTION

Graphs are ubiquitous in the real world, representing complex relationships between objects such as social networks, e-commerce networks, biological networks and traffic networks. How to mine the rich value underlying graph data has long been an important research direction. Graph Convolutional Networks (GCNs) are a type of neural network model for graphs that recently attracts considerable research attention [3, 8, 18]. State-of-the-art GCNs usually follow a "message-passing" framework [10, 35] where each node aggregates information from its immediate neighbors in each convolutional layer. GCNs have been shown to achieve promising results in many graph analytic tasks such as node classification.

Despite their remarkable performance, recent studies show that GCNs are vulnerable to adversarial attacks [7, 37], i.e. carefully designed small perturbations in graph structures and node attributes. By changing only a few links or node attributes which are unnoticeable to the users, the performance of GCNs can be drastically degraded, which poses great challenges for applying GCNs to real world applications, especially those risk-sensitive scenarios such as finance networks or medical networks. Therefore, how to design a robust GCN model in the presence of adversarial attacks is a crucial open question.

The challenges of this problem are two-fold. Firstly, we need to enhance the robustness of GCNs while maintaining its effectiveness, which in turn requires that we need to empower GCNs to tolerate adversarial information while not changing its backbone architecture. Secondly, since nodes in the graphs are entangled, the adversarial attacks can propagate to affect many other nodes besides the directly attacked nodes. How to prevent such propagation in the "message-passing" framework is another key issue.

In this paper, we propose Robust GCN (RGCN), a novel model to improve the robustness of GCNs in the presence of adversarial attacks. Specifically, rather than representing nodes by plain vectors as in all existing GCNs, we propose adopting Gaussian distributions as the hidden representations of nodes in all graph convolutional layers. The motivation is that when the graph data is attacked, the perturbations in graph structures and node attributes are usually abnormal compared to the existing information. For example, the attacker tends to connect nodes from different communities to confuse the classifier [7]. While plain vectors cannot adapt to such changes, Gaussian distributions can automatically absorb the effects of such unexpected adversarial changes in the variances [1, 36]. As a result, using Gaussian distributions can enhance the

robustness of GCNs. Furthermore, in order to remedy the propagation of adversarial attacks in GCNs, we propose a variance-based attention mechanism, i.e. assigning different weights to node neighborhoods according to their variances in the convolution operation. More concretely, when one node is attacked in previous GCNs, the adversarial effects will propagate to its neighbors through the convolutions, affecting a large proportion of the graph. In our model, the effect of such propagation will be greatly reduced since the attacked nodes tend to have large variances and small weights in affecting other nodes.

To verify the efficacy of our proposed method, we conduct extensive experiments on several benchmark datasets for the task of node classification. Experimental results show that our RGCN consistently outperforms state-of-the-art GCNs under various adversarial attack strategies, demonstrating the robustness of our proposed method. We also empirically analyze the reasons behind the effectiveness of our proposed method.

The contributions of this paper are summarized as follows:

- We propose RGCN, a novel model that explicitly enhances the robustness of GCNs against adversarial attacks. To the best of our knowledge, we are the first to investigate this critical and challenging problem.
- We propose using Gaussian distributions in graph convolutional layers to absorb the effects of adversarial attacks and introduce a variance-based attention mechanism to prevent the propagation of adversarial attacks in GCNs.
- Extensive experimental results demonstrate the effectiveness of our proposed method under various adversarial attack strategies.

The rest of the paper is organized as follows. In Section 2, we review related works. In Section 3, we summarize the notations and give some preliminaries. We introduce our proposed method in Section 4 and report experimental results in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

Our work builds upon two categories of recent research: graph convolutional networks and graph adversarial attacks.

2.1 Graph Convolutional Networks

Graph convolutional networks (GCNs), aiming to generalize convolutional neural networks to graph data, have drawn increasing research interests in the past few years. Next, we briefly review some representative GCNs, and readers are referred to [2, 35] for some comprehensive surveys.

Bruna et al. [3] first introduce convolutions for graph data using graph signal processing [25]. By using the eigen-decomposition of the graph Laplacian matrix, the convolution is defined in the graph spectral domain. However, since the full eigenvectors of the Laplacian matrix are needed, the time complexity of such graph convolution is very high. To solve the efficiency problem, ChebNet [8] proposes to use a K-order Chebyshev polynomial [13] to approximate the convolutional filters in the spectral domain. By using the recurrence relation of Chebyshev polynomials, the exact eigen-decomposition of the Laplacian matrix is avoided, thus reducing the overall time complexity. Kipf and Welling [18] further propose to simplify the graph convolution using only the 1^{st} order polynomial, i.e. the immediate neighborhoods. By stacking multiple convolutional layers, this GCN variant achieves state-of-the-art performance in the semi-supervised node classification task. They also design a variational graph autoencoder using GCN as the encoder [17]. MPNNs [10] and GraphSAGE [12] unify these approaches using the "message-passing" framework, i.e. defining the graph convolution as nodes aggregating information from immediate neighborhoods. Further improvements include adding an attention mechanism to assign different weights in aggregating node neighborhoods [28, 30, 31, 34], adding residual and jumping connections [32], sampling to improve efficiency [4, 5, 14, 33] considering edge attributes [15, 23, 26], disentangling node representations [20] and automatically selecting hyper-parameters [29]. However, all of these works do not consider the robustness of the GCN models.

