ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200233

Abstract Interpretation Based Robustness Certification for Graph Convolutional Networks

Yang Liu[1,2] and **Jiaying Peng**[1,2] and **Liang Chen**[1,2,*] and **Zibin Zheng**[1,2]

Abstract. Graph convolutional networks (GCNs) have attracted much attention and become a powerful tool for graph data analysis. However, recent studies show that these methods are vulnerable to adversarial attacks (e.g. changing node attributes). Although several works have proposed to improve the robustness of GCNs, only a few works address their *provable robustness*. In this work, we propose a novel Abstract Interpretation (AI) based method for scalable robustness certification of graph convolutional networks. Different from the AI-based certification in the image classification task whose data is continuous, the considered perturbation on node attributes in this paper is binary. To address this challenge, our central idea is to over-approximate all possible perturbations of the first layer output instead of the input layer. Abstract transformers for graph convolutional operations are further defined to prove the robustness automatically. Experimental results on three public graph datasets demonstrate that our method is faster than the state-of-the-art certification approach.

1 Introduction

Many real-world applications are built on graph data including social networks [11], citation networks [3], heterogeneous information networks [2, 22] and so on. Among these applications, node classification is one of fundamental tasks which aims to predict the class label of nodes. Over the last few years, graph convolutional networks (GCNs) [10] have become increasingly popular and achieved state-of-the-art on the node classification. Multiple safety critical node classification tasks such as disease prediction [15] and malicious account detection [12] have adopted GCNs to make decisions. Therefore, it is important to ensure the predicted result is reliable.

Unfortunately, recent studies [26, 25, 5] show that GCNs are vulnerable to adversarial attacks. By slightly changing nodes' features or graph structures, two very similar nodes are classified into different classes. Worse still, since GCNs utilize neighbors' information to make predictions, even perturbing features of the target node's neighbors can lead to the wrong prediction [25]. The unreliable results offer the opportunity for attackers to exploit these vulnerabilities and restrict applications of GCNs.

In this work, we tackle a fundamental challenge: how to ensure that small changes to node features do not change its label? Specifically, given a trained GCN, a node is certifiably robustness if it is proved that no perturbations w.r.t. a certain space of perturbations can change the node's label (i.e., predicted result). Here we consider perturbations on node features and the data space is binary/discrete. Although several works [23, 24] have proposed to improve the robustness of graph convolutional networks such as adversarial training [23], few works have considered analyzing the provable robustness of the GCN. So far, there is only one work [27] studying the provable robustness of GCNs. To give robustness certificates to a node, they compute the worst case margin between the predicted class and all other classes achievable under admissible perturbations to the node attributes. If all the worst case margins are positive, then no adversarial examples (within the defined admissible perturbations) can change the prediction of the node. They show that the computation of a target node's margins can be interpreted as a backward pass on a sliced GCN which only contains the l-hop neighbors of the target node (suppose the GCN has l layers). However, such method suffers the efficiency problem. Although only a small number of nodes can not be certificated by the current verifier [27], according to their work, to certificate nodes of GCNs, the time complexity of current verifier is related to the number of nodes and edges in graph which is inefficient.

To address above problems, we propose to prove the robustness of graph convolutional networks via *abstract interpretation* (AI) [4] which has gained much attention and been adopted to certificate the robustness of fully connected and convolutional neural networks [6, 18, 17, 14] on the image classification task. Compared to the work [27], the AI-based verifier performs a "forward" pass through GCNs. The idea is to check the output of all possible perturbations. If all these outputs are classified into the same class of the target instance, then the input data (e.g. an image) is certifiably robust. Nevertheless, the set of possible perturbations is usually large, leading to high time complexity to check the robustness. To avoid this, abstract interpretation verifies the robustness on an overapproximation of the perturbation set, which is called abstract domain. The common choices of abstract domains include interval and zonotopes. The defined abstract domain is then propagated through the neural network to obtain an overapproximation of all possible outputs. Finally the robustness is checked on the overapproximation outputs. If the lower bounds between the predicted class and other classes are positive, then the input data is certifiably robustness since even in the worst case the value of the predicted class is higher then other classes.

Based on the abstract interpretation, we can improve the efficiency of the verifier. Abstract domains and transformers for GCNs are defined to prove the robustness of GCNs. Instead of replacing the ℓ_0 constraints perturbations to ℓ_1 constraints, the abstract domains of the first layer's output is proposed to overapproximate the effect of all possible perturbations. The interval domain is employed to measure the range of each component of the first layer's output under

[1] School of Data and Computer Science, Sun Yat-sen University, China, email: {liuy296, pengjy36}@mail2.sysu.edu.cn, {chenliang6, zhzibin}@mail.sysu.edu.cn, *corresponding author
[2] National Engineering Research Center of Digital Life, Sun Yat-sen University, China.

