Enhancing Node-Level Adversarial Defenses by Lipschitz Regularization of Graph Neural Networks

Yaning Jia*
School of Cyber Science and Engineering, National Engineering Research Center for Big Data Technology and System, Hubei Engineering Research Center on Big Data Security, Hubei Key Laboratory of Distributed System Security, Services Computing Technology and System Lab, Huazhong University of Science and Technology
Wuhan, Hubei, China
jiayaning@hust.edu.cn

Dongmian Zou*
Zu Chongzhi Center for Mathematics and Computational Sciences, Data Science Research Center, Division of Natural and Applied Sciences, Duke Kunshan University
Kunshan, Jiangsu, China
dongmian.zou@duke.edu

Hongfei Wang†
School of Cyber Science and Engineering, National Engineering Research Center for Big Data Technology and System, Hubei Engineering Research Center on Big Data Security, Hubei Key Laboratory of Distributed System Security, Services Computing Technology and System Lab, Huazhong University of Science and Technology
Wuhan, Hubei, China
hongfei@hust.edu.cn

Hai Jin
School of Computer Science and Technology, Cluster and Grid Computing Lab, National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Huazhong University of Science and Technology
Wuhan, Hubei, China
hjin@hust.edu.cn

ABSTRACT

Graph neural networks (GNNs) have shown considerable promise for graph-structured data. However, they are also known to be unstable and vulnerable to perturbations and attacks. Recently, the Lipschitz constant has been adopted as a control on the stability of Euclidean neural networks, but calculating the exact constant is also known to be difficult even for very shallow networks. In this paper, we extend the Lipschitz analysis to graphs by providing a systematic scheme for estimating upper bounds of the Lipschitz constants of GNNs. We also derive concrete bounds for widely used GNN architectures including GCN, GraphSAGE and GAT. We then use these Lipschitz bounds for regularized GNN training for improved stability. Our numerical results on Lipschitz regularization of GNNs not only illustrate enhanced test accuracy under random noise, but also show consistent improvement for state-of-the-art defense methods against adversarial attacks.

CCS CONCEPTS

• **Computing methodologies** → Neural networks; **Neural networks**; **Regularization**; • **Mathematics of computing** → **Graph algorithms**.

KEYWORDS

graph neural networks, Lipschitz constants, stability, adversarial defense

ACM Reference Format:
Yaning Jia, Dongmian Zou, Hongfei Wang, and Hai Jin. 2023. Enhancing Node-Level Adversarial Defenses by Lipschitz Regularization of Graph Neural Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599335

1 INTRODUCTION

Recently, graph neural networks (GNNs) have produced impressive results in various machine learning tasks, such as social recommendation [53], image understanding [23], and drug design [29]. For graph structured data, GNNs are capable of transmitting and accumulating features as "messages" according to the topology of its underlying graph. However, similar to their Euclidean counterpart, whose performance drops dramatically when affected by adversarial attacks [21, 54], GNNs also suffer from such vulnerability. In particular, GNNs are unstable and small perturbations in the input feature or the underlying graph structure may degrade their generalization performance [30, 41, 68, 69]. Especially, when GNNs are applied to safety-critical fields, their vulnerability strikes serious security concerns. For example, for credit prediction systems, an

*Both authors contributed equally to this research.
†Corresponding author.

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 the author(s) 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 '23, August 6–10, 2023, Long Beach, CA, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599335

attacker could create fake friendship, i.e., graph edges that should not have existed, to get a high credit score [9].

Recently, an increasing number of studies have been devoted to the study of GNN vulnerability and proposed various methods that greatly fortify GNNs against imperceptible perturbations [11, 31, 52, 69]. However, these methods mainly focus on purifying the graph data and do not seek to improve the stability of GNNs themselves. They are based on priors about perturbed data and their performance depends on whether the prior beliefs match the specific attack types. Unlike these approaches, in this paper, we propose to enhance GNNs by improving their stability as a built-in property, measured by the Lipschitz constants of GNNs.

The Lipschitz constant of a neural network represents the worst-case change of the output in correspondence to a perturbation of the input. Recent years have witnessed many applications in this regard [12, 26, 43, 62]. Despite their importance, the complexity of neural network architectures makes it hard to accurately compute their Lipschitz constants. Indeed, finding the Lipschitz constant of neural networks is known to be NP-hard even for a simple two-layer fully connected network [49]. Therefore, recent works mainly focus on finding upper bounds of Lipschitz constants, which are called Lipschitz bounds for simplicity [13, 44]. Similar to the Euclidean case, Lipschitz constants are also important stability guarantees to GNNs [16]. Despite the rich literature on Lipschitz estimation in the Euclidean domain, there is little work on the Lipschitz bounds of GNNs. The study of GNNs Lipschitz constants differs from that of Euclidean neural networks in two ways: first, for node-level tasks, a perturbation exerts different effects on different graph nodes and thus we need to study the Lipschitz constants in a node-wise manner; second, the input features on the graph nodes are not independent and they have to be considered at the same time according to the graph topology. As a consequence, it is more difficult to analyze stability of GNNs. The way a GNN depends on the graph topology is specified by its convolutional layers, also known as message passing layers. In this paper, our analysis is based on the study of these layers. Following such analysis, we also provide concrete and implementable bounds for well-known GNNs. With such bounds, we obtain regularized training of GNNs and use them in defense against perturbations and attacks.

The contribution of this paper is summarized as follows:

- We derive the Lipschitz bounds of GNN layers based on the $\ell_{\infty,2}$-norm of a "Lipschitz matrix". We then quantify the influence of the nonlinear activation functions on the local Lipschitz bounds by considering the interaction between the activation and the output of the convolutional layer. Accordingly, Lipschitz bounds are derived for generic GNNs.
- We apply the above analysis to derivation of compact forms of Lipschitz bounds of widely used GNNs including the Graph Convolutional Network (GCN) [35], GraphSAGE [24] and the Graph Attention Network (GAT) [47].
- We perform regularized training of GNNs using the derived Lipschitz bounds. Our experiments show that the regularization substantially improves the stability of GNN under perturbations by additive Gaussian noise in the input space. Moreover, when the graph data are poisoned using adversarial attacks, our approach is not only an effective standalone defense method, but also advances state-of-the-art adversarial defense methods when used on top of them.

2 RELATED WORKS

Lipschitz constants of neural networks. The early work [44] attributed the instability of a neural network to a large Lipschitz constant and proposed to use the product of the spectral norms of all the layers as an upper bound for the Lipschitz constant. Later works [13, 25, 32, 36, 49] formulated various optimization frameworks to obtain tighter Lipschitz bounds. Works that studied specific types of neural networks that contain convolutional layers or attention layers include [1, 34, 46, 66]. As to GNNs, Dasoulas et al. [10] proposed a Lipschitz normalization method for self-attention layers of GAT. Their proposed Lipschitz normalization enables training deep GAT by alleviating gradient explosion. Our work is different from theirs as we have a different analysis on more general GNNs, and perform a different regularization instead of normalization. More recently, Gama and Sojoudi [18] estimated the filter Lipschitz constant using the infinite norm of a matrix, where each element is the Lipschitz constant of the corresponding position filter. The Lipschitz matrix in our work follows a different definition and has different choice of norm types. Moreover, our task of making GNNs more robust against attacks is not considered in the above works.

Stability guarantee of GNN. Many recent works studied the stability properties of GNN. For instance, Verma and Zhang [48] showed that the stability of a single-layer GNN is determined by the largest eigenvalue of the graph Laplacian. Also, by deriving Lipschitz properties, various works [17, 19, 67] studied the stability of graph scattering transform, which is considered as a special GNN with fixed weights. Gama et al. [16] studied how various Lipschitz conditions of GNN filters affect the stability of the network. Keriven et al. [33] inspected the limit behavior regarding stability when the number of nodes approaches infinity. Our work is different from the above works in two ways: first, we have a different setting that is directly related to node-level defense; second, our bounds for specific GNNs are easily implementable for regularizing GNNs.

Denfense methods for graph adversarial attacks. With the advancement of studies on GNNs these years, there are many works on adversarial attacks and defenses of graph data such as [11, 31, 51, 52, 55, 60, 69, 70]. One work related to ours is [62], which designed attack adaptive 1-Lipschitz GNNs by constraining the weight norms of the network. However, their approach to computing the Lipschitz constants is based on a simple setting of linear layers, which neither takes into account the node-level scenario nor accommodates various GNN architectures. Other ways of constructing GNNs that are robust against attacks include [3, 4, 28, 45, 65]. For example, Zhu et al. [65] represented node features as Gaussian distributions, so that adversarial changes can be absorbed in the variance. Another defense approach is finding perturbed data by exploiting the essential difference between perturbed and clean data [27, 52, 56].

Regularization methods for neural networks. Zhang et al. [59] and Loshchilov and Hutter [38] both discussed ℓ_2 regularization of general neural networks for improved generalizability. Chan et al. [6] applied Jacobian regularization for adversarial robustness. The earliest works that used Lipschitz regularization in neural networks found natural applications in generative adversarial networks [2, 39], where constraining the Lipschitz constant is required in the Wasserstein formulation. Lipschitz regularization for general neural networks was studied in [22, 40, 58]. For GNNs, various regularization methods were proposed [7, 61, 63, 64]. However, their methods are targeted to solving the oversmoothing problem of GNNs and have no application to robustness.

3 PRELIMINARIES

We first review some definitions regarding Lipschitz functions. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be Lipschitz continuous on an input set $\mathcal{X} \subseteq \mathbb{R}^n$ if there exists a constant $K \geq 0$ such that for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, f satisfies the following inequality:

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq K \|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}. \tag{1}$$

The smallest possible K for which (1) holds is called the Lipschitz constant of f and we say that f is a K-Lipschitz function. For clarity, we use $\mathrm{Lip}(f)$ to denote the Lipschitz constant of f. That is,

$$\mathrm{Lip}(f) = \sup_{\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}, \boldsymbol{x} \neq \boldsymbol{y}} \frac{\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|}{\|\boldsymbol{x} - \boldsymbol{y}\|}. \tag{2}$$

It is clear from the definition that the Lipschitz constant of a function is the largest possible change of the output corresponding to a perturbation of the input of unit norm. Therefore, the Lipschitz constant of a neural network measures its stability with respect to the input features. Because of the difficulty in finding exact constant, we focus on obtaining an upper bound. Such upper bound is called a Lipschitz bound.