2.2 Graph Adversarial Attacks

Recently, to show the vulnerability of GCNs, some adversarial attack methods have been proposed. The basic idea is to perturb graph structures and nodes attributes so that GCNs cannot correctly classify certain nodes. Meanwhile, the perturbations (i.e. attacks) should be made unnoticeable to users. Based on different settings, graph adversarial attacks can be divided into following categories [27]:

- **Poisoning** or **Evasion**. Poisoning attacks [37, 38] try to attack the model by changing training data and evasion attacks [7, 37] try to generate fake samples for a trained model, i.e. the attacks are categorized by whether they are before (poisoning attacks) or after (evasion attacks) the training phase of GCNs.
- **Targeted** or **Non-targeted**. In targeted attacks [7, 37], the attacker focus on misclassifying some target nodes while in non-targeted attacks [38], the attacker aims to reduce the overall model performance.
- **Direct** or **Influence**. Targeted attacks can be further divided into two categories based on attack settings. In direct attacks [7, 37], the attacker can directly manipulate the edges or features of the target nodes. In influence attacks [37], the attacker can only manipulate other nodes except the targets.

Graph adversarial attacks have posed great challenges to the robustness of GCNs, which severely limits the applicability of GCNs in real world applications. To the best of our knowledge, there is no study on improving the robustness of GCNs or preventing them from adversarial attacks.

3 NOTATIONS AND PRELIMINARIES

In this section, we summarize the notations used in this paper and introduce some preliminaries of GCNs.

3.1 Notations

In this paper, a graph is defined as $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{v_1, ..., v_N\}$ denotes the set of nodes, $N = |\mathbf{V}|$ is the number of nodes, and $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ is the set of edges between nodes. Let $\mathbf{A}$ denote the adjacency matrix and $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{i,j}$ denote the diagonal degree

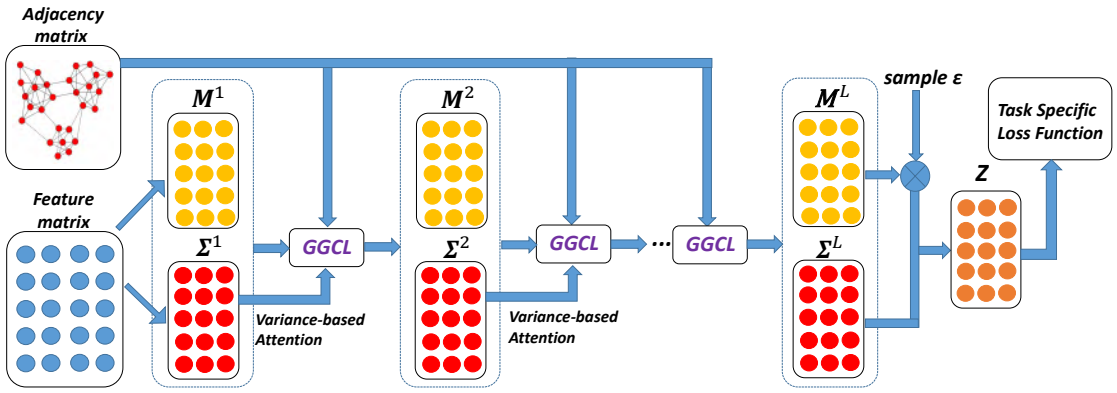

Figure 1: The framework of our proposed RGCN. GGCL represents Gaussian-based Graph Convolution Layer introduced in Section 4.2.

matrix. Let $\mathbf{X}$ be a matrix of node feature vectors. We define $\mathbf{H}^{(l)} = \left[\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, ..., \mathbf{h}_N^{(l)}\right]$ as the hidden representations of nodes in the l^{th} layer of a deep model where $\mathbf{h}_i^{(l)}$ is the representation of node v_i and $[\cdot,\cdot]$ is concatenation. We further define f_l as the dimensionality of $\mathbf{h}_i^{(l)}$ and L as the number of layers. For convenience, we also denote $\mathbf{X}$ as $\mathbf{H}^{(0)}$. The immediate neighborhoods of node v_i, including v_i itself, are denoted as $ne(i)$.

3.2 Graph Convolutional Networks

Though a number of different GCN methods have been proposed, here we focus on a representative one proposed by Kipf and Welling [18]. Here, the $(l+1)^{th}$ convolutional layer is defined as:

$$\mathbf{h}_i^{(l+1)} = \rho\left(\sum_{j \in ne(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}} \mathbf{h}_j^{(l)} \mathbf{W}^{(l)}\right), \tag{1}$$

or in the equivalent matrix form:

$$\mathbf{H}^{(l+1)} = \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{2}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$, $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}_N$, $\mathbf{I}_N$ is the identity matrix, $\rho(\cdot)$ is a non-linear activation function such as ReLU and $\mathbf{W}^{(l)}$ are trainable parameters. The general philosophy is that nodes should exchange information with their immediate neighbors in each convolutional layer, followed by applying learnable filters and some non-linear transformation. This architecture can be trained end-to-end using task-specific loss function, for example, the cross entropy loss in semi-supervised node classification as follows:

$$\mathcal{L}_{cls} = -\sum_{v \in \mathbf{V}^L} \sum_{c=1}^{C} \mathbf{Y}_{vc} \log \hat{\mathbf{Y}}_{vc}, \tag{3}$$

where $\mathbf{V}^L$ is the set of labeled nodes, C is the number of classes, $\mathbf{Y}$ is the label matrix and $\hat{\mathbf{Y}} = softmax(\mathbf{H}^{(L)})$ are predictions of GCN by passing the hidden representation in the final layer $\mathbf{H}^{(L)}$ to a softmax function.

Although this GCN variant is shown extremely effective in the node classification task, later works show that it can be easily fooled by adversarial attacks, as discussed in Section 2.2.

4 ROBUST GRAPH CONVOLUTIONAL NETWORK

In this section, we introduce our proposed method. We first show the overall framework, and then elaborate on the technical details.

4.1 Framework

To enhance the robustness of GCNs against adversarial attacks, we propose Robust Graph Convolutional Network (RGCN) and the framework is shown in Figure 1. Compared with existing methods, we introduce two novel modifications: we use Gaussian distributions as the hidden representations of nodes in graph convolutional layers and assign attention weights to node neighbors according to their variances. Meanwhile, our model explicitly considers the mathematical relevance between means and variances by the sampling process and regularizations. In the following subsections, we will introduce how to realize our model in detail.