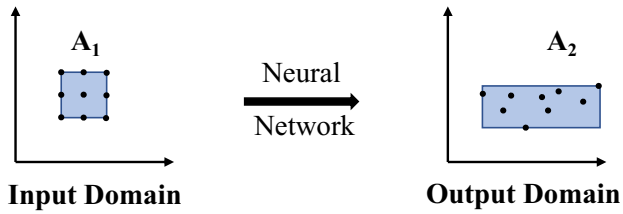

Figure 1. The illustration of abstract interpretation. The points denote the real elements and the rectangles (i.e. A_1 and A_2) represent the interval domain which overapproximate the real elements.

perturbations. These bounds are then transformed via the GCN to obtain the intervals of the predication layer. To further enhance efficiency, the matrix-form propagation rule is proposed to transform the domains of all nodes. We first define the lower and upper bound matrices (i.e. input abstract domain for all nodes) and demonstrate that the abstract domains of the entire nodes can be transformed together without slicing the input data. The experimental results show that our methods spend less time to give robustness certificates to nodes compared to the state-of-the-art verifier.

To summarize, this paper makes the following contributions.

- We propose a novel method for the provable robustness of GCNs based on abstract interpretation.
- We present a matrix-form propagation rule to efficiently certificate the robustness of nodes.
- Extensive experiments conducted on three public graph datasets demonstrate our methods spend less time to verify the robustness of GCNs.

2 Preliminaries

In this section, two important concepts of the abstract interpretation (i.e. the abstract domain and abstract transformer) are firstly presented. Then we elaborate the formulation of GCNs and define the problem of robustness verification. For notations, we use bold uppercase letters to denote matrices (e.g., $\boldsymbol{W}$), bold lowercase letters to denote vectors (e.g., $\boldsymbol{w}$), and non-bold letters to denote scalars or indices (e.g., w). The uppercase calligraphic symbols (e.g., $\mathcal{W}$) stand for sets.

Abstract Interpretation Let $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ denote the GCN with d input features and k output classes. Note that perturbations can either be added on x's and its neighbors' features. We employ $P_q(\boldsymbol{X})$ be the set of all $q \in \mathbb{N}$ possible binary perturbations for the target node $\boldsymbol{x} \in \mathbb{R}^d$. q is the perturbation budget. The goal is to verify whether f assigns the same class of the target node to all possible perturbed feature matrices $\hat{\boldsymbol{X}} \in P_q(\boldsymbol{X})$. The challenge is that it is not practical to enumerate all points in $P_q(\boldsymbol{X})$ to verify the robustness. To remedy this, abstract interpretation is adopted to obtain sound, computable, and precise finite approximations of these potentially infinite sets of perturbed points.

Two components are needed to define for abstract interpretation: (1) a suitable abstract domain $\mathcal{D}$ which overapproximates the set $P_q(\boldsymbol{X})$. There are multiple choices of the abstract domain such as Interval (also called Box), Zonotope, and Polyhedral domain [6]. In our work, the interval domain is adopted which is significantly faster than the zonotope and polyhedral domain [14]. Moreover, since the major operations in GCNs are multiplication and addition, it is efficient for the transformation of Interval domains to parallelize on GPU [14]. The interval domain represents a d-dimensional box which consists of constraints of the form $s < x_i < t$ where $s, t \in \mathbb{R}$ is the lower and upper bound of the interval and x_i is the i-th component of $\boldsymbol{x}$. (2) abstract transformers which overapproximates the effect of the GCN f. Abstract transformers are applied on abstract domains to obtain an output abstract domains which capture transformations of the GCN function f. Thus, the robustness can be checked on the abstract domain of final layer returned by abstract transformers.

Figure 1 is a high-level illustration of abstract interpretation where the area A_1 and A_2 represent the input and output abstract domain. The points represent all the possible inputs and outputs under admissible perturbations, respectively. From the figure, we can observe that both A_1 and A_2 overapproximate the real perturbation set and the neural network will change the shape of abstract domains. Therefore, if we can prove that all points in entire area A_2 are the same class, then the input data is certifiably robustness w.r.t. the model.

Graph Convolutional Networks Follow the previous work [27], we introduce our method under the context of node classification. The purpose of graph convolutional networks is to leverage the structure information into the neural networks. A GCN could be formally formulated as follows: Given an undirected attributed graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ that has n nodes, with $\boldsymbol{A} \in \{0,1\}^{n \times n}$ denoting the adjacency matrix and $\boldsymbol{X} \in \{0,1\}^{n \times d}$ representing the nodes' binary feature where d is the feature dimension, the l-th hidden layer of GCN [10] is defined as:

$$\boldsymbol{H}^{(l)} = \sigma(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l-1)}), \tag{1}$$

where $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}$ and $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is the identity matrix. $\boldsymbol{D}$ represents the degree matrix where the i-th entry $d_{ii} = \sum_j a_{ij}$, $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d^{(l-1)} \times d^{(l)}}$ is a trainable weight matrix of layer l which learns useful features and transforms the embedding size from $d^{(l-1)}$ to $d^{(l)}$, and $\sigma(\cdot)$ denotes the activation function, which we set as ReLU [7] in this paper. Initially, $\boldsymbol{H}^{(0)} = \boldsymbol{X}$. The GCN is learned by minimizing the cross-entropy loss on given training label nodes.

Problem Definition Similar to the previous work, we consider the situation that the input node features are discrete/binary. To certificate the robustness of a node, margins between the predicted class and other classes are needed to compute. Formally, the problem are defined as follows:

Problem 1 *Given a graph $\mathcal{G}$, a trained model f, a target node t, a perturbation budget q, let y_t denotes the predicted class of node t and $\boldsymbol{h}^t$ denotes the model prediction for node t. The worst-case margin between class y_t and class c under all possible perturbations $P_q(\boldsymbol{X})$ is:*

$$m^t_{y_t,c} = \min_{\hat{\boldsymbol{X}} \in P_q(\boldsymbol{X})} h^t_{y_t} - h^t_c \tag{2}$$

If $m^t_{y_t,c} > 0$ for all $c \neq y_t$, the node t is certifiably robust w.r.t. the model f and the perturbation space $P_q(\boldsymbol{X})$.