For machine learning tasks on graphs, since the input features are usually known at the available nodes, it is naturally more interesting to consider local perturbations and responses. In this regard, we define the local Lipschitz constant at $\boldsymbol{x}$ to be

$$\mathrm{Lip}_{\boldsymbol{x}, \mathcal{U}_0(\boldsymbol{x})}(f) = \sup_{\boldsymbol{y} \in \mathcal{U}_0(\boldsymbol{x})} \frac{\|f(\boldsymbol{x}) - f(\boldsymbol{y})\|}{\|\boldsymbol{x} - \boldsymbol{y}\|}, \tag{3}$$

where $\mathcal{U}_0(\boldsymbol{x}) \subset \mathcal{X}$ is a punctured neighborhood of $\boldsymbol{x}$. We also denote

$$\mathrm{Lip}_{\boldsymbol{x}}(f) = \mathrm{Lip}_{\boldsymbol{x}, \mathcal{X} - \{\boldsymbol{x}\}}(f), \tag{4}$$

when the neighborhood is the whole space $\mathcal{X}$.

Next, we introduce some terminology for discussing GNN. We assume a given graph $G = G(V, E)$, where V denotes the set of nodes and E denotes the set of edges. We will use $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\} \subset \mathbb{R}^F$ to denote the N node features in $\mathbb{R}^F$, as the input of any layer of a GNN. By abuse of notation, when there is no confusion, we also follow GNN literature and consider $\boldsymbol{X}$ as the $\mathbb{R}^{N \times F}$ matrix whose i-th row is given by $\boldsymbol{x}_i^\top$, $i = 1, \cdots, N$, though it unnecessarily imposes an ordering of the graph nodes.

We consider a GNN as a function $f = f(\boldsymbol{A})$, where $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of G. Formally, we consider a GNN according to the following definition.

Definition 3.1. An L-layer GNN is defined to be a function $f : \mathbb{R}^{N \times F^{\text{in}}} \to \mathbb{R}^{N \times F^{\text{out}}}$ depending on the graph adjacency matrix $\boldsymbol{A}$, such that

$$f = h_L \circ \rho_{L-1} \circ \cdots \rho_1 \circ h_1, \tag{5}$$

where $h_l : \mathbb{R}^{F^{l-1}} \to \mathbb{R}^{F^l}$ is the l-th convolutional (message passing) layer of the GNN and $\rho_l : \mathbb{R}^{F^l} \to \mathbb{R}^{F^l}$ is the activation function in the l-th layer, $l = 1, \cdots, L$. Also, $F^{\text{in}} = F^0$ and $F^{\text{out}} = F^L$.

Notations. We use $\{\}$ to denote sets and $()$ to denote vectors. For $n \in \mathbb{N}$, we denote $[n] = \{1, \cdots, n\}$. We use regular letter to denote scalars, lower-case bold letters to denote vectors, and upper-case bold letters to denote matrices. For instance, $\boldsymbol{x} = (x_1, \cdots, x_n)^\top \in \mathbb{R}^n$ and $\boldsymbol{X} = [X_{ik}]_{i \in [n], k \in [m]} \in \mathbb{R}^{n \times m}$. For any vector $\boldsymbol{x} \in \mathbb{R}^n$, we use $\|\boldsymbol{x}\|$ to denote its vector ℓ_2-norm: $\|\boldsymbol{x}\| = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$. For any matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, we use $\boldsymbol{X}_{i,:}$ to denote its i-th row and $\boldsymbol{X}_{:,k}$ to denote its k-th column, and we denote the $(\infty, 2)$-norm of $\boldsymbol{X}$ by $\|\boldsymbol{X}\|_{\infty,2} = \max_{i \in [n]} \|\boldsymbol{X}_{i,:}\|$. We use $\otimes$ to denote tensor product and $\odot$ to denote the Hadamard (elementwise) product of two matrices or vectors. Let $G = G(V, E)$ be a graph and suppose an order of the nodes are given. We denote its adjacency matrix by $\boldsymbol{A}$ such that $A_{ij} = 1$ if $\{i, j\} \in E$ and $A_{ij} = 0$ if $\{i, j\} \notin E$. When it is clear from the context, we use $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ to denote a feature matrix whose i-th row corresponds to the features on the i-th node. When we consider the i-th node of a graph, we use $\tilde{\boldsymbol{X}}$ to denote another input feature so that $\tilde{\boldsymbol{X}}_{j,:} = \boldsymbol{X}_{j,:}$ for $j \neq i$.

4 LIPSCHITZ BOUNDS OF GNNS

We derive Lipschitz bounds of GNN layers and particularly, closed-form formulas for Lipschitz bounds of widely used GNN convolutional layers in §4.1. We provide local Lipschitz bounds of general GNNs with 1-Lipschitz activation functions in §4.2. The proofs of the theoretical results in this section can be found in Appendix A.

4.1 Lipschitz bounds of convolutional layers

Since local graph structures impose non-homogeneity to the nodes, we need to consider all the nodes at the same time when deriving a Lipschitz bound. That is, we need to consider $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ instead of $\boldsymbol{x} \in \mathbb{R}^F$. Therefore, in addition to (1) and (3), we need to derive a Lipschitz bound for vector-valued functions.

Lemma 4.1. *Let $g : \mathbb{R}^n \to \mathbb{R}^m$ be a Lipschitz continuous function. Denote $g_i : \mathbb{R}^n \to \mathbb{R}$ to be the i-th component of g, $i = 1, \cdots, m$. Then the Lipschitz constant of g satisfies*

$$\mathrm{Lip}(g) \leq \left\| [\mathrm{Lip}(g_i)]_{i=1}^m \right\|, \tag{6}$$

where $[\mathrm{Lip}(g_i)]_{i=1}^m$ denotes the m-dimensional vector whose i-th component is $\mathrm{Lip}(g_i)$.

Consider a single convolutional layer $h : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times F'}$ of a GNN. Assume that the Lipschitz constant of h_j for the i-th node is

$$\mathrm{Lip}(h_{ij}) := \sup_{\boldsymbol{x}_i \neq \tilde{\boldsymbol{x}}_i} \frac{\|h(\boldsymbol{X})_j - h(\tilde{\boldsymbol{X}})_j\|}{\|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|} < \infty.$$

We remark that in general, even if we only care about the i-th node, $\mathrm{Lip}(h_{ij})$ also depends on $\boldsymbol{x}_{i'}$ for $i' \neq i$. This makes the analysis of GNN different than Euclidean neural networks where the input samples are assumed to be independently drawn. In order to analyze the Lipschitz bounds of h, we define the following Lipschitz matrix:

$$\boldsymbol{M}_{\text{Lip}}(h) = \begin{bmatrix} \text{Lip}(h_{11}) & \text{Lip}(h_{12}) & \cdots & \text{Lip}(h_{1,F'}) \\ \text{Lip}(h_{21}) & \text{Lip}(h_{22}) & \cdots & \text{Lip}(h_{2,F'}) \\ \vdots & \vdots & \vdots & \vdots \\ \text{Lip}(h_{N1}) & \text{Lip}(h_{N2}) & \cdots & \text{Lip}(h_{N,F'}) \end{bmatrix}. \tag{7}$$

Combining (7) with (6), we can calculate the Lipschitz bound for each node, and estimate the local Lipschitz constant of h by $\text{Lip}(h_i) \leq \left\| \left[\text{Lip}(h_{ij}) \right]_{j=1}^{F'} \right\|$. By virtue of this estimation, we define a Lipschitz bound of h, denoted as $\text{LB}(h)$, to be the maximum value of the Lipschitz bounds of all the nodes. That is,

$$\text{LB}(h) := \max_{i \in [N]} \left\| \left[\text{Lip}(h_{ij}) \right]_{j=1}^{F'} \right\| = \left\| \boldsymbol{M}_{\text{Lip}}(h) \right\|_{\infty,2}. \tag{8}$$

The local Lipschitz bounds $\text{LB}_{\boldsymbol{X},\mathcal{U}_0(\boldsymbol{X})}(h)$ can be defined similarly, by replacing Lip above with $\text{Lip}_{\boldsymbol{x}_i,\mathcal{U}_0(\boldsymbol{x}_i)}$. That is,

$$\text{LB}_{\boldsymbol{X},\mathcal{U}_0(\boldsymbol{X})}(h) := \max_{i \in [N]} \left\| \left[\text{Lip}_{\boldsymbol{x}_i,\mathcal{U}_0(\boldsymbol{x}_i)}(h_{ij}) \right]_{j=1}^{F'} \right\|. \tag{9}$$

To apply the Lipschitz bounds to controlling the stability of GNNs, it is crucial to derive analytical formulas with terms that can be extracted from GNN parameters. In what follows, we derive Lipschitz bounds for widely used GNN layers including GCN [35], GraphSAGE [24] and GAT [47]. For simplicity, we use GCNConv, SAGEConv and GATConv to express a convolution layer of these GNNs, respectively.

GCNConv. A GCNConv layer processes an input feature $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ according to

$$\boldsymbol{Z} = \text{GCNConv}(\boldsymbol{X}) := \hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}, \tag{10}$$

in which $\boldsymbol{W} \in \mathbb{R}^{F \times F'}$ is a trainable weight matrix and $\hat{\boldsymbol{A}} = (\boldsymbol{D}+\boldsymbol{I})^{-1/2}(\boldsymbol{A}+\boldsymbol{I})(\boldsymbol{D}+\boldsymbol{I})^{-1/2}$ where $\boldsymbol{D}$ is the diagonal degree matrix such that $\boldsymbol{D}_{ii} = \sum_{j=1}^{N} \boldsymbol{A}_{ij}$. The Lipschitz bound of GCNConv depends on $\hat{\boldsymbol{A}}$ and $\boldsymbol{W}$, which is summarized in the following theorem.