4.2 Gaussian-based Graph Convolution Layer

Since we use Gaussian distributions in the hidden layers, the existing graph convolutions are no longer applicable. Next, we formally define a Gaussian-based graph convolution layer to perform convolution operations between Gaussian distributions. Denote $\mathbf{h}_i^{(l)} = \mathcal{N}(\boldsymbol{\mu}_i^{(l)}, diag(\boldsymbol{\sigma}_i^{(l)}))$ as the latent representation of node v_i in layer l, where $\boldsymbol{\mu}_i^{(l)} \in \mathbb{R}^{f_l}$ is the mean vector and $diag(\boldsymbol{\sigma}_i^{(l)}) \in \mathbb{R}^{f_l \times f_l}$ is the diagonal variance matrix[1]. We use $\mathbf{M}^{(l)} = \left[\boldsymbol{\mu}_1^{(l)}, ..., \boldsymbol{\mu}_N^{(l)}\right] \in \mathbb{R}^{N \times f_l}$ and $\Sigma^{(l)} = \left[\boldsymbol{\sigma}_1^{(l)}, ..., \boldsymbol{\sigma}_N^{(l)}\right] \in \mathbb{R}^{N \times f_l}$ to denote the matrix of means and variances for all nodes, respectively.

[1] In this paper, we focus on diagonal variance matrices, but it can be extended to more general cases [22]. We also use $\boldsymbol{\sigma}$ rather than $\boldsymbol{\sigma}^2$ to represent variances for the ease of presentation.

Consider n independent random variables following Gaussian distributions. Their weighted sum also follows a Gaussian distribution, as we show in the following theorem.

THEOREM 4.1. *If* $\mathbf{x}_i \sim \mathcal{N}\left(\boldsymbol{\mu}_i, diag(\boldsymbol{\sigma}_i)\right)\ i = 1, ..., n$, *and they are independent, then for any fixed weights* w_i, *we have:*

$$\sum_{i=1}^{n} w_i \mathbf{x}_i \sim \mathcal{N}\left(\sum_{i=1}^{n} w_i \boldsymbol{\mu}_i, diag\left(\sum_{i=1}^{n} w_i^2 \boldsymbol{\sigma}_i\right)\right). \tag{4}$$

PROOF. The proof can be found in probability textbooks such as [22] and is omitted for brevity. □

Using this theorem and assuming all hidden representations of nodes are independent[2], we can aggregate node neighbors as follows:

$$\begin{aligned}\mathbf{h}_{ne(i)}^{(l)} &= \sum_{j \in ne(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}} \mathbf{h}_j^{(l)} \\ &\sim \mathcal{N}\left(\sum_{j \in ne(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}} \boldsymbol{\mu}_j^{(l)}, diag\left(\sum_{j \in ne(i)} \frac{1}{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}} \boldsymbol{\sigma}_j^{(l)}\right)\right).\end{aligned} \tag{5}$$

In other words, the aggregation of node neighbors also follows a Gaussian distribution.

In addition, to prevent the propagation of adversarial attacks in GCNs, we propose an attention mechanism to assign different weights to neighbors based on their variances since larger variances indicate more uncertainties in the latent representations and larger probability of having been attacked. Specifically, we use a smooth exponential function to control the effect of variances on weights

$$\boldsymbol{\alpha}_j^{(l)} = exp(-\gamma \boldsymbol{\sigma}_j^{(l)}), \tag{6}$$

where $\boldsymbol{\alpha}_j^{(l)}$ are the attention weights of node v_j in the layer l and γ is a hyper-parameter. Then, we modify Eq. (5) as follows:

$$\begin{aligned}\mathbf{h}_{ne(i)}^{(l)} &= \sum_{j \in ne(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}} (\mathbf{h}_j^{(l)} \odot \boldsymbol{\alpha}_j^{(l)}) \\ &\sim \mathcal{N}\left(\sum_{j \in ne(i)} \frac{\boldsymbol{\mu}_j^{(l)} \odot \boldsymbol{\alpha}_j^{(l)}}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}}, diag\left(\sum_{j \in ne(i)} \frac{\boldsymbol{\sigma}_j^{(l)} \odot \boldsymbol{\alpha}_j^{(l)} \odot \boldsymbol{\alpha}_j^{(l)}}{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}\right)\right),\end{aligned} \tag{7}$$

where $\odot$ is the element-wise product. Note that the attention weights are exerted for different dimensions separately.

Following the "message-passing" framework, next we need to apply learnable filters and non-linear activation functions to $\mathbf{h}_{ne(i)}^{(l)}$ to get $\mathbf{h}_i^{(l+1)}$. However, this is mathematically intractable since $\mathbf{h}_{ne(i)}^{(l)}$ is a Gaussian distribution. Alternatively, we directly impose layer-specific parameters and non-linear activation functions to the means and variances respectively. Formally, we define Gaussian-based graph convolution as follows:

$$\begin{aligned}\boldsymbol{\mu}_i^{(l+1)} &= \rho\left(\sum_{j \in ne(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}}} \left(\boldsymbol{\mu}_j^{(l)} \odot \boldsymbol{\alpha}_j^{(l)}\right) \mathbf{W}_\mu^{(l)}\right) \\ \boldsymbol{\sigma}_i^{(l+1)} &= \rho\left(\sum_{j \in ne(i)} \frac{1}{\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j}} \left(\boldsymbol{\sigma}_j^{(l)} \odot \boldsymbol{\alpha}_j^{(l)} \odot \boldsymbol{\alpha}_j^{(l)}\right) \mathbf{W}_\sigma^{(l)}\right),\end{aligned} \tag{8}$$

or equivalently in the matrix form:

$$\begin{aligned}\mathbf{M}^{(l+1)} &= \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\left(\mathbf{M}^{(l)} \odot \mathcal{A}^{(l)}\right)\mathbf{W}_\mu^{(l)}\right) \\ \Sigma^{(l+1)} &= \rho\left(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1}\left(\Sigma^{(l)} \odot \mathcal{A}^{(l)} \odot \mathcal{A}^{(l)}\right)\mathbf{W}_\sigma^{(l)}\right),\end{aligned} \tag{9}$$

where $\mathbf{W}_\mu, \mathbf{W}_\sigma$ are parameters for the means and the variances respectively and $\mathcal{A}^{(l)} = exp(-\gamma\Sigma^{(l)})$.