From Problem 1, if all $m^t_{y_t,c}$ (the predicted class and other classes) are positive, there is no adversarial example in perturbation space $P_q(\boldsymbol{X})$ can changes the prediction of model f.

3 Method

In this section, we elaborate how to obtain the interval of $m^t_{y_t,c}$ in Problem 2. Specifically, $m^t_{y_t,c}$ can be computed as follows:

$$m^t_{y_t,c} = \min_{\hat{\boldsymbol{X}} \in P_q(\boldsymbol{X})} \hat{\boldsymbol{A}}\hat{\boldsymbol{H}}\boldsymbol{\delta}, \tag{3}$$

where $\hat{\boldsymbol{H}}$ denote the output before the prediction layer and $\boldsymbol{\delta} = \boldsymbol{w}_{y_t} - \boldsymbol{w}_c$. $\boldsymbol{w}_c$ is the c-th column of the final layer matrix. The lower bound of $m^t_{y_t,c}$ can be computed via interval arithmetic [6]. We first introduce how to define the input interval (i.e. abstract domain) and then show that how these intervals transform through a GCN layer.

3.1 Specifying the Abstract Domain

In the image classification task, the perturbations are often considered as ℓ_∞-ball of radius ϵ. Thus, the abstract domain for a point $\boldsymbol{x}$ (e.g. an image.) can be defined as $\{x_i - \epsilon \leq x_i \leq x_i + \epsilon\}_{i=1}^d$. However, such a solution is not work for the considered data whose perturbation is binary (i.e. $\{-1, 1\}$). If the method is applied to the input features, the resulting perturbation space is $\{0 \leq x_i \leq 1\}_{i=1}^d$ which contains all possible inputs and leads to an inaccurate domain. To address this problem, instead of directly specifying the abstract domain on input features, we measure each output component's intervals of the first GCN layer. We first show how to compute a specific component h of the first layer's hidden representation of a certain node v, then we measure the range of h. To better illustrate our method, the computation of h is split into two steps – aggregating the features of the node and its neighbors and transforming the aggregated feature to a hidden representation.

$$\boldsymbol{x}_{\text{agg}} = \sum_{v' \in \mathcal{N}(v)} \hat{a}_{vv'} \boldsymbol{x}_{v'}, \tag{4}$$

$$h = \boldsymbol{x}_{\text{agg}} \boldsymbol{w}, \tag{5}$$

where $h \in \mathbb{R}$ is one of the components of node v's first layer output and $\boldsymbol{w} \in \mathbb{R}^{d \times 1}$ is the corresponding column of weight matrix $\boldsymbol{W}^{(0)}$. $\mathcal{N}(v)$ denotes the set of the node v's neighbors and v itself. $\hat{a}_{vv'}$ is the weight between the node v and v'. Specifically, the weight is defined as follows:

$$\hat{a}_{vv'} = \frac{1}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(v')|}} \tag{6}$$

Note that $\hat{a}_{vv'}$ is always positive. Here we omit the ReLU activation function and discuss it later in section 3.2.

After obtaining the result of h, we can measure the range of it now. The insight behind our method is that perturbations on binary features can only change the zero component to one or change the one component to zero. Therefore, the change of h w.r.t a perturbation on a specific component (node v's or its neighbor's) can be measured according to equation (4) and (5). Let x_i denotes the perturbed component on node v_p, then the change of h is:

$$\delta = \begin{cases} \hat{a}_{vv_p} \times w_i & x_i = 0 \\ -\hat{a}_{vv_p} \times w_i & x_i = 1 \end{cases}, \tag{7}$$

where w_i represents the corresponding i-th component of $\boldsymbol{w}$. With this definition, the upper (lower) bound of h under q perturbation is h plus the sum of top-q maximum (minimum) changes:

$$h + q\_\text{smallest}\_\delta \leq h \leq h + q\_\text{largest}\_\delta \tag{8}$$

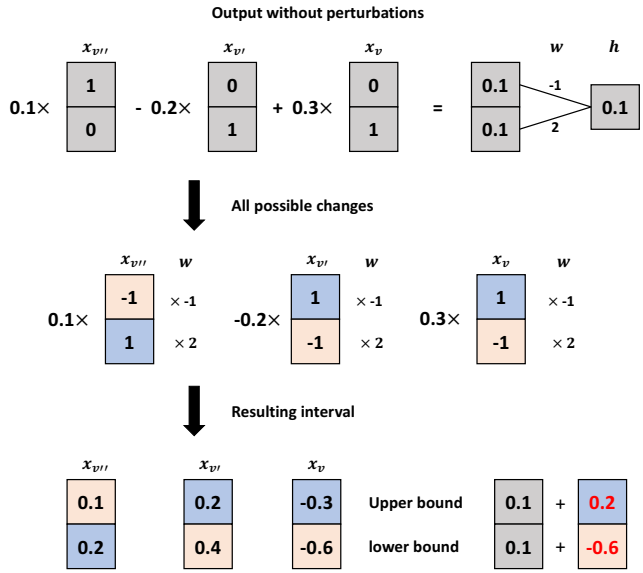

Figure 2. A running example of how to measure the range of component h. The perturbation budget q is equal to 1.