Theorem 4.2. *The Lipschitz bound of GCNConv satisfies*

$$\text{LB}(\text{GCNConv}) \leq \max_{i \in [N]} \left\| \left[\left| \hat{A}_{ii} \right| \left\| \boldsymbol{W}_{:,k} \right\| \right]_{k=1}^{F'} \right\|, \tag{11}$$

where $\hat{A}_{ii}$ is the (i,i)-th entry of $\hat{\boldsymbol{A}}$ and $\boldsymbol{W}_{:,k}$ is the k-th column of $\boldsymbol{W}$.

SAGEConv. The original GraphSAGE considers a convolutional layer that includes two parts: sampling and aggregation. Since sampling (and similarly dropout) only affects the Lipschitz constant proportionally to the sampling probability, in the current work, it suffices to consider aggregation alone. With this understanding, a SAGEConv layer processes an input feature $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ by

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}_1 + \boldsymbol{V}\sigma(\boldsymbol{X}\boldsymbol{W}_2 + \boldsymbol{1} \otimes \boldsymbol{b}^\top)\boldsymbol{W}_3, \tag{12}$$

where $\boldsymbol{V} = \boldsymbol{D}^{-1}\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the mean operator which averages neighboring features for all nodes, $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix whose i-th diagonal entry satisfies $D_{ii} = \sum_j A_{ij}$. $\boldsymbol{W}_1 \in \mathbb{R}^{F \times F'}, \boldsymbol{W}_2 \in \mathbb{R}^{F \times F''}, \boldsymbol{W}_3 \in \mathbb{R}^{F'' \times F'}$ are weight matrices and $\boldsymbol{1} \otimes \boldsymbol{b}^\top$ is the bias matrix whose rows are identically given by the transpose of $\boldsymbol{b} \in \mathbb{R}^{F''}$.

Theorem 4.3. *The Lipschitz bound of SAGEConv satisfies*

$$\begin{aligned} \text{LB}(\text{SAGEConv}) \leq \max_{i \in [N]} \Big\| \Big[& \left\| (\boldsymbol{W}_1)_{:,k} \right\| + \\ & V_{ii} \left\| (\boldsymbol{W}_2\boldsymbol{W}_3)_{:,k} \right\| \Big]_{k=1}^{F'} \Big\| \end{aligned} \tag{13}$$

where V_{ii} is the (i,i)-th entry of $\boldsymbol{V}$, $(\boldsymbol{W}_1)_{:,k}$ is the k-th column of $\boldsymbol{W}_1$ and $(\boldsymbol{W}_2\boldsymbol{W}_3)_{:,k}$ is the k-th column of the matrix $\boldsymbol{W}_2\boldsymbol{W}_3$.

GATConv. GAT uses an attention mechanism to determine importance of different nodes. A single-head GAT processes an input feature $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ by

$$\boldsymbol{Z} = \boldsymbol{S}\boldsymbol{X}\boldsymbol{W}, \tag{14}$$

where $\boldsymbol{W} \in \mathbb{R}^{F \times F'}$ is the trainable weight matrix, and $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ is the attention coefficient matrix whose (i,j)-th entry is given by

$$S_{ij} = \frac{\exp(\sigma(\boldsymbol{a}[\boldsymbol{W}\boldsymbol{x}_i \parallel \boldsymbol{W}\boldsymbol{x}_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\sigma(\boldsymbol{a}[\boldsymbol{W}\boldsymbol{x}_i \parallel \boldsymbol{W}\boldsymbol{x}_k]))}. \tag{15}$$

Here $\parallel$ is the concatenation operator, $\boldsymbol{a} = [\boldsymbol{a}_1, \boldsymbol{a}_2]$ is a column vector where $\boldsymbol{a}_l \in \mathbb{R}^{F' \times 1}, l = 1, 2$, and σ is an activation function (e.g. LeakyReLU).

Theorem 4.4. *Let $\boldsymbol{M}_{\text{LB}}(\text{GATConv})$ be the matrix whose (i,k)-th entry is given by*

$$\begin{aligned} \boldsymbol{M}_{\text{LB}}(\text{GATConv})_{ik} = (S_{ii}\boldsymbol{X}_{i,:}\boldsymbol{W}_{:,k} - S_{ii} & \sum_{j \in \mathcal{N}_i} \\ S_{ij}\boldsymbol{X}_{j,:}\boldsymbol{W}_{:,k}) \left\| \boldsymbol{v} \right\| & + S_{ii} \left\| \boldsymbol{W}_{:,k} \right\|, \end{aligned} \tag{16}$$

where $\boldsymbol{X}_{i,:} \in \mathbb{R}^{1 \times F}$ is the i-th row of $\boldsymbol{X}$, $\boldsymbol{W}_{:,k} \in \mathbb{R}^{F \times 1}$ is the k-th column of $\boldsymbol{W}$, $\boldsymbol{v} = \boldsymbol{a}_2\boldsymbol{W}^T$, and $\mathcal{N}_i$ represents the set of neighbors of the i-th node. Then for any $\epsilon > 0$, there exists a punctured neighborhood $\mathcal{U}_0(\boldsymbol{X})$ of $\boldsymbol{X}$, such that

$$\text{LB}_{\boldsymbol{X},\mathcal{U}_0(\boldsymbol{X})}(\text{GATConv}) \leq \left\| \boldsymbol{M}_{\text{LB}}(\text{GATConv}) \right\|_{\infty,2} + \epsilon. \tag{17}$$

For multi-head GAT, we simply estimate the Lipschitz bound using the number of heads multiplied by the largest Lipschitz bound of any single head. Unlike GCNConv and SAGEConv, the Lipschitz bound of GATConv is not only related to the graph topology, but also related to the input features of the layer. Also, unlike the global Lipschitz bounds of GCNConv and SAGEConv, the Lipschitz bounds of GATConv are local. We do not expect to remove the locality because of the exponential functions used in (15). Also, according to the discussion by Dasoulas et al. [10], a GATConv layer is not necessarily globally Lipschitz.

4.2 Local Lipschitz bounds of GNNs

In this section, we consider a GNN with 1-Lipschitz activation functions such as ReLU, LeakyReLU and tanh. Propagating features through these activation functions does not increase the Lipschitz constant of the underlying GNN. Therefore, when estimating the Lipschitz constant of GNN, one can use a simple upper bound which is the product of the Lipschitz bounds from all convolutional layers. That is,

$$\begin{aligned} \text{LB}(f) \leq \text{LB}_{\text{Node}}(f) &:= \max_{i \in [N]} \text{LB}_{\text{Node}}(f_i), \\ \text{where} \quad \text{LB}_{\text{Node}}(f_i) &:= \prod_{l=1}^{L} \text{LB}(h_i^l). \end{aligned} \tag{18}$$

Nevertheless, local Lipschitz constants can be more heavily affected by the values of the activation functions. Let $h : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times F'}$ be a GNN layer. Given an input $X \in \mathbb{R}^{N\times F}$, let $Z = h(X)$. We define the Lipschitz matrix of $\rho \circ h$ at X as

$$M_{\mathrm{Lip}}(\rho \circ h) = Q \odot M_{\mathrm{Lip}}(h), \tag{19}$$

where $Q_{ij} = \rho'(Z_{ij})$ and we omit the dependence on X for clarity. In other words, we take entrywise product of the Lipschitz matrix of h and a mask matrix $Q \in \mathbb{R}^{N\times F'}$ constructed by derivatives of the activation. In particular, if $\rho = \mathrm{ReLU}$, then

$$Q_{ij} = \begin{cases} 1, & \text{if} \quad Z_{ij} \geq 0; \\ 0, & \text{otherwise}. \end{cases} \tag{20}$$

Let

$$\mathrm{LB}(\rho \circ h) = \left\| M_{\mathrm{Lip}}(\rho \circ h) \right\|_{\infty,2}. \tag{21}$$

We provide an estimate for the Lipschitz bound of GNNs using the Lipschitz bounds for their convolutional layers. Given a GNN f defined in (5), we have the following Lipschitz bound of f

$$\begin{aligned} &\mathrm{LB}_\rho(f) = \max_{i\in[N]} \mathrm{LB}_\rho(f_i), \\ &\text{where} \quad \mathrm{LB}_\rho(f_i) := \mathrm{LB}(h_i^L) \prod_{l=1}^{L-1} \mathrm{LB}(\rho \circ h_i^l). \end{aligned} \tag{22}$$

5 EXPERIMENTS

We perform regularized training of GNNs as an application of the Lipschitz bounds derived in the previous section. We verify improved robustness against noisy node features in §5.1, and adversarial perturbations on graph topology structure in §5.2.

Setup. We use PyTorch-Geometric [14] for implementing various GNNs and Deep-Robust [37] for implementing various methods for adversarial attacks and defenses. In both §5.1 and §5.2, we use the same split of training, validation and test set to obtain comparable results, whereas the split details can be found in Table 1.

The Lipschitz regularized training uses the Lipschitz bound as an additive penalty term. The weight λ of the penalty term is regarded as a hyperparameter, which is chosen from {0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001} by validation. In our networks, we consider ReLU activation function and denote our regularization method as "LipReLU". On the other hand, we use Normal to denote the non-regularized training and L2-Reg to denote training with ℓ_2 regularization (the hyperparameter is chosen from the same validation set). We perform three implementations for each experiment setting and record the mean and standard deviation.