For the first layer, since input features are vectors rather than Gaussian distributions, we adopt a fully connected layer as follows:

$$\mathbf{M}^{(1)} = \rho\left(\mathbf{H}^{(0)}\mathbf{W}_\mu^{(0)}\right), \Sigma^{(1)} = \rho\left(\mathbf{H}^{(0)}\mathbf{W}_\sigma^{(0)}\right). \tag{10}$$

4.3 Loss Functions

Next, we introduce the loss functions of our proposed method. In this paper, we focus on the task of node classification. Considering that the hidden representations of our method are Gaussian distributions, we first adopt a sampling process in the last hidden layer

$$\mathbf{z}_i \sim \mathcal{N}\left(\boldsymbol{\mu}_i^{(L)}, diag\left(\boldsymbol{\sigma}_i^{(L)}\right)\right), \tag{11}$$

i.e. $\mathbf{z}_i$ is sampled from $\mathbf{h}_i^{(L)}$. Next, $\mathbf{z}_i$ is passed to a softmax function to get the predicted labels:

$$\hat{\mathbf{Y}} = softmax(\mathbf{Z}), \mathbf{Z} = [\mathbf{z}_1, .., \mathbf{z}_N]. \tag{12}$$

Then, we can use the same cross entropy loss $\mathcal{L}_{cls}$ defined in Eq. (3) as the objective function for the task of node classification. Moreover, we can use the "reparameterization trick" [9] to optimize this loss function. Mathematically, we first sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then compute $\mathbf{z}_i = \boldsymbol{\mu}_i^{(L)} + \boldsymbol{\epsilon} \odot \sqrt{\boldsymbol{\sigma}_i^{(L)}}$. Given a fixed $\boldsymbol{\mu}_i^{(L)}, \boldsymbol{\sigma}_i^{(L)}$ and $\boldsymbol{\epsilon}$, the loss function is deterministic and continuous with respect to model parameters.

In addition, to ensure that the learned representations are indeed Gaussian distributions, we use an explicit regularization to constrain the latent representations in the first layer as follows:

$$\mathcal{L}_{reg1} = \sum_{i=1}^{N} KL\left(\mathcal{N}(\boldsymbol{\mu}_i^{(1)}, diag(\boldsymbol{\sigma}_i^{(1)}))||\mathcal{N}(\mathbf{0}, \mathbf{I})\right), \tag{13}$$

where $KL(\cdot||\cdot)$ is the KL-divergence between two distributions [19]. Note that we only need to regularize $\mathbf{H}^{(1)}$ as deeper layers are naturally Gaussian distributions by using our Gaussian-based Graph Convolutions.

Following the original GCN model [18], we also impose L_2 regularization on parameters of the first layer as follows:

$$\mathcal{L}_{reg2} = \left\|\mathbf{W}_\mu^{(0)}\right\|_2^2 + \left\|\mathbf{W}_\sigma^{(0)}\right\|_2^2. \tag{14}$$

[2] We make this assumption to make the computations tractable.

Algorithm 1 Robust Graph Convolutional Network (RGCN)

Input: Graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, Feature Matrix $\mathbf{X}$, Number of Layers L, Dimensionalities $\{f_1, ..., f_L\}$, Hyper-parameters $\{\gamma, \beta_1, \beta_2\}$, Task Specific Loss $\mathcal{L}_{cls}$

Output: Latent representations $\{\mathbf{z}_i\}_{i=1}^{N}$ and RGCN parameters $\mathbf{\Theta} = \{\mathbf{W}_{\mu}^{(i)}, \mathbf{W}_{\sigma}^{(i)}\}_{i=0}^{L-1}$

Initialize all parameters $\mathbf{\Theta}$
while $\mathcal{L}$ does not converge **do**
  Calculate $\mathbf{M}^{(1)}, \Sigma^{(1)}$ using Eq. (10)
  for l $\leftarrow$ 2 to L **do**
    Calculate $\mathbf{M}^{(l)}, \Sigma^{(l)}$ using Eq. (9)
  end for
  Sample $\mathbf{Z}$ from Gaussian distributions $\mathbf{H}^{(L)}$ using Eq. (11).
  Calculate $\mathcal{L}$ using Eq. (15)
  Update $\mathbf{\Theta}$ using back-propagation
end while

Finally, we jointly minimize the loss function by combining the above terms:

$$\mathcal{L} = \mathcal{L}_{cls} + \beta_1 \mathcal{L}_{reg1} + \beta_2 \mathcal{L}_{reg2}, \tag{15}$$

where β_1 and β_2 are hyper-parameters that control the impact of different regularizations.

4.4 Optimization and Complexity Analysis

We show the pseudo-code of RGCN in Algorithm 1. By using the "reparameterization trick" introduced in the last subsection, our model can be trained end-to-end using back-propagation, and thus we can use gradient descent to optimize the model. The time complexity of our method is $O\left(M\sum_{i=0}^{L} f_l + N\sum_{i=1}^{L} f_{i-1}f_i\right)$ where $N = |\mathbf{V}|$ is the number of nodes, $M = |\mathbf{E}|$ is the number of edges and f_l is the dimensionality of the l^{th} hidden layer, i.e. our method is linear with respect to the number of nodes and number of edges in the graph respectively, which is in the same order as other GCNs. Note that since our method also follows the "message-passing" framework, many sampling strategies for GCNs can be directly applied, e.g. [4, 5, 14, 33].