To further reduce the computation budget, we first treat all components of a node and its neighbors' features perturbable. Then the sum of top-q maximum (minimum) changes is relaxed by q times the maximum (minimum) changes. The resulting abstract domain can be specified as follows:

$$h + q \times \delta_{min} \leq h \leq h + q \times \delta_{max}, \tag{9}$$

where δ_{min} is the minimum of all possible changes. Analogously, δ_{max} is the maximum of all possible changes. Therefore, to compute the interval of h, we only need to find the maximum (minimum) of $\hat{\boldsymbol{a}}$ and $\boldsymbol{w}$.

Figure 2 displays a running example where the number of perturbations $q = 1$. As can be seen from the figure, without the perturbation, the output h is equal to 0.1. The second row illustrates the computation of h's possible changes. The blue components denote perturbations changed from 0 and the orange components denote perturbations changed from 1. The last row demonstrates the results of maximum and minimum changes and the resulting interval. As we can see, the lower bound is $0.1 - 0.6 = -0.5$ and the upper bound is $0.1 + 0.2 = 0.3$. Note that it is unnecessary to compute all changes in implementation. The computation process is shown to better illustrate our method.

3.2 Abstract Transformers for GCN

Suppose we have obtained the node v's intervals $\{s_i^{(l)} \leq h_i^{(l)} \leq t_i^{(l)}\}_{i=1}^{d^{(l)}}$ of the layer l's hidden representations $\boldsymbol{h}^{(l)} \in \mathbb{R}^{d'}$, the goal is to compute the next layer's intervals $\{s_i^{(l+1)} \leq h_i^{(l+1)} \leq t_i^{(l+1)}\}_{i=1}^{d^{(l+1)}}$ so that after specifying the abstract domain of the first layer, the intervals of predication layer can be computed automatically. Similar to section 3.1, the computation of $h_i^{(l+1)} \in \mathbb{R}^{d''}$ is

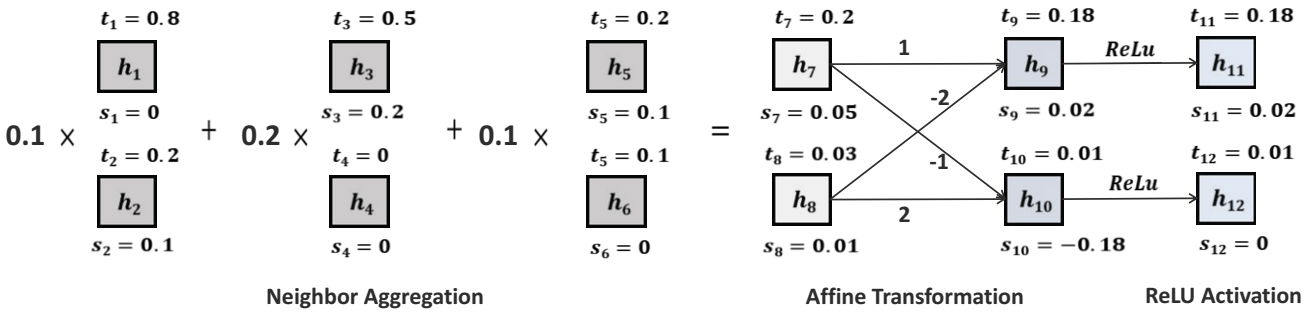

Figure 3. A running example of three define abstract transformers for neighbor aggregation, affine transformation, and ReLU activation. For better illustration, we use number subscripts to denote each component and its lower and upper bounds.

split into three steps:

$$\boldsymbol{h}_{\text{agg}}^{(l)} = \sum_{v' \in \mathcal{N}(v)} \hat{a}_{vv'} \boldsymbol{h}_{v'}^{(l)}, \tag{10}$$

$$h_{\text{affine}}^{(l)} = \boldsymbol{h}_{\text{agg}}^{(l)} \boldsymbol{w}^{(l)}, \tag{11}$$

$$h^{(l+1)} = \text{ReLU}(h_{\text{affine}}^{(l)}), \tag{12}$$

where $\boldsymbol{h}_{\text{agg}}^{(l)} \in \mathbb{R}^{d^{(l)}}, h_{\text{affine}}^{(l)}, \ h^{(l+1)} \in \mathbb{R}$ represents the hidden representation after neighbor aggregation, affine transformation, and ReLU activation, respectively. $\boldsymbol{w}^{(l)} \in \mathbb{R}^{d^{(l)} \times 1}$ is a certain column of the l-th layer weighted matrix $\boldsymbol{W}^{(l)}$. Here the subscript i is omitted for simplicity. Now we define three abstract transformers which corresponding to the above computation steps.