5.1 Defense against noisy data

Datasets. We consider the commonly used datasets in node classification tasks from GNN literature as follows: citation networks including Cora and Citeseer [57], where graph nodes represent publications and edges represent citations; as well as the Wisconsin dataset [8], where nodes represent webpages and edges represent hyperlinks between them. The statistics of all datasets as well as the number of nodes in the train, validation and test sets are available in Table 1. We add zero-mean Gaussian noise to the input features of validation and test sets. For each dataset, we choose a distinct noise level (nl) under which there is observable deterioration of performance using a regular GNN. Specifically, nl = 0.05 in Cora, nl = 0.01 in Citeseer, and nl = 0.10 in Wisconsin.

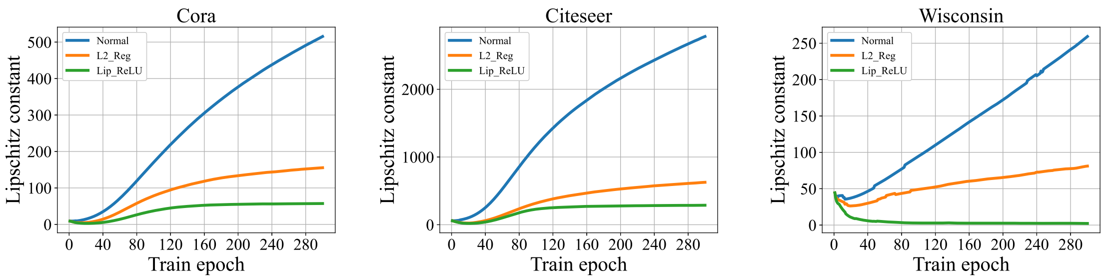
Settings. To illustrate how Lispchitz regularization enhances robustness of GNNs, we conduct experiments on GCN, GraphSAGE and GAT. We consider different GCN structures where the number of convolutional layers vary from 2 to 5, but only consider a 2-layer GraphSAGE and a 2-layer GAT. We remark that deeper architectures do not enhance the performance of GNNs because of the well-known "oversmoothing" phenomenon [5, 7]. The number of hidden units is 16 for GCN and GraphSAGE, and 8 for GAT in each of the 8 heads. We train 300 epochs over a full batch of nodes for all models and methods, using the Adam optimizer with an initial learning rate of 0.01 for GCN and GraphSAGE and 0.05 for GAT.

Results. We first report how Lipschitz bounds change during training. To this end, we fix the architecture of a two-layer GCN (other architectures show similar patterns) and plot the change of the Lipschitz bounds with epochs in Figure 1a. For comparison, in Figure 1b, we plot the change of training loss with epochs. For all the datasets, we observe that consistently, non-regularized training achieves the largest Lipschitz bounds while Lipschitz regularization achieves the smallest bounds. In Cora and Citeseer, with Lipschitz regularization, the Lipschitz bound still increases since it is necessary for the GCN to be sufficiently discriminative. Nevertheless, in Wisconsin, the Lipschitz bound decreases with training epochs, which leads to a much smaller Lipschitz bound than the initialized GCN. Another observation is that although the training losses oscillate with epochs, the Lipschitz constants show a much smoother change during training. This is not surprising since unlike the training losses, expressions of the Lipschitz bounds do not explicitly depend on the training data.

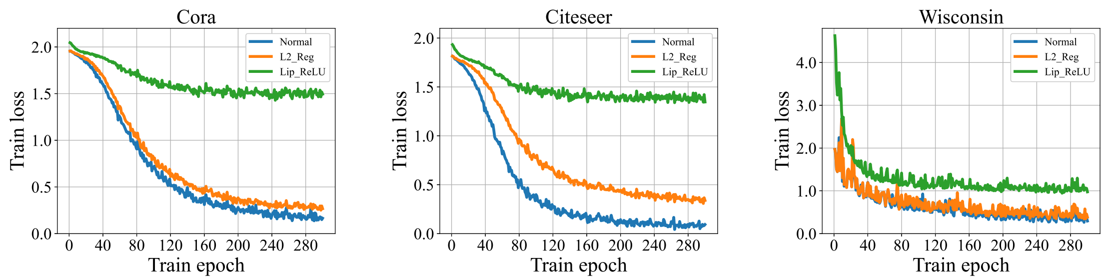
Next, we consider GCNs with $L = 2, 3, 4, 5$ layers and report the test accuracy for all the datasets in Table 2a. From the results, it is clear that Lipschitz regularization significantly improve the performance of GCN when the test dataset is noisy. In all the cases, Lipschitz regularization obtains the best performance. It is also worth noting that, when the number of layers increases, the performance of ℓ_2 regularization degrades considerably and can produce worse accuracy than non-regularized GCN. The reason is that a small ℓ_2 norm may lead to a small Dirichlet energy and thus causes oversmoothing [5, Lemma 3.2].

Table 1: Statistics of datasets used in the experiments

Datasets	#Nodes	#Edges	#Features	#Classes	#Training	#Validation	#Test
Cora	2,708	5,429	1,433	7	140	500	1,000
Citeseer	3,327	4,732	3,703	6	120	500	1,000
Wisconsin	251	466	1,703	5	60	70	120



(a) Change of Lipschitz bounds with epochs.



(b) Change of training losses with epochs.

Figure 1: The change of Lipschitz bounds and training losses for the GCN with $L = 2$ layers

In Table 2b, we report the test accuracy of GraphSAGE and GAT with $L = 2$ layers. We observe that similar to GCN, Lipschitz regularized GraphSAGE and GAT achieve significantly better accuracy than non-regularized or ℓ_2 regularized GraphSAGE and GAT.

Table 2: The test accuracy for noisy graph data

(a) The test accuracy of GCN with 2 to 5 layers

Method	L	Cora	Citeseer	Wisconsin
Normal	2	38.24±0.34	50.20±0.24	48.76±0.45
L2-Reg		54.15±0.76	64.30±0.39	53.72±0.52
LipReLU		**55.76**±0.12	**66.30**±0.27	**60.05**±0.39
Normal	3	35.74±0.41	47.90±0.14	55.37±0.33
L2-Reg		49.75±0.35	57.90±0.30	54.55±0.19
LipReLU		**54.45**±0.82	**62.00**±0.20	**58.99**±0.43
Normal	4	41.54±0.22	38.90±0.65	51.24±0.74
L2-Reg		40.14±0.55	48.30±0.44	52.30±0.46
LipReLU		**53.65**±0.55	**53.60**±0.47	**58.40**±0.42
Normal	5	36.04±0.39	42.20±0.56	50.26±0.60
L2-Reg		36.14±0.33	39.90±0.78	49.91±0.42
LipReLU		**49.45**±0.77	**48.80**±0.23	**53.17**±0.77

(b) The test accuracy of GraphSAGE and GAT with 2 layers

Model	Method	Cora	Citeseer	Wisconsin
GraphSAGE	Normal	38.44±0.20	45.20±0.36	42.15 ±0.25
	L2-Reg	49.15± 0.34	**54.25**±0.45	48.76±0.64
	LipReLU	**50.55**±0.67	**54.25**±0.42	**52.89**±0.57
GAT	Normal	30.83±0.56	31.70±0.21	48.76±0.24
	L2-Reg	36.64±0.32	37.30±0.67	52.07±0.52
	LipReLU	**43.94**±0.22	**41.10**±0.55	**52.89**±0.67

For clear comparison, we also present the performance of the above models on unperturbed data in Table 3. We observe that

Table 3: The test accuracy for clean graph data

(a) The test accuracy of GCN with 2 to 5 layers

Method	L	Cora	Citeseer	Wisconsin
Normal	2	78.48± 0.66	63.04± 1.76	63.36± 0.39
L2-Reg		81.64± 0.42	69.24± 0.41	64.73± 0.42
LipReLU		**81.68**± 0.56	**70.40**± 0.35	**68.60** ±0.33
Normal	3	74.97± 1.97	59.48± 1.13	61.15± 0.67
L2-Reg		78.36± 0.66	60.50± 0.98	63.36± 0.39
LipReLU		**79.58**± 0.34	**66.80**± 0.31	**63.64** ±0.77
Normal	4	71.63± 0.85	58.36± 0.85	55.64± 0.19
L2-Reg		69.99± 0.52	55.14± 0.53	54.27± 0.77
LipReLU		**78.28**± 0.21	**58.80**± 0.09	**59.50** ±0.44
Normal	5	66.77± 0.76	51.70± 0.09	54.27± 0.77
L2-Reg		60.94± 0.39	49.08± 0.14	54.29± 1.06
LipReLU		**74.17**± 0.11	**58.60**± 0.12	**56.20** ±0.89

(b) The test accuracy of GraphSAGE and GAT with 2 layers

Model	Method	Cora	Citeseer	Wisconsin
GraphSAGE	Normal	82.80± 0.74	69.58± 1.56	46.28±0.45
	L2-Reg	83.90± 0.37	**70.60**± 0.73	48.76±0.24
	LipReLU	**84.58**± 0.23	70.30± 0.86	**53.72**±0.94
GAT	Normal	81.28± 0.69	68.26± 0.81	63.69±0.74
	L2-Reg	82.08± 0.33	70.31± 0.32	65.00±0.38
	LipReLU	**82.18**± 0.45	**71.60**±0.56	**68.60**±0.55

interestingly, even for clean data, for all models but one, Lipschitz regularization obtains the best test accuracy, albeit not with a significant advantage. This agrees with the well-known fact that Lipschitz regularized neural networks have better generalization capability [15, 42]. Moreover, for Citeseer and Wisconsin, the test accuracy of GCN for noisy data are close to clean data. This implies that the Lipschitz regularized GCNs for those two datasets are very robust.

5.2 Defense against adversarial attacks

Datasets. We use the same datasets as in Table 1 and consider poisoning the dataset with untargeted attacks which are designed to hinder the overall performance of the models (rather than a targeted subset of the test data). The perturbations are applied to the graph topology structure, e.g., by adding or deleting edges, or changing the edge weights. We consider the following three untargeted attacks: Topology attack [55], DICE [51], and Metaattack [70]. Here, Topology attack and DICE are both white-box attacks while Metaattack is a gray-box attack. We consider these methods used as poisoning attacks which does not depend on the follow-up GNN training.