5 EXPERIMENTS

In this section, we empirically evaluate the effectiveness of our proposed method. We first introduce the experimental settings and then present our experimental results. Some additional experiment details can be found in the appendix.

5.1 Experimental Settings

5.1.1 Baselines and Adversarial Attack Methods. To evaluate the robustness of RGCN, we compare it with two state-of-the-art GCN models:

- GCN [18]: As introduced in Section 3.2 , this is the original GCN model which defines graph convolution as aggregating features from neighborhoods.
- GAT [30]: This model enhances GCN by introducing multi-head self-attention to assign different weights to different neighbors.

We compare these two GCNs and our proposed RGCN under three attacking methods:

- Random Attack: We randomly generate fake edges and add them into the graph. We regard this method as an illustrating example of non-targeted attacks.
- RL-S2V [7] [3]: This method generates adversarial attacks on graph-structured data based on reinforcement learning. It is designed for evasion and targeted attacks and can only perform direct attacks.
- NETTACK [37]: This method also generates adversarial perturbations targeting GCNs. It is designed for targeted attacks and can perform both direct and influence attacks, which we denote as NETTACK-Di and NETTACK-In respectively. We focus on the poisoning attack of NETTACK since it better matches the transductive node classification setting.

For all attacking methods, we focus on changing graph structures since it is more related to the graph setting, but our method can be directly applied to changing node attributes as well.

5.1.2 Datasets. In order to comprehensively evaluate the effectiveness of our proposed method, we adopt three citation networks commonly used in previous works [18, 30]: Cora, Citeseer and Pubmed [24], where nodes represent documents and edges represent citations. Nodes are also associated with bag-of-words features and corresponding ground-truth labels. The statistics of the datasets are summarized in Table 1.

We closely follow the experimental setting in previous works [18, 30]. Specifically, we use all node features and 20 labels per class as the training set and another 500 labels for validation and early-stopping. The performance of different methods is evaluated on a separate test set of 1000 labels. We adopt the same dataset splits as in [18] and report the average results of 10 runs.

Table 1: Statistics of datasets. $|V|$ denotes the number of nodes, $|E|$ denotes the number of edges, $|C|$ denotes the number of classes and $|F|$ denotes the number of features.

	Cora	Citeseer	Pubmed
$\|V\|$	2,708	3,327	19,717
$\|E\|$	5,429	4,732	44,338
$\|C\|$	7	6	3
$\|F\|$	1,433	3,703	500

5.1.3 Parameter Settings. In experiments, we set the number of layers as two for all methods as suggested by previous works [18, 30]. For GCN and RGCN, we set the number of hidden units as 32. Note that our method RGCN has both mean and variance vectors. For a fair comparison, we set their length as 16 so that the total number of hidden units matches 32. For GAT, we use 8 attention heads with each head containing 8 features, i.e. a total of 64 hidden units as suggested by the authors.

For RGCN, the hyper-parameters are set as follows: $\gamma = 1, \beta_1 = 5 \cdot 10^{-4}, \beta_2 = 5 \cdot 10^{-4}$ on all datasets. We set the dropout rate for RGCN as 0.6 and use Xavier initialization [11] for all weight

[3]The original paper proposes three different attack methods under different settings. We choose the most effective method, RL-S2V, in our experiments.

matrices. The non-linear activation is $ELU(\cdot)$ [6] and $ReLU(\cdot)$ [21] for means and variances respectively since variances are required to be non-negative while means can take negative values. For the optimization, we use Adam [16] with a fixed learning rate of 0.01 and set epoch number as 200 with early stopping on the validation set. For GCN and GAT, we use the default optimization settings in the authors implementations.

5.2 Node Classification on Clean Datasets

To build a reference line, we first conduct experiments on the clean datasets, i.e. datasets that are not attacked. The experimental results are shown in Table 2. We can see that our proposed method RGCN slightly outperforms the baseline methods on Pubmed, while having comparable performance on Cora and Citeseer. The strong performance of RGCN on clean datasets shows that our proposed Gaussian-based graph convolution is as effective as traditional graph convolutions in capturing the graph structure, which lays the foundation for applying it to the adversarial settings.

Table 2: The results of node classification accuracy (in percentages) on clean datasets.

	Cora	Citeseer	Pubmed
GCN	81.5 ± 0.5	70.9 ± 0.5	79.0 ± 0.3
GAT	**83.0 ± 0.7**	**72.5 ± 0.7**	79.0 ± 0.3
RGCN	82.8 ± 0.6	71.2 ± 0.5	**79.1 ± 0.3**

5.3 Against Non-targeted Adversarial Attacks

In this section, we first evaluate the classification accuracy of different methods against non-targeted adversarial attacks, i.e. perturbations that aim to reduce the overall classification accuracy of the model. To the best of our knowledge, there is no publicly available non-targeted attack method. Thus we use Random Attack as an illustrating example of the non-targeted attack method.

Specifically, we focus on adding edges and performing poisoning attacks, i.e. we randomly choose some node pairs without edges as noise edges to add into the graph, train different GCN methods on the modified graph and evaluate the classification accuracy on the test set. We vary the ratio of noise edges to clean edges from 0.1 to 1 and report the experimental results in Figure 2.

The figure shows that RGCN consistently outperforms both baselines on all datasets, demonstrating that we can improve the robustness of GCNs under non-targeted attacks. By using Gaussian distributions as hidden representations and assigning variance-based attention weights to neighborhoods, RGCN can absorb the effects of noise edges and is thus less sensitive to such adversarial information. On the other hand, although GAT also has attention mechanisms and achieves the best results on the clean Cora and Citeseer dataset, the performance of GAT drops rapidly, indicating that GAT is very sensitive to adversarial attacks and its attention mechanism cannot adapt to prevent the adversarial attacks.