Neighbor Aggregation Abstract Transformer The neighbor aggregation step is equal to the weighted addition of a node and its neighbors' representations. Since the weight $\hat{a}_{vv'}$ is always greater than zero, the neighbor aggregation abstract transformer can be defined as follows:

$$s_{\text{agg}}^{(l)} = \sum_{v' \in \mathcal{N}(v)} \hat{a}_{vv'} s_{v'}^{(l)}, \tag{13}$$

$$t_{\text{agg}}^{(l)} = \sum_{v' \in \mathcal{N}(v)} \hat{a}_{vv'} t_{v'}^{(l)}, \tag{14}$$

where $s_{\text{agg}}^{(l)}$ and $t_{\text{agg}}^{(l)}$ is the lower and upper bound after aggregation.

Affine Abstract Transformer Let $w_j^{(l)} \in \mathbb{R}$ denotes the j-th component of $\boldsymbol{w}^{(l)}$. Then, the lower and upper bounds of $h_{\text{affine}}^{(l)}$ after the transformation are:

$$s_{\text{affine}}^{(l)} = \sum_{1 \le j \le d^{(l)}} (max(0, w_j^{(l)}) \times s_{\text{agg}}^{(l)} + min(w_j^{(l)}, 0) \times t_{\text{agg}}^{(l)}), \tag{15}$$

$$t_{\text{affine}}^{(l)} = \sum_{1 \le j \le d^{(l)}} (max(0, w_j^{(l)}) \times t_{\text{agg}}^{(l)} + min(w_j^{(l)}, 0) \times s_{\text{agg}}^{(l)}), \tag{16}$$

where d' represents the dimension size of $\boldsymbol{w}^{(l)}$.

ReLU Abstract Transformer For each input $x \in \mathbb{R}$, the rectified linear unit (ReLU) activation function is defined as $\text{ReLU}(x) = \max(0, x)$. We apply the ReLU function to each component of the above transformed results.

$$[s^{(l+1)}, t^{(l+1)}] = \begin{cases} [s_{\text{affine}}^{(l)}, t_{\text{affine}}^{(l)}] & s_{\text{affine}}^{(l)} \ge 0 \\ [0, 0] & t_{\text{affine}}^{(l)} \le 0 \\ [0, t_{\text{affine}}^{(l)}] & s_{\text{affine}}^{(l)} < 0, t_{\text{affine}}^{(l)} > 0 \end{cases} \tag{17}$$

In the implementation, the bounds $s_{\text{affine}}^{(l)}$ and $t_{\text{affine}}^{(l)}$ are directly fed to a ReLU function to generate the next layer's bounds. Note that for the prediction layer, the ReLU function is omitted.

Figure 3 is a simple running example of the entire transformation where the feature dimension size is 2. As the figure shows, the lower bound s_7 and upper bound t_7 are firstly aggregated by specific weights:

$$\begin{aligned} s_7 &= 0.1 \times s_1 + 0.2 \times s_3 + 0.1 \times s_5 \\ t_7 &= 0.1 \times t_1 + 0.2 \times t_3 + 0.1 \times t_5 \end{aligned} \tag{18}$$

The bounds of h_8 are aggregated in the same way. Then these bounds are transformed by a weighted matrix:

$$\begin{aligned} s_7 - 2 \times t_8 \le h_9 \le t_7 - 2 \times s_8 \\ -1 \times t_7 + 2 \times s_8 \le h_{10} \le -1 \times s_7 + 2 \times t_8 \end{aligned} \tag{19}$$

Theses bounds are fed to a ReLU activation function to generate the next layer's bound:

$$\begin{aligned} \text{ReLU}(s_9, 0) \le h_{11} \le \text{ReLU}(t_9, 0) \\ \text{ReLU}(s_{10}, 0) \le h_{12} \le \text{ReLU}(t_{10}, 0) \end{aligned} \tag{20}$$

Note that if this layer is the prediction layer, the intervals of $m_{y_t,c}^t$ can be obtained. If the lower bound of $m_{y_t,c}^t$ is positive, then no perturbations in the considered perturbation space can change the prediction from class y_t to c. The target node is certifiably robust if $m_{y_t,c}^t$ is positive between y_t and all other class c.

3.3 Matrix-Form Abstract Interpretation

To offer a holistic view of the abstract interpretation for GCNs, we follow the previous work [10] and provide a matrix-form of overall steps. The central idea is to maintain all nodes' bounds of each layer. Firstly, the initial lower bound matrix $\boldsymbol{S} \in \mathbb{R}^{n \times d}$ and upper bound matrix $\boldsymbol{T} \in \mathbb{R}^{n \times d}$ are constructed according to Section 3.1. Then

these matrices are fed to the ReLU activation function to obtain the interval of the first layer output.

$$\boldsymbol{S}^{(1)} = \text{ReLU}(S), \tag{21}$$

$$\boldsymbol{T}^{(1)} = \text{ReLU}(T), \tag{22}$$

where $\boldsymbol{S^{(1)}} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{T}, \boldsymbol{T^{(1)}} \in \mathbb{R}^{n\times d}$ represent the matrices of all nodes' lower and upper bounds before and after ReLU activation, respectively. $\hat{\boldsymbol{a}}_{\text{max}} \in \mathbb{R}^{n\times 1}$ and $\hat{\boldsymbol{a}}_{\text{min}} \in \mathbb{R}^{n\times 1}$ are the column-wise maximum and minimum elements. Similarly, $\boldsymbol{w}_{min}, \boldsymbol{w}_{max} \in \mathbb{R}^{1\times d}$ denote the row-wise minimum and maximum elements of $\boldsymbol{W}^{(1)}$, respectively.