We use the perturbation rate (ptb%) to represent the percentage of edges that can be perturbed in the graph, which varies from 0% to 25% (0% represents clean data) in our experiments.

Settings. We consider a 2-layer GCN and consider both a non-regularized model and a Lipschitz regularized model. In addition, we consider the following adversarial defense methods: GCN-Jaccard [31] and GCN-SVD [11]. We remark that the citation datasets and the Wisconsin dataset perform differently under the baseline defense methods. These methods build up adversarial defense by attempting to reconstruct pure graph data and are thus independent from regularization of GCN. Therefore, we also consider applying Lipschitz regularization on top of these methods. In addition, to study the defense effect of Lipschitz regularization on a different model, we also report the performances from a 2-layer GAT, with both a non-regularized model and a Lipschitz regularized model. Notice that GCN-Jaccard and GCN-SVD are both designed for GCN and thus do not apply to GAT.

In all experiments, we train 300 epochs using the Adam optimizer with a learning rate of 0.01 for GCN models and 0.05 for GAT models. The architectures of the GNNs are the same as in §5.1.

Results. In Figures 2a–2c, we show the results from the experiments on GCNs against attacks and defenses. In each figure, each bar has two colors, representing a method with non-regularized training and with Lipschitz regularized training. From these results, we can draw the following conclusions. First, when used alone, our Lipschitz regularization consistently and significantly improves the robustness of GCN. Also, in some cases, Lipschitz regularization alone without any defense achieves state-of-the-art test accuracy. Second, when used on top of adversarial defense methods, our Lipschitz regularization also consistently improves the test accuracy over the original defense in all cases. Although there is no single defense method excels all datasets and all attacks, using our Lipschitz regularization will always enhance the robustness of GCN and thus the defense performance.

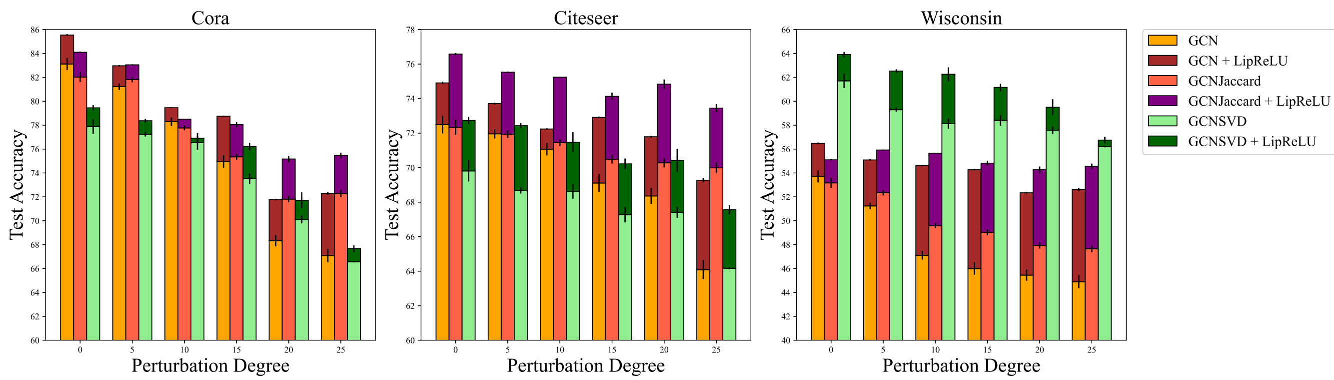
Next, we report the defense results for GATs under all untargeted attacks. The results are presented in Figure 3. Again, each bar uses two colors to compare the non-regularized and Lipschitz regularized settings. It is clear from the results that Lipschitz regularization consistently enhances robustness of GAT against all three attacking methods. In particular, for Citeseer and Wisconsin, the enhancement is significant in most cases. For instance, for the Wisconsin dataset, the test accuracy with clean data and with ptb% = 25% differ only by less than 2% when the GAT is Lipschitz regularized, but they can differ by more than 10% without regularization.

Table 4: The Lipschitz bounds corresponding to Figure 2a

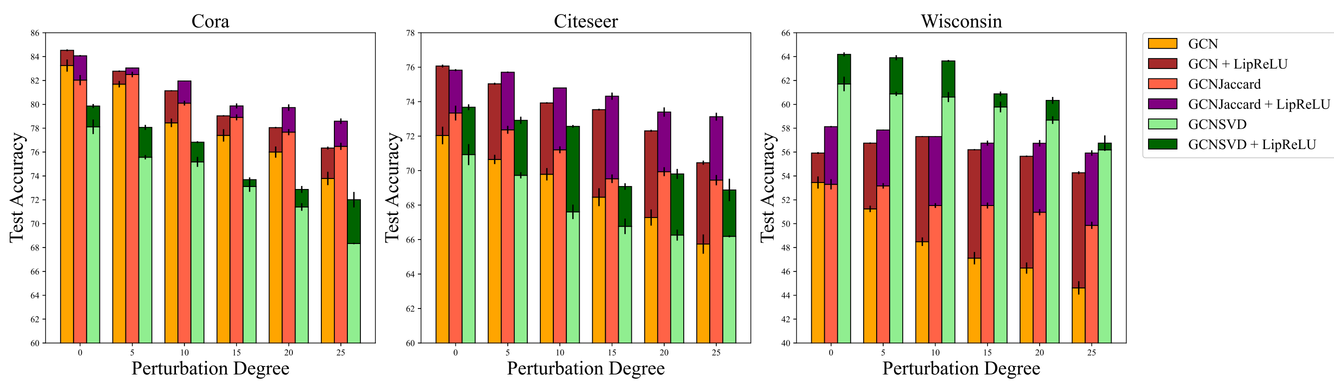
Dataset	ptb%	0	5	10	15	20	25
Cora	GCN	28.40±1.68	102.75±9.00	110.27±13.40	115.17±7.11	123.59±2.18	122.99±2.97
	GCN+LipReLU	9.17±0.35	33.48±4.53	46.41±0.44	50.86±2.02	43.51±3.34	43.95±4.71
	Jaccard	113.96±1.46	112.62±6.35	107.96±9.72	101.30±4.38	115.25±6.11	136.52±1.64
	Jaccard + LipReLU	44.80±0.55	41.22±1.69	45.97±1.48	46.04±3.53	41.90±1.08	49.16±3.36
	SVD	214.03±6.35	254.74±30.76	259.23±12.03	264.31±26.30	231.66±24.24	216.65±11.94
	SVD + LipReLU	42.61±5.46	42.16±0.73	31.92±1.63	132.32±11.29	165.93±9.24	102.02±0.48
Citeseer	GCN	23.07±0.91	22.56±0.73	87.57±3.53	77.59±8.09	81.45±9.18	77.53±9.94
	GCN+LipReLU	2.42±0.38	3.20±0.17	14.80±3.99	10.37±0.71	15.26±1.06	15.58±2.04
	Jaccard	85.75±5.22	85.90±4.36	91.99±10.42	87.47±5.49	83.47±4.29	87.54±1.37
	Jaccard + LipReLU	11.31±0.52	11.12±0.71	15.25±2.90	16.45±0.28	13.02±1.07	12.48±0.82
	SVD	107.89±1.73	83.02±0.66	76.97±1.90	90.26±1.66	73.12±1.12	84.25±2.92
	SVD + LipReLU	15.92±1.10	16.51±3.07	13.09±0.87	17.56±3.26	20.03±1.87	22.44±3.64
Wisconsin	GCN	18.97±4.32	15.02±0.54	15.91±1.55	20.30±3.95	14.79±0.93	29.08±0.51
	GCN+LipReLU	1.07±0.04	1.26±0.09	0.71±0.09	1.35±0.09	0.49±0.02	0.84±0.24
	Jaccard	24.53±2.74	30.78±7.68	30.89±8.20	27.98±3.89	24.11±5.68	31.45±1.57
	Jaccard + LipReLU	15.40±0.03	6.17±0.35	2.66±0.04	2.03±1.24	5.91±0.38	5.83±0.40
	SVD	50.88±3.46	38.30±2.69	32.91±5.88	44.16±2.38	33.25±0.51	33.02±1.22
	SVD + LipReLU	27.80±0.77	23.95±0.75	10.04±0.72	27.29±2.58	20.58±0.14	9.24±0.73

For completeness, we also report the numerics corresponding to Figures 2 and 3 in Appendix B. As summarized from these experimental results, when performing adversarial defense methods, it is beneficial to use our Lipschitz regularization as an individual add-on component in GNN training.

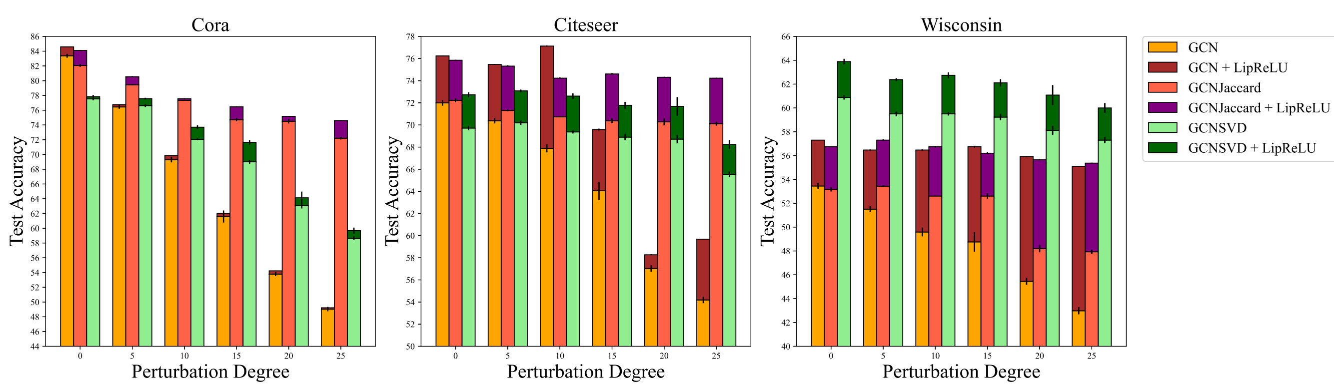
We also examine the Lipschitz bounds of both GCN and GAT after training. We report the bounds of the GCNs under Topology attack in Table 4, and report the bounds of the GATs under all attacks in Table 5. For completeness, we also present the bounds of the GCNs under DICE and Metaattack in Appendix B. These tables clearly reveal that the Lipschitz bounds of the GCNs and GATs after regularized training are much smaller than their non-regularized counterparts. We attribute robustness of the regularized models to these smaller Lipschitz bounds.