5.4 Against Targeted Adversarial Attacks

In this section, we continue to evaluate the node classification accuracy of different methods against targeted adversarial attacks. Specifically, we first use adversarial attack methods to generate different numbers of perturbations for the targeted nodes, where one perturbation is defined as adding or deleting one edge in the graph. Then, we either retrain the GCN model for poisoning attacks (i.e. when use NETTACK) or keep the GCN model unchanged for evasion attacks (i.e. when use RL-S2V). Finally, we test the performance of different GCNs on the targeted nodes, i.e. whether we successfully defend the attacks. Considering that adversarial attacks are designed to be unnoticeable and high-degree nodes usually have richer values, we set the targeted nodes as all nodes in the test set with degree larger than 10. For Cora and Citeseer, we repeat the above process for all targeted nodes. For Pubmed, since running the adversarial methods for all targeted nodes is very time consuming, we randomly sample 10% of them.

Next, we show the results when adopting different attack methods in the following subsections.

5.4.1 RL-S2V. In this subsection, we report the results when adopting RL-S2V as the attack method. Recall that RL-S2V is an evasion attack and can only perform direct attacks. We use the default parameter settings of RL-S2V in the authors implementation. The experimental results are shown in Figure 3.

The results show that RGCN again consistently achieves better results compared to all the baselines, demonstrating the robustness of RGCN under targeted attacks. Note that RGCN does not utilize any information of the attack method, including which nodes are the targets. So rather than specifically "defending" certain attacks, the architecture of RGCN actually improves the overall robustness of the model and is general to any adversarial attack strategy, which is desirable in practice since we may not know the attack method or which nodes are the targets in advance.

5.4.2 NETTACK. In this section, we adopt NETTACK as the attack method and keep all the default parameter settings in the authors implementation. Since NETTACK can handle both direct and influence attacks, we report the results of NETTACK-Di and NETTACK-In in Figure 4 and Figure 5 respectively. In Figure 4, we focus on the first 5 perturbations because more perturbations will lead to too low performance for all methods which is unbearable in practice.

From Figure 4, we can see that the performance of all methods decays rapidly with respect to the number of perturbations, demonstrating that NETTACK-Di is a very strong attack method. Despite that, we can see that RGCN is still consistently more robust than both baselines on all datasets. On the other hand, for influence attacks shown in Figure 5, all methods have relative higher accuracy, proving that influence attacks are usually less effective than direct attacks, Nevertheless, our method still outperforms all baselines.

The experimental results show that no matter how strong the attacks are, RGCN consistently outperforms the baselines, demonstrating that our proposed architecture can prevent GCNs from various targeted attack strategies.

5.5 Parameter Analysis

In this section, we conduct some parameter analysis to further investigate the reasons behind the robustness of RGCN.

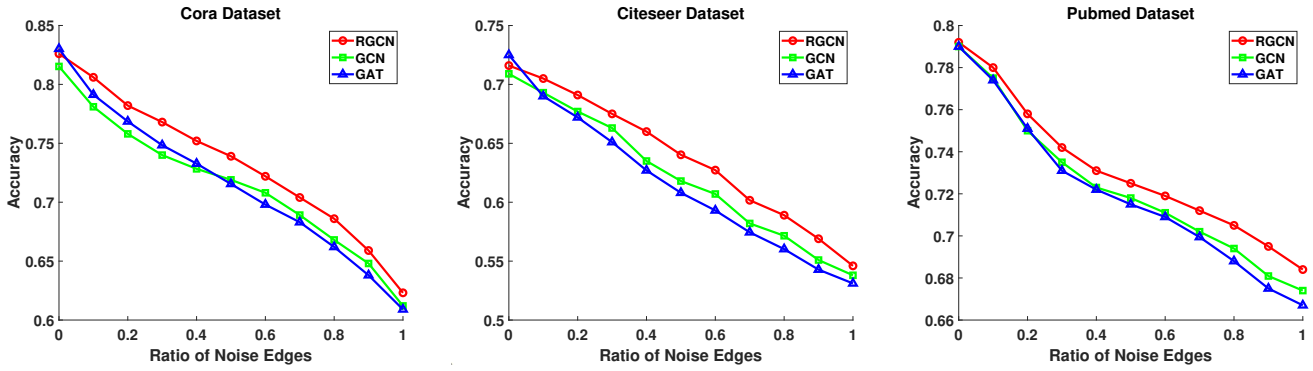

Figure 2: Results of different methods when adopting Random Attack as the attack method.

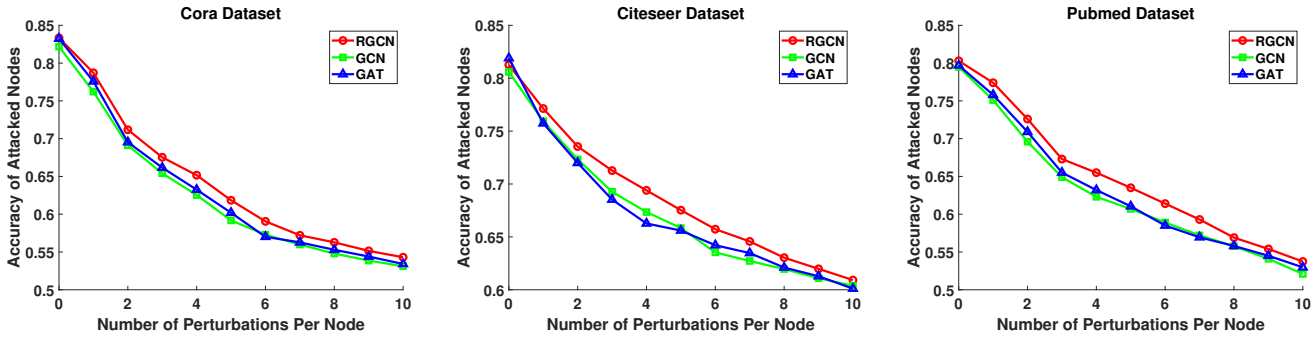

Figure 3: Results of different methods when adopting RL-S2V as the attack method.