Follow the description of section 3.2, after obtaining the matrix-form bounds of layer l, we aim to compute the bounds of the next layer. The corresponding matrix-form neighbor aggregation abstract transformer is defined as:

$$\boldsymbol{S}^{(l)}_{\text{agg}} = \hat{\boldsymbol{A}}\boldsymbol{S}^{(l)}, \tag{23}$$

$$\boldsymbol{T}^{(l)}_{\text{agg}} = \hat{\boldsymbol{A}}\boldsymbol{T}^{(l)}, \tag{24}$$

where $\boldsymbol{A}$ is the adjacency matrix. Next the corresponding matrix-form affine abstract transformer is:

$$\boldsymbol{S}^{(l)}_{\text{affine}} = \boldsymbol{S}^{(l)}_{\text{agg}} \times \max(\boldsymbol{W}^{(l)}, 0) + \boldsymbol{T}^{(l)}_{\text{agg}} \times \min(\boldsymbol{W}^{(l)}, 0) \tag{25}$$

$$\boldsymbol{T}^{(l)}_{\text{affine}} = \boldsymbol{T}^{(l)}_{\text{agg}} \times \max(\boldsymbol{W}^{(l)}, 0) + \boldsymbol{S}^{(l)}_{\text{agg}} \times \min(\boldsymbol{W}^{(l)}, 0) \tag{26}$$

Here, the min and max function represent the element-wise minimum and maximum which does not change the shape of $\boldsymbol{W}^{(l)}$. Finally, if the l-th layer is not the prediction layer, the ReLU abstract transformer is applied to the generated matrix-form bounds:

$$\boldsymbol{S}^{(l+1)} = \text{ReLU}(\boldsymbol{S}^{(l)}), \tag{27}$$

$$\boldsymbol{T}^{(l+1)} = \text{ReLU}(\boldsymbol{T}^{(l)}), \tag{28}$$

where $\boldsymbol{S}^{(l+1)}$ and $\boldsymbol{T}^{(l+1)}$ are the next layer's lower and upper bound matrix. Finally, the robustness of GCNs can be checked on output lower bound matrices.

4 Experiments

4.1 Experimental Settings

Dataset The experiments are conducted on three widely used datasets: Cora_ML [13], Citeseer [16], and PubMed [16]. Dataset statistics are summarized in Table 1. Following the previous work [27], the datasets are split into 10% training labeled and 90% testing unlabeled nodes.

Baseline and metric We compare our method with the only existing robustness certification [27] whose code[3] is provided by authors. Different from the work [27], we jointly consider the local (perturb features of the target node) and global (perturb features of all nodes) perturbations. Through setting the same value of local and global perturbation budget of the existing certificate [27], we obtain the same problem in our work. We refer our method built on Equation (8) and Equation (9) as Ours-1 and Ours-2. To compare the robustness of models, we display the average of nodes' largest q that they can be certifiably robust.

[3] https://www.kdd.in.tum.de/robust-gcn

Parameter setting The hyperparameter settings are the same as the baseline model [27]. Stochastic gradient descent with mini-batches and Adam Optimizer [9] are used to train GCN models where the batch size is set to 8. The dropout ratio, L_2 regularization and learning rate are set to 0.5, 0.00001, 0.001, respectively. The dimension of the $(l+1)$-th layer is half of the (l)-th layer while we set the dimension of the first hidden layer to 8. Our method is implemented using Pytorch 1.2.0 and all experiments are run in NVIDIA GeForce RTX 2080ti GPU.

Table 1. The statistics of datasets.

Dataset	#Node	#Edge	#Feature	#Class
Cora-ML	2,995	8,416	2,879	7
Citeseer	3,312	4,715	3,703	6
PubMed	19,717	44,324	500	3

4.2 Performance Comparison

Overall performance comparison Table 2 displays the overall performance comparison between our methods and baseline model. The avg-time is the average runtime of certificating all nodes under different perturbation budget q. From Table 2, we observe that our method are significantly faster than the baseline model. For example, ours-1 is 24, 16, and 15 times faster than the baseline model on Cora-ml, Citeseer, and PubMed, respectively. Meanwhile, the avg-max q of ours-1 is half of the baseline model. This can be attributed to the matrix-form propagation rule of our method. The gap of avg-max q can be attributed to that we treat each hidden unit independently.

Table 2. Overall performance comparison.

Dataset	Method	Acc.	Metric	
			Avg-max q	Avg-time (s)
Cora-ML	Baseline	0.84	16.82	24.38
	Ours-1		7.26	<1
	Ours-2		6.34	<1
Citeseer	Baseline	0.70	6.75	16.53
	Ours-1		3.79	1
	Ours-2		3.35	<1
PubMed	Baseline	0.86	9.65	89.73
	Ours-1		5.78	6
	Ours-2		5.10	1

Certification performance w.r.t. q To further investigate the certification performance of different methods, we plot the number of certifiably robust nodes in Figure 4. As can be seen from the figure, when the perturbation budget q is small, the number of certifiably robust nodes of all methods is almost the same. With q increasing, the performance of our methods drops faster than the baseline model.