(a) The test accuracy under Topology Attack on GCN



(b) The test accuracy under DICE Attack on GCN



(c) The test accuracy under Metaattack on GCN

Figure 2: The test accuracy under adversarial attacks on GCN

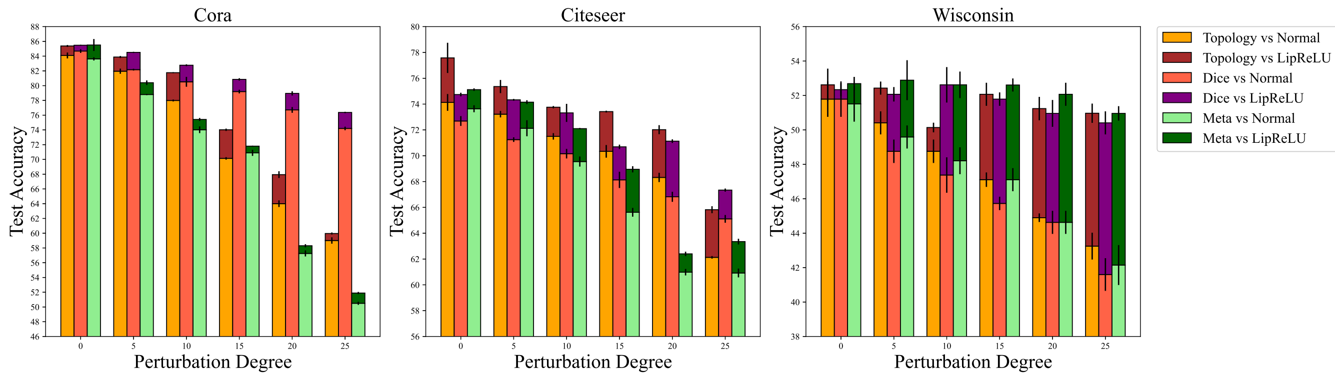


Figure 3: The test accuracy under untargeted attacks on GAT

Table 5: The Lipschitz bounds corresponding to Figure 3

Dataset	ptb%	0	5	10	15	20	25
Topology attack							
Cora	GAT	90.89±2.86	145.40±3.22	145.92±19.95	189.35±8.08	127.00±4.72	120.66±7.85
	GAT+LipReLU	39.70±0.85	37.33±1.56	36.48±0.84	83.32±5.38	39.60±0.58	45.33±0.48
Citeseer	GAT	51.08±8.23	56.63±4.00	63.16±3.46	58.68±2.81	65.88±9.01	64.52±4.40
	GAT+LipReLU	3.72±0.36	18.30±0.07	16.65±0.98	16.78±0.10	14.75±1.22	15.58±0.53
Wisconsin	GAT	241.01±6.88	162.12±2.96	281.07±3.96	273.83±10.70	229.10±3.67	138.15±6.37
	GAT+LipReLU	2.95±0.39	6.67±0.86	6.44±1.54	4.12±0.89	3.10±1.14	9.37±1.17
DICE							
Cora	GAT	170.56±8.33	206.26±9.10	279.84±9.00	241.84±5.33	163.75±8.87	251.11±1.63
	GAT+LipReLU	14.31±1.05	37.67±0.40	54.68±2.99	52.19±0.62	38.74±1.66	52.95±9.94
Citeseer	GAT	69.68±2.39	57.82±5.48	63.44±2.26	92.43±2.37	62.62±3.99	56.03±8.43
	GAT+LipReLU	4.89±0.14	4.97±0.61	17.86±0.20	17.44±0.36	15.00±0.86	14.96±1.35
Wisconsin	GAT	231.03±7.87	200.97±23.13	236.82±18.30	151.71±6.79	196.91±2.85	230.18±8.18
	GAT+LipReLU	6.78±1.85	10.99±3.13	12.88±3.36	9.29±2.96	8.05±3.10	8.21±2.43
Metaattack							
Cora	GAT	184.54±7.84	222.14±18.91	133.09±24.66	140.32±22.29	149.34±16.69	136.97±7.81
	GAT+LipReLU	8.09±0.18	8.75±0.81	8.45±0.38	25.26±1.55	29.34±0.32	28.43±0.11
Citeseer	GAT	39.52±10.79	28.82±4.64	45.69±8.27	26.80±1.25	34.65±3.23	33.36±2.34
	GAT+LipReLU	3.60±0.22	3.64±0.17	6.85±1.80	7.39±0.33	7.56±0.06	14.98±4.78
Wisconsin	GAT	217.29±22.91	193.31±16.94	241.97±16.98	167.66±19.38	189.97±17.43	224.12±15.50
	GAT+LipReLU	1.85±0.48	3.48±2.36	3.51±1.72	2.34±0.11	5.29±2.23	5.89±3.54

6 CONCLUSION AND LIMITATION

In this paper, we analyzed Lipschitz constants of GNNs for node-level tasks. We derived generic upper bounds for the constants as well as closed-form bounds for well-known GNNs including GCN, GraphSAGE and GAT.

We applied the Lipschitz bounds to regularization of GNNs. Our experiments showed that our Lipschitz regularization significantly improved robustness of the GNNs against perturbations on both input features and graph topology. Moreover, it can serve as a plug-in component used together with adversarial defense methods, which can advance state-of-the-art defense performances.

We discuss some limitations of this work. First, our estimation of Lipschitz bounds is derived for node-level tasks. For edge-level and graph-level tasks, a different definition is needed and we leave it to future exploration. Second, Lipschitz methods are more suitable for small graph datasets. For large datasets, incorporating the calculation of Lipschitz bounds increase the time complexity and we will explore more efficient implementation in future work.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (No. 62172173).

REFERENCES

[1] Alexandre Araujo, Benjamin Negrevergne, Yann Chevaleyre, and Jamal Atif. 2021. On Lipschitz regularization of convolutional layers using toeplitz matrix theory. In *35th AAAI Conference on Artificial Intelligence, Vancouver, Canada*.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 214–223.

[3] Aleksandar Bojchevski and Stephan Günnemann. 2019. Certifiable robustness to graph perturbations. *Advances in Neural Information Processing Systems* 32 (2019).

[4] Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. 2020. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *International Conference on Machine Learning*. PMLR, 1003–1013.

[5] Chen Cai and Yusu Wang. 2020. A note on over-smoothing for graph neural networks. In *ICML2020 Workshop on Graph Representation Learning and Beyond*.

[6] Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. 2020. Jacobian Adversarially Regularized Networks for Robustness. In *International Conference on Learning Representations*.

[7] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3438–3445.

[8] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial intelligence* 118, 1-2 (2000), 69–113.

[9] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.

[10] George Dasoulas, Kevin Scaman, and Aladin Virmaux. 2021. Lipschitz normalization for self-attention layers with application to graph neural networks. In *International Conference on Machine Learning*. PMLR, 2456–2466.

[11] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. 2020. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 169–177.

[12] Filippo Fabiani and Paul J Goulart. 2022. Neural network controllers for uncertain linear systems. *arXiv preprint arXiv:2204.13209* (2022).

[13] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. 2019. Efficient and accurate estimation of Lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems* 32 (2019).

[14] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[15] Chris Finlay, Jeff Calder, Bilal Abbasi, and Adam Oberman. 2018. Lipschitz regularized deep neural networks generalize and are adversarially robust. *arXiv preprint arXiv:1808.09540* (2018).

[16] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. 2020. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing* 68 (2020), 5680–5695.

[17] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. 2019. Stability of graph scattering transforms. *Advances in Neural Information Processing Systems* 32 (2019).

[18] Fernando Gama and Somayeh Sojoudi. 2022. Distributed linear-quadratic control with graph neural networks. *Signal Processing* 196 (2022), 108506.

[19] Feng Gao, Guy Wolf, and Matthew Hirn. 2019. Geometric scattering for graph data analysis. In *International Conference on Machine Learning*. PMLR, 2122–2131.

[20] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems* 34 (2021), 7637–7649.

[21] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

[22] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. 2021. Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning* 110, 2 (2021), 393–416.

[23] Xin Guo, Luisa Polania, Bin Zhu, Charles Boncelet, and Kenneth Barner. 2020. Graph neural networks for image understanding based on multiple cues: Group emotion recognition and event recognition as use cases. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2921–2930.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[25] Yujia Huang, Huan Zhang, Yuanyuan Shi, J Zico Kolter, and Anima Anandkumar. 2021. Training Certifiably Robust Neural Networks with Efficient Local Lipschitz Bounds. *Advances in Neural Information Processing Systems* 34 (2021).

[26] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. 2018. Limitations of the lipschitz constant as a defense against adversarial examples. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 16–29.

[27] Vassilis N Ioannidis, Dimitris Berberidis, and Georgios B Giannakis. 2019. GraphSAC: Detecting anomalies in large-scale graphs. *arXiv preprint arXiv:1910.09589* (2019).

[28] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, and Neil Zhenqiang Gong. 2020. Certified robustness of community detection against adversarial structural perturbation via randomized smoothing. In *Proceedings of The Web Conference*. 2718–2724.