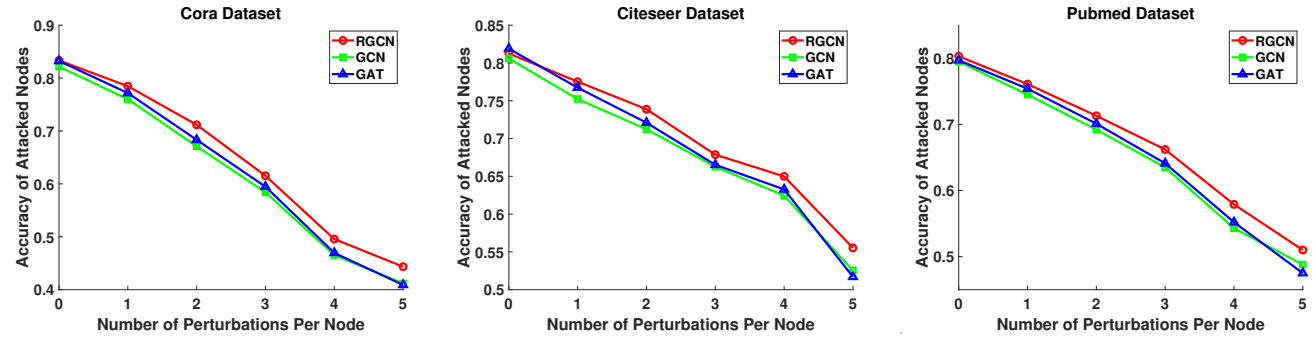

Figure 4: Results of different methods when adopting NETTACK-Di as the attack method.

5.5.1 Analysis of Variances. While designing the model, we claim that using Gaussian distributions as hidden layers can absorb the effects of adversarial attacks in the variances to enhance robustness. To verify that, we analyze how the variances change when the graph is attacked. Specifically, we follow the experimental setting in Section 5.4 and plot the variances of the attacked nodes with respect to the number of perturbations. For brevity, we only report the variances of the first layer, i.e. $\boldsymbol{\sigma}^{(1)}$, when using NETTACK-Di as the attack model, but similar patterns are observed in other cases.

From Figure 6a, we can see that with the number of perturbations increasing, the variances also increase significantly, which is in accordance with our intuition. By absorbing the effects of adversarial attacks in variances, RGCN is more robust than only using plain vectors. This also lays the foundation for using variance-based attention weights.

5.5.2 Analysis of Imposing Variance-based Attention Weights. Moreover, we also conduct some experiments to analyze whether imposing variance-based attention weights in our model can help remedy the propagation of adversarial attacks. Specifically, we follow the experimental setting of non-targeted attacks in Section 5.3 and vary γ, the hyper-parameter in setting the attention weights. For the ease of presentation, we only show the results on Cora when setting

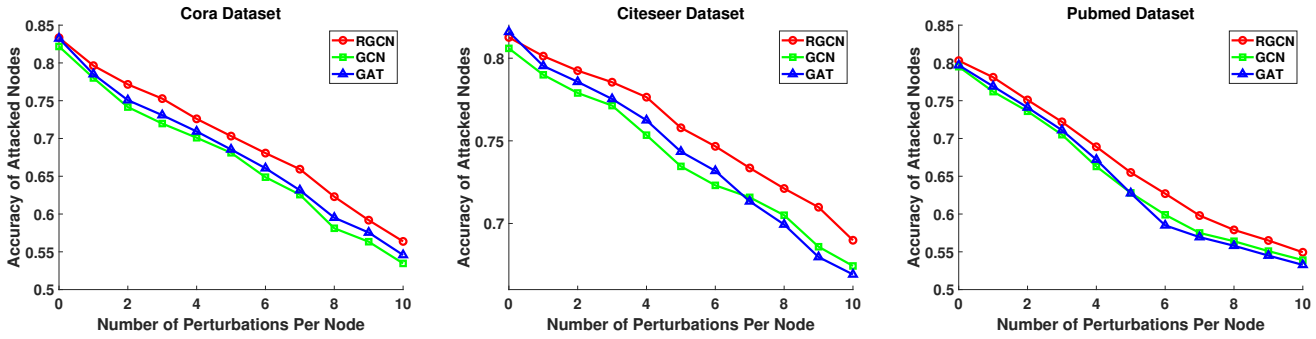

Figure 5: Results of different methods when adopting NETTACK-In as the attack method.

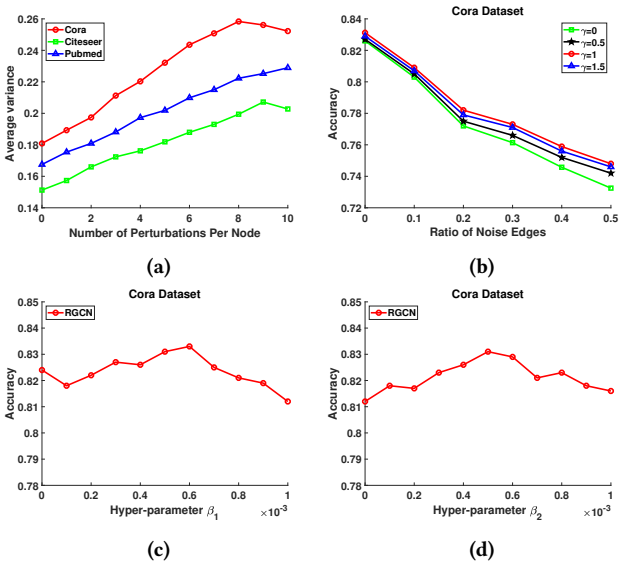

Figure 6: Results of parameter analysis: (a) the average variance of targeted nodes w.r.t the number of perturbations; (b) the parameter sensitivity with respect to γ; (c) the parameter sensitivity with respect to β_1; (d) the parameter sensitivity with respect to β_2.

the ratio of noise edges between 0 and 0.5, while other datasets show consistent results.