Certification time w.r.t. q To investigate the certification time w.r.t. the perturbation budget, we plot the runtime in Figure 5. From Fig. 5, we can observed that AI-based methods spend much less time than the baseline model. Moreover, when the number of perturbations increases, the runtime of our methods remains almost the same while the runtime of the baseline model tends to increase.

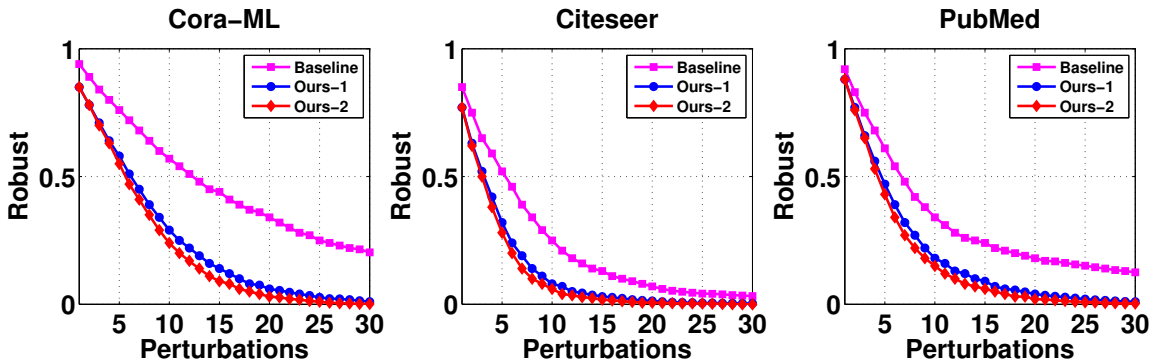

Figure 4. The performance comparison w.r.t the number of perturbations. Our methods generally outperform the baseline model in most cases.

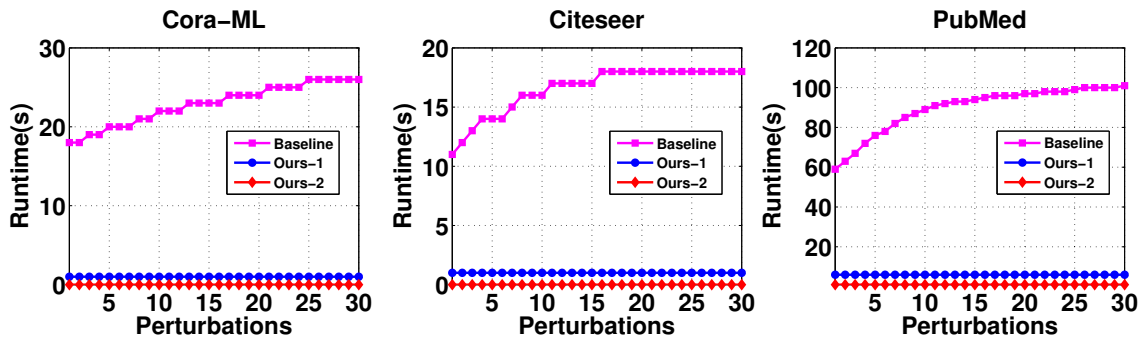

Figure 5. The runtime comparison w.r.t. the number of perturbations. Our methods are faster than the baseline model and the runtime will not increase when the number of perturbations increases.

5 Related Work

In this work, we develop a AI-based robustness certification framework for GCNs. Thus, we categorize previous studies related to our work into two categories: adversarial robustness for GCNs and AI-based robustness certification.

Adversarial robustness for GCNs Adversarial perturbations on image classification [8, 20] have been studied extensively. In recent years, multiple works [26, 25, 5, 1] show that graph convolutional networks are vulnerable to small designed perturbations (i.e. add/drop edges or change node attributes) as well. NETTACK [25] is the first work of adversarial attacks on graph convolutional networks. It firstly performs attacks on a surrogate model, then the generated perturbations are transferred to the real model. Following this work, METATTACK [26] is proposed to decrease the overall performance of graph convolutional networks. RL-S2V [5] considers black-box attacks via reinforcement learning.

To resist such attacks, several heuristic approaches [23, 21, 24] are proposed recently. These methods can be classified to adversarial training [23], malicious edge detection [21] and uncertainty estimation [24]. Although these methods empirically show that they improve GCNs' robustness against certain types of adversarial attacks, they can not verify the robustness w.r.t a certain space of perturbations. Considering this situation, the first robustness certificate is proposed by the work [27] and they further develop robust training method for GCNs based on the certificate.

Different from the works focus on improving the robustness of graph convolutional networks, in this work, the target is to propose a robustness analyzer of the GCN itself. Compare with the existing robustness verifier [27], we employ a different overapproximate strategy which is shown fast and effective.

AI-based robustness certification For formal verification of fully connected or convolutional neural network's robustness, the most related methods are abstract interpretation based verifiers [6, 18, 17, 14, 19]. Different from existing works which focus on the perturbations on continuous features, the perturbations of the discrete domain are ℓ_0 constraints. Therefore, it is hard to define an abstract domain to capture all perturbed inputs. Furthermore, there are no works on transforming the defined abstract domain through graph convolutional layers so far. In this work, we shed light on these problems. Instead of estimating the domain of input features, the possible perturbations are captured by the output of the first GCN layer. The interval domain is employed to approximate the perturbations of each layer and we define how the interval domain transforms through graph convolutional layers.