[29] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*. PMLR, 2323–2332.

[30] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. 2021. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter* 22, 2 (2021), 19–34.

[31] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 66–74.

[32] Matt Jordan and Alexandros G Dimakis. 2020. Exactly computing the local Lipschitz constant of relu networks. *Advances in Neural Information Processing Systems* 33 (2020), 7344–7353.

[33] Nicolas Keriven, Alberto Bietti, and Samuel Vaiter. 2020. Convergence and stability of graph convolutional networks on large random graphs. *Advances in Neural Information Processing Systems* 33 (2020), 21512–21523.

[34] Hyunjik Kim, George Papamakarios, and Andriy Mnih. 2021. The Lipschitz constant of self-attention. In *International Conference on Machine Learning*. PMLR, 5562–5571.

[35] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[36] Fabian Latorre, Paul Rolland, and Volkan Cevher. 2020. Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *International Conference on Learning Representations*.

[37] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. 2021. DeepRobust: a Platform for Adversarial Attacks and Defenses. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 18 (May 2021), 16078–16080. https://doi.org/10.1609/aaai.v35i18.18017

[38] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

[39] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.

[40] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.

[41] Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. 2022. Are Defenses for Graph Neural Networks Robust?. In *Advances in Neural Information Processing Systems*.

[42] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. 2017. Exploring generalization in deep learning. *Advances in neural information processing systems* 30 (2017).

[43] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. 2018. Reachability Analysis of Deep Neural Networks with Provable Guarantees. In *IJCAI*.

[44] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

[45] Shuchang Tao, Huawei Shen, Qi Cao, Liang Hou, and Xueqi Cheng. 2020. Adversarial immunization for improving certifiable robustness on graphs. (2020).

[46] Matthieu Terris, Audrey Repetti, Jean-Christophe Pesquet, and Yves Wiaux. 2020. Building firmly nonexpansive convolutional neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 8658–8662.

[47] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

[48] Saurabh Verma and Zhi-Li Zhang. 2019. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1539–1548.

[49] Aladin Virmaux and Kevin Scaman. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems* 31 (2018).

[50] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (2020), 396–413.

[51] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour* 2, 2 (2018), 139–147.

[52] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. (7 2019), 4816–4823. https://doi.org/10.24963/ijcai.2019/669

[53] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)* (2020).

[54] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing* 17, 2 (2020), 151–178.

[55] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. (7 2019), 3961–3967. https://doi.org/10.24963/ijcai.2019/550

[56] Xiaojun Xu, Yue Yu, Bo Li, Le Song, Chengfeng Liu, and Carl Gunter. 2018. Characterizing malicious edges targeting on graph neural networks. (2018).

[57] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.

[58] Yuichi Yoshida and Takeru Miyato. 2017. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941* (2017).

[59] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. 2019. Three Mechanisms of Weight Decay Regularization. In *International Conference on Learning Representations*.

[60] Xiang Zhang and Marinka Zitnik. 2020. GNNGuard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems* 33 (2020), 9263–9275.

[61] Lingxiao Zhao and Leman Akoglu. 2020. Pairnorm: Tackling oversmoothing in GNNs. In *International Conference on Learning Representations*.

[62] Xin Zhao, Zeru Zhang, Zijie Zhang, Lingfei Wu, Jiayin Jin, Yang Zhou, Ruoming Jin, Dejing Dou, and Da Yan. 2021. Expressive 1-lipschitz neural networks for robust multiple graph learning against adversarial attacks. In *International Conference on Machine Learning*. PMLR, 12719–12735.

[63] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. 2020. Towards deeper graph neural networks with differentiable group normalization. *Advances in Neural Information Processing Systems* 33 (2020), 4917–4928.

[64] Kaixiong Zhou, Xiao Huang, Daochen Zha, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. 2021. Dirichlet energy constrained learning for deep graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021).

[65] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1399–1407.

[66] Dongmian Zou, Radu Balan, and Maneesh Singh. 2019. On Lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory* 66, 3 (2019), 1738–1759.

[67] Dongmian Zou and Gilad Lerman. 2020. Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis* 49, 3 (2020), 1046–1074.

[68] 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*. 2847–2856.

[69] Daniel Zügner, Oliver Borchert, Amir Akbarnejad, and Stephan Günnemann. 2020. Adversarial attacks on graph neural networks: Perturbations and their patterns. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 5 (2020), 1–31.

[70] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations*.

A PROOF OF RESULTS

A.1 Proof of Lemma 4.1

Proof. First notice that

$$\begin{aligned}\frac{\|g(\boldsymbol{x})-g(\boldsymbol{y})\|}{\|\boldsymbol{x}-\boldsymbol{y}\|} &= \frac{\left\|[g_i(\boldsymbol{x})-g_i(\boldsymbol{y})]_{i=1}^n\right\|}{\|\boldsymbol{x}-\boldsymbol{y}\|} \\ &= \left\|\left[\frac{|g_i(\boldsymbol{x})-g_i(\boldsymbol{y})|}{\|\boldsymbol{x}-\boldsymbol{y}\|}\right]_{i=1}^n\right\|.\end{aligned} \tag{23}$$

Moreover, for each $i \in [n]$, $\frac{|g_i(\boldsymbol{x})-g_i(\boldsymbol{y})|}{\|\boldsymbol{x}-\boldsymbol{y}\|} \leq \mathrm{Lip}(g_i)$. Therefore,

$$\begin{aligned}\mathrm{Lip}(g) &= \sup_{\boldsymbol{x}\neq\boldsymbol{y}} \frac{\|g(\boldsymbol{x})-g(\boldsymbol{y})\|}{\|\boldsymbol{x}-\boldsymbol{y}\|} \\ &= \sup_{\boldsymbol{x}\neq\boldsymbol{y}} \left\|\left[\frac{|g_i(\boldsymbol{x})-g_i(\boldsymbol{y})|}{\|\boldsymbol{x}-\boldsymbol{y}\|}\right]_{i=1}^n\right\| \\ &\leq \sup_{\boldsymbol{x}\neq\boldsymbol{y}} \left\|[\mathrm{Lip}(g_i)]_{i=1}^n\right\| = \left\|[\mathrm{Lip}(g_i)]_{i=1}^n\right\|.\end{aligned} \tag{24}$$

□

A.2 Proof of Theorem 4.2

Proof. Given an input feature $\boldsymbol{X}$, GCNConv produces the output $\boldsymbol{Z} = \mathrm{GCNConv}(\boldsymbol{X}) = \hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}$. In particular, fixing $i \in [N]$ and $k \in [F']$, we obtain

$$\begin{aligned}Z_{ik} &= \hat{\boldsymbol{A}}_{i,:}\boldsymbol{X}\boldsymbol{W}_{:,k} \\ &= \hat{A}_{ii}\boldsymbol{X}_{i,:}\boldsymbol{W}_{:,k} + \sum_{j\neq i}\hat{A}_{ij}\boldsymbol{X}_{j,:}\boldsymbol{W}_{:,k}.\end{aligned} \tag{25}$$

Let $\tilde{\boldsymbol{X}}$ be another input feature so that $\tilde{\boldsymbol{X}}_{j,:} = \boldsymbol{X}_{j,:}$ for $j \neq i$, and $\tilde{\boldsymbol{Z}} = \mathrm{GCNConv}(\tilde{\boldsymbol{X}})$. We obtain

$$\tilde{Z}_{ik} = \hat{A}_{ii}\tilde{\boldsymbol{X}}_{i,:}\boldsymbol{W}_{:,k} + \sum_{j\neq i}\hat{A}_{ij}\tilde{\boldsymbol{X}}_{j,:}\boldsymbol{W}_{:,k}. \tag{26}$$

Combining (25) and (26) yields

$$\tilde{Z}_{ik} - Z_{ik} = \hat{A}_{ii}(\tilde{\boldsymbol{X}}_{i,:} - \boldsymbol{X}_{i,:})\boldsymbol{W}_{:,k}. \tag{27}$$

Hence,

$$\mathrm{Lip}(\mathrm{GCNConv}_{ik}) = \sup_{\boldsymbol{X},\tilde{\boldsymbol{X}}} \frac{\left|\tilde{Z}_{ik} - Z_{ik}\right|}{\left\|\tilde{\boldsymbol{X}}_{i,:} - \boldsymbol{X}_{i,:}\right\|} \leq \left|\hat{A}_{ii}\right| \left\|W_{:,k}\right\|. \tag{28}$$

Therefore, the Lipschitz bound of GCNConv satisfies

$$\mathrm{LB}(\mathrm{GCNConv}) \leq \max_{i\in[N]} \left\|\left[\left|\hat{A}_{ii}\right|\left\|\boldsymbol{W}_{:,k}\right\|\right]_{k=1}^{F'}\right\|. \tag{29}$$

□

A.3 Proof of Theorem 4.3

Proof. For an input feature $\boldsymbol{X}$, a SAGEConv layer produces $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}_1 + \boldsymbol{V}\sigma(\boldsymbol{X}\boldsymbol{W}_2 + \boldsymbol{1}\otimes\boldsymbol{b}^\top)\boldsymbol{W}_3$. In particular, fixing $i \in [N]$ and $k \in [F']$, we have

$$\begin{aligned}Z_{ik} &= \boldsymbol{X}_{i,:}(\boldsymbol{W}_1)_{:,k} + \boldsymbol{V}_{i,:}\sigma(\boldsymbol{X}\boldsymbol{W}_2 + \boldsymbol{1}\otimes\boldsymbol{b}^\top)(\boldsymbol{W}_3)_{:,k} \\ &= \boldsymbol{X}_{i,:}(\boldsymbol{W}_1)_{:,k} + V_{ii}\sigma(\boldsymbol{X}_i\boldsymbol{W}_2 + \boldsymbol{1}\otimes\boldsymbol{b}^\top)(\boldsymbol{W}_3)_{:,k} + \\ &\quad \sum_{j\neq i} V_{ij}\sigma(\boldsymbol{X}_j\boldsymbol{W}_2 + \boldsymbol{1}\otimes\boldsymbol{b}^\top)(\boldsymbol{W}_3)_{:,k}.\end{aligned} \tag{30}$$