Figure 6b shows that imposing variance-based attention weights ($\gamma > 0$) outperforms not using attention ($\gamma = 0$), demonstrating that such attention mechanism can indeed improve the robustness of RGCN. Moreover, we can observe that the margin becomes larger as the ratio of noise edges increases, indicating the attention mechanism is more effective when there are more noise edges. Moreover, setting γ too large will degrade the performance, probably because the "message-passing" in real edges is also blocked. We find that uniformly setting $\gamma = 1$ works well in our experiments.

5.5.3 Analysis of Regularizations. In this section, we investigate the effectiveness of two regularization terms. More specifically, we evaluate how different values of hyper-parameter β_1 and β_2 affect our method. We report the results of node classification on clean dataset Cora, while other experiments exhibit similar patterns.

Figure 6c and Figure 6d show that choosing an appropriate value for both β_1 and β_2 can increase the accuracy of RGCN, which is in line with our expectations of regularization constraints. However, setting β_1 or β_2 too large will also hurt the performance. In our experiments, setting $\beta_1 = \beta_2 = 5 \cdot 10^{-4}$ on all datasets lead to satisfying results.

6 CONCLUSIONS

In this paper, we propose RGCN, a novel graph convolution model to explicitly enhance the robustness of GCNs against adversarial attacks. By using Gaussian distributions in hidden layers and introducing variance-based attention weights in aggregating node neighborhoods, our proposed method can effectively reduce the impacts of adversarial attacks. Experimental results demonstrate that our proposed method can consistently improve the robustness of GCNs under various adversarial attack strategies. To the best of our knowledge, this is the first study on this critical and challenging problem. Future directions include conducting more experiments besides citation networks. It is also interesting to extend this framework to other deep learning models on graphs and more complicated graph structures such as heterogeneous graphs or graphs with edge attributes.

ACKNOWLEDGMENTS

This work was supported in part by National Program on Key Basic Research Project (No. 2015CB352300), National Natural Science Foundation of China Major Project (No.U1611461), National Natural Science Foundation of China (No. 61772304, No. 61521002, No. 61531006), the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and the Young Elite Scientist Sponsorship Program by CAST. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *Proceedings of the 7th International Conference on Learning Representations.*

[2] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the 3rd International Conference on Learning Representations*.
[4] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: fast learning with graph convolutional networks via importance sampling. In *Proceedings of the 7th International Conference on Learning Representations*.
[5] Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *International Conference on Machine Learning*. 941–949.
[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *Proceedings of the 5th International Conference on Learning Representations*.
[7] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial Attack on Graph Structured Data. In *Proceedings of the 35th International Conference on Machine Learning*. 1115–1124.
[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.
[9] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).
[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *International Conference on Machine Learning*. 1263–1272.
[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 249–256.
[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.
[13] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.
[14] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive Sampling Towards Fast Graph Representation Learning. In *Advances in Neural Information Processing Systems*. 4563–4572.
[15] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design* 30, 8 (2016), 595–608.
[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.
[17] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[18] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 6th International Conference on Learning Representations*.
[19] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
[20] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled Graph Convolutional Networks. In *International Conference on Machine Learning*.
[21] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.
[22] Valentin Vladimirovich Petrov. 2012. *Sums of independent random variables*. Vol. 82. Springer Science & Business Media.
[23] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. (2018), 593–607.
[24] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.
[25] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* (2013).
[26] Martin Simonovsky and Nikos Komodakis. 2017. Dynamic edgeconditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[27] Lichao Sun, Ji Wang, Philip S Yu, and Bo Li. 2018. Adversarial Attack and Defense on Graph Data: A Survey. *arXiv preprint arXiv:1812.10528* (2018).
[28] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. 2018. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735* (2018).
[29] Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, and Wenwu Zhu. 2019. AutoNE: Hyperparameter Optimization for Massive Network Embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
[30] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 7th International Conference on Learning Representations*.
[31] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, P Yu, and Yanfang Ye. 2019. Heterogeneous Graph Attention Network. *arXiv preprint arXiv:1903.07293* (2019).
[32] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *International Conference on Machine Learning*.
[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
[34] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*.
[35] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2018. Deep Learning on Graphs: A Survey. *arXiv preprint arXiv:1812.04202* (2018).
[36] Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. 2018. Deep variational network embedding in wasserstein space. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2827–2836.
[37] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2847–2856.
[38] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *Proceedings of the 8th International Conference on Learning Representations*.

A ADDITIONAL EXPERIMENT DETAILS

A.1 Hardware and Software Configurations

All experiments are conducted on a server with the following configurations:

- Operating System: Ubuntu 18.04.1 LTS
- CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
- GPU: GeForce GTX TITAN X
- Software: Python 3.6.7, TensorFlow 1.12.0, SciPy 1.1.0, NumPy 1.15.4.

A.2 Baselines and Adversarial Attack Methods

We use the following publicly available implementation of baseline methods and adversarial attack methods:

- GCN: https://github.com/tkipf/gcn
- GAT: https://github.com/PetarV-/GAT
- RL-S2V: https://github.com/Hanjun-Dai/graph_adversarial_attack
- NETTACK: https://github.com/danielzuegner/nettack

A.3 Dataset Splits

We use the following publicly available dataset splits for all three datasets: https://github.com/tkipf/gcn/tree/master/gcn/data