6 Conclusion and Future Work

In this work, we present a novel and efficient robustness certification method for graph convolutional networks based on abstract interpretation. The core idea is to first define the abstract domain which over-approximates the effect of perturbations and then propagate the abstract domain to obtain the lower and upper bounds of each class. We propose a novel method to specify the abstract domain for binary features and define a set of abstract transformers for graph convolutional networks. Furthermore, a matrix-form propagating rule is developed to efficiently compute the lower and upper bounds of all nodes. Extensive experiments performed on three graph datasets demonstrate the effectiveness of our method.

In future, we plan to extend our work in following directions: (1) Although the interval domain is fast, existing works [17, 14] have demonstrated that the zonotope domain is more precise. Therefore, we will try to improve the precision of AI-based certification using the zonotope domain. (2) How to improve the robustness based on the certification method is another important problem. We aim to develop a robust training algorithm based on this work. (3) Besides adversarial feature attacks, graph convolutional networks also suffer the adversarial structure attacks which change the node prediction by modifying the graph structure. Thus, we will address the problem of how to verify a node's robustness under the structure perturbation based on abstract interpretation in future.

Acknowledgements

The paper was supported by the National Natural Science Foundation of China (61702568, U1711267), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (2017ZT07X355) and the Fundamental Research Funds for the Central Universities under Grant (17lgpy117).

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann, 'Adversarial attacks on node embeddings via graph poisoning', in *ICML*, pp. 695–704, (2019).
[2] Liang Chen, Yang Liu, Zibin Zheng, and Philip S. Yu, 'Heterogeneous neural attentive factorization machine for rating prediction', in *CIKM*, pp. 833–842, (2018).
[3] Zitai Chen, Chuan Chen, Zong Zhang, Zibin Zheng, and Qingsong Zou, 'Variational graph embedding and clustering with laplacian eigenmaps', in *IJCAI*, pp. 2144–2150, (2019).
[4] Patrick Cousot and Radhia Cousot, 'Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints', in *POPL*, pp. 238–252. ACM, (1977).
[5] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song, 'Adversarial attack on graph structured data', in *ICML*, pp. 1123–1132, (2018).
[6] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev, 'AI2: safety and robustness certification of neural networks with abstract interpretation', in *IEEE S&P*, pp. 3–18, (2018).
[7] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, 'Deep sparse rectifier neural networks', in *AISTATS*, pp. 315–323, (2011).
[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, 'Explaining and harnessing adversarial examples', in *ICLR*, (2015).
[9] Diederik P. Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', in *ICLR*, (2015).
[10] Thomas N. Kipf and Max Welling, 'Semi-supervised classification with graph convolutional networks', in *ICLR*, (2017).
[11] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua, 'Attributed social network embedding', *IEEE Trans. Knowl. Data Eng.*, **30**(12), 2257–2270, (2018).
[12] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song, 'Heterogeneous graph neural networks for malicious account detection', in *CIKM*, pp. 2077–2085. ACM, (2018).
[13] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore, 'Automating the construction of internet portals with machine learning', *Inf. Retr.*, **3**(2), 127–163, (2000).
[14] Matthew Mirman, Timon Gehr, and Martin T. Vechev, 'Differentiable abstract interpretation for provably robust neural networks', in *ICML*, pp. 3575–3583, (2018).
[15] Sarah Parisot, Sofia Ira Ktena, Enzo Ferrante, Matthew C. H. Lee, Ricardo Guerrero, Ben Glocker, and Daniel Rueckert, 'Disease prediction using graph convolutional networks: Application to autism spectrum disorder and alzheimer's disease', *Medical Image Analysis*, **48**, 117–130, (2018).
[16] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad, 'Collective classification in network data', *AI Magazine*, **29**(3), 93–106, (2008).
[17] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev, 'Fast and effective robustness certification', in *NeurIPS*, pp. 10825–10836, (2018).
[18] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev, 'An abstract domain for certifying neural networks', *PACMPL*, **3**(POPL), 41:1–41:30, (2019).
[19] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev, 'Boosting robustness certification of neural networks', in *ICLR*, (2019).
[20] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus, 'Intriguing properties of neural networks', in *ICLR*, (2014).
[21] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu, 'Adversarial examples for graph data: Deep insights into attack and defense', in *IJCAI*, pp. 4816–4823, (2019).
[22] Fenfang Xie, Liang Chen, Yongjian Ye, Yang Liu, Zibin Zheng, and Xiaola Lin, 'A weighted meta-graph based approach for mobile application recommendation on heterogeneous information networks', in *ICSOC*, pp. 404–420, (2018).
[23] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin, 'Topology attack and defense for graph neural networks: An optimization perspective', in *IJCAI*, pp. 3961–3967, (2019).
[24] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu, 'Robust graph convolutional networks against adversarial attacks', in *KDD*, pp. 1399–1407, (2019).
[25] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann, 'Adversarial attacks on neural networks for graph data', in *KDD*, pp. 2847–2856, (2018).
[26] Daniel Zügner and Stephan Günnemann, 'Adversarial attacks on graph neural networks via meta learning', in *ICLR*, (2019).
[27] Daniel Zügner and Stephan Günnemann, 'Certifiable robustness and robust training for graph convolutional networks', in *KDD*, pp. 246–256, (2019).