Let $\tilde{\boldsymbol{X}}$ be another input feature so that $\tilde{\boldsymbol{X}}_{j,:} = \boldsymbol{X}_{j,:}$ for $j \neq i$, and $\tilde{\boldsymbol{Z}} = \mathrm{GCNConv}(\tilde{\boldsymbol{X}})$. Since σ (typically ReLU) is 1-Lipschitz, similar to (25), we obtain

$$\begin{aligned}\mathrm{Lip}(\mathrm{SAGEConv}_{ik}) &= \sup_{\boldsymbol{X},\tilde{\boldsymbol{X}}} \frac{\left|\tilde{Z}_{ik} - Z_{ik}\right|}{\left\|\tilde{\boldsymbol{X}}_{i,:} - \boldsymbol{X}_{i,:}\right\|} \\ &\leq \left\|(\boldsymbol{W}_1)_{:,k}\right\| + V_{ii}\left\|(\boldsymbol{W}_2\boldsymbol{W}_3)_{:,k}\right\|.\end{aligned} \tag{31}$$

Therefore the Lipschitz bound of GraphSAGE satisfies

$$\mathrm{LB}(\mathrm{SAGEConv}) \leq \max_{i\in[N]} \left\|\left[\left\|(\boldsymbol{W}_1)_{:,k}\right\| + V_{ii}\left\|(\boldsymbol{W}_2\boldsymbol{W}_3)_{:,k}\right\|\right]_{k=1}^{F'}\right\|. \tag{32}$$

□

A.4 Proof of Theorem 4.4

Proof. For an input feature $\boldsymbol{X}$, a GATConv layer produces

$$\boldsymbol{Z} = \boldsymbol{S}\boldsymbol{X}\boldsymbol{W}. \tag{33}$$

In particular, fixing $i \in [N]$ and $k \in [F']$, we obtain

$$\begin{aligned}Z_{ik} &= \boldsymbol{S}_{i,:}\boldsymbol{X}\boldsymbol{W}_{:,k} \\ &= S_{ii}\boldsymbol{X}_{i,:}\boldsymbol{W}_{:,k} + \sum_{i\neq j} S_{ij}\boldsymbol{X}_{j,:}\boldsymbol{W}_{:,k}.\end{aligned} \tag{34}$$

We consider local perturbations by taking the gradient as

$$D_{\boldsymbol{X}_{i,:}}(Z_{ik}) = D_{\boldsymbol{X}_{i,:}}(S_{ii})\boldsymbol{X}_{i,:}\boldsymbol{W}_{:,k} + S_{ii}\boldsymbol{W}_{:,k} + \sum_{j\neq i} D_{\boldsymbol{X}_{i,:}}(S_{ij})\boldsymbol{X}_{j,:}\boldsymbol{W}_{:,k}, \tag{35}$$

where we recall $S_{ij} = \frac{\exp(\sigma(\boldsymbol{a}[\boldsymbol{W}\boldsymbol{x}_i \parallel \boldsymbol{W}\boldsymbol{x}_j]))}{\sum_{k\in\mathcal{N}_i}\exp(\sigma(\boldsymbol{a}[\boldsymbol{W}\boldsymbol{x}_i \parallel \boldsymbol{W}\boldsymbol{x}_k]))}$. Since σ (typically leakyReLU) is 1-Lipschitz, when considering the effect on gradients, it is helpful to denote

$$\breve{S_{ij}} = \frac{\exp(\boldsymbol{a}[\boldsymbol{W}\boldsymbol{x}_i \parallel \boldsymbol{W}\boldsymbol{x}_j])}{\sum_{k\in\mathcal{N}_i}\exp(\boldsymbol{a}[\boldsymbol{W}\boldsymbol{x}_i \parallel \boldsymbol{W}\boldsymbol{x}_k])}. \tag{36}$$

Let $e_{ij} = \boldsymbol{a}[\boldsymbol{W}\boldsymbol{x}_i \parallel \boldsymbol{W}\boldsymbol{x}_j]$, we simply write $\breve{S_{ij}} = \frac{\exp e_{ij}}{\sum_{k\in\mathcal{N}_i}\exp e_{ik}}$. We further calculate

$$D_{\boldsymbol{X}_{i,:}}(e_{ij}) = \begin{cases}\boldsymbol{a}_1\boldsymbol{W}_{:,k} =: \boldsymbol{p}, & \text{if } i \neq j; \\ \boldsymbol{a}_1\boldsymbol{W}_{:,k} + \boldsymbol{a}_2\boldsymbol{W}_{:,k} =: \boldsymbol{q}, & \text{if } i = j.\end{cases} \tag{37}$$

Thus

$$D_{\boldsymbol{X}_{i,:}}(S_{ij}) = D_{\boldsymbol{X}_{i,:}}(\breve{S_{ij}}) = \frac{\exp(e_{ii})\exp(e_{ij})(\boldsymbol{p}-\boldsymbol{q})}{\sum^2_{k\in\mathcal{N}_i}\exp(e_{ik})}, \quad i \neq j, \tag{38}$$

while

$$D_{\boldsymbol{X}_{i,:}}(S_{ii}) = D_{\boldsymbol{X}_{i,:}}(\breve{S_{ii}}) = \frac{\exp(e_{ii})(\boldsymbol{p}-\boldsymbol{q})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})}\left(\frac{\exp(e_{ii})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})} - 1\right). \tag{39}$$

Combining (35), (38) and (39) yields

$$D_{\boldsymbol{X}_{i,:}}(Z_{ik}) = S_{ii}\boldsymbol{X}_{i,:}W_{:,k}\boldsymbol{v} - S_{ii}\sum_{j\in\mathcal{N}_i} S_{ij}\boldsymbol{X}_{j,:}\boldsymbol{W}_{:,k}\boldsymbol{v} + S_{ii}\boldsymbol{W}_{:,k}, \tag{40}$$

where $\boldsymbol{v} = \boldsymbol{q} - \boldsymbol{p}$. Taking norms on both sides of (40), we have

$$\begin{aligned} \left\|D_{\boldsymbol{X}_{i,:}}(Z_{ik})\right\| &= \left\|(S_{ii}\boldsymbol{X}_{i,:}W_{:,k} - S_{ii}\sum_{j\in\mathcal{N}_i} S_{ij}\boldsymbol{X}_{j,:}\boldsymbol{W}_{:,k})\boldsymbol{v} + S_{ii}\boldsymbol{W}_{:,k}\right\| \\ &\leq \left|S_{ii}\boldsymbol{X}_{i,:}W_{:,k} - S_{ii}\sum_{j\in\mathcal{N}_i} S_{ij}\boldsymbol{X}_{j,:}\boldsymbol{W}_{:,k}\right|\|\boldsymbol{v}\| + S_{ii}\left\|\boldsymbol{W}_{:,k}\right\|, \end{aligned} \tag{41}$$

Therefore we can derive the Lipschitz bound of GATConv Lipschitz matrix

$$\mathrm{Lip}(\mathrm{GATConv}_{ik}) \leq \left|S_{ii}\boldsymbol{X}_{i,:}\boldsymbol{W}_{:,k} - S_{ii}\sum_{j\in\mathcal{N}_i} S_{ij}\boldsymbol{X}_{j,:}\boldsymbol{W}_{:,k}\right|\|\boldsymbol{v}\| + S_{ii}\left\|\boldsymbol{W}_{:,k}\right\|, \tag{42}$$

which is equivalent to (16). (17) then follows a standard argument on limits. □

B MORE EXPERIMENTS ON ADVERSARIAL ATTACKS AND DEFEND METHODS

We present experiments on larger datasets including Pubmed [57] and ogbn-arxiv [50]. Table 6 reports the test accuracy of GCN using noisy data, where the setting is the same as §5.1. Table 7 reports the test accuracy of GCN under adversarial attacks designed for large graphs including GR-BCD and PR-BCD [20], with various budgets for the attack. We observe that LipReLU can consistently improve the performance of GNN in both cases.

Furthermore, we present more experimental results and more details at https://github.com/TechnologyAiGroup. Specifically, this repository includes the following and will be updated if new results are available:

(1) The experimental results for Figures 2 and 3 of §5.2 presented as numerics.
(2) The Lipschitz bounds of trained GCNs corresponding to Figures 2b and 2c of §5.2.
(3) The run time of various GNNs for Figures 2 and 3 of §5.2.
(4) The experimental results for additional baseline methods including GNNGuard [60] and RGCN [65].

Table 6: The test accuracy for additional noisy graph data using GCN

Dataset	Layer	Normal	L2-Reg	LipReLU
Pubmed	2	52.25±0.14	54.65±0.24	**56.36**±0.12
	3	55.66±0.22	57.06±0.33	**61.86**±0.12
	4	56.16±0.27	51.45±0.45	**59.16**±0.65
	5	59.86±0.33	57.36±0.45	**63.23**±0.21
ogbn-arxiv	2	47.46±0.44	48.18±0.52	**51.02**±0.34
	3	47.02±0.10	47.75±0.22	**50.90**±0.34
	4	47.75±0.11	48.42±0.33	**50.73**±0.29
	5	48.12±0.14	49.65±0.44	**50.19**±0.52

Table 7: The test accuracy under adversarial attacks for obgn-arxiv

Attack	method/budget	0.01	0.05	0.10
GR-BCD	GCN	51.82±0.32	46.36±0.18	42.04±0.27
	GCN+LipReLU	54.51±0.23	48.91±0.56	42.97±0.19
PR-BCD	GCN	52.27±0.12	44.95±0.08	39.59±0.56
	GCN+LipReLU	53.57±0.44	46.40±0.12	42.10±0.25