CS 199: Implementing Model Robustness Algorithms and Graph Convolutional Network Robustness Certification

Lucas Tecot
Computer Science
UCLA
Los Angeles, CA
UID: 704611946
lucastecot@gmail.com

Abstract

Certifying robustness in neural networks by finding a lower bound of the minimum adversarial distortion is desirable not only for network guarantees, but also for optimization to improve model robustness. Previous work has shown that efficient algorithms can be used to achieve non-trivial bounds on neural networks. We extend this work by introducing an algorithm to provide similar bounds for graph convolutional networks, and by implementing existing algorithms for the purposes of further experimentation.

1 Introduction

Providing certifiable perturbation bounds of neural networks is a growing field of research with the goal of preventing adversarial attacks on networks. For the contexts of project, the efforts were two fold. Firstly, we implemented the equivalent of the CROWN [8] algorithm in Pytorch by modifying code developed by Wong and Kolter [3]. Secondly, we derived a CROWN-like algorithm to compute robustness bounds on graph convolutional networks.

2 Implementing CROWN in Pytorch

Previous work had found that algorithms such as CROWN [8] were capable of producing much tighter bounds than that of others such as fast-lin [7]. As such, we wanted to experiment and see if we could improve model robustness during training via gradient descent on the maximum output perturbation bound produced by CROWN.

Unfortunately our existing implementations of CROWN only exist in Numpy, which doesn't support desirable training features such as autodiff and various utilities. However, we had previously verified that the code developed by Wong and Kolter [3] produced identical bounds to that of fast-lin. Their code is implemented in Pytorch, which provides all the desired functionality we needed. As such, we decided to set out to modify said code into CROWN.

The difference between fast-lin and CROWN is relatively simple to understand. Essentially in both algorithms we require defining linear lower and upper bounds for the non-linear activation of each neuron. In fast-lin, the slopes of each of these bounds are set to be identical. However, in CROWN they remove this constraint, and simply choose a slope that minimizes the average distance from the bounds to the actual value of the non-linear activation.

Preprint. Work in progress.

However, Wong and Kolter define their bounds in a dual formulation, which made it difficult initially to understand how it translated to fast-lin. But after a lot of tinkering, it turned out to be a relatively simple modification to just the class representing the ReLu function in the bound computation.

Future work next quarter will involve training networks using this new bound, and making various modifications to further experiment on existing algorithms. (The code is in a private repository, so if you wish to see it please contact the author.)

3 GCN Robustness: Previous Work

Before we dive into the derivations for deriving GCN pertebation bounds, we'll first introduce the previous work.

3.1 Graph Convolutional Networks

Graph convolutional networks (GCNs) are a form of neural network designed to deal with graphical data [2] [1]. There are different flavors and formulations, but the general concept is that we add layers to a neural network that sum up data from nodes in the graph that are connected. This creates a form of input data embedding where each data point has been affected by the propagated effects of it's neighbors.

For the purposes of this paper, we'll use the following definitions:

$$H^{l+1} = \sigma(\tilde{D}^{-1}\tilde{A}H^{l}W^{l}) \qquad (H^{0} = X) \tag{1}$$

$$\tilde{A} = A + I_N \tag{2}$$

$$\tilde{D}_{ii} = \sum_{j} \tilde{A}_{ij} \tag{3}$$

σ is a non-linear activation, H^l are the values of layer l in the GCN, and W^l are trainable weights associated with said layer. A is the adjacency matrix, where $A_{ij} = 1$ if there is a connection in the graph between node i and j, and 0 otherwise. Note this is a symmetric matrix, so $A_{ij} = A_{ji}$. $\tilde{D}$ serves as a regularization term, such that the summation of the nodes doesn't change the scale of the feature vectors.

This is the notation used by Kipf and Welling [2], with the only difference being we use regular normalization whereas they use symmetric normalization ($H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l}W^{l})$).

3.2 Model Robustness

There have been a number of previous recent works that provide algorithms to derive lower bounds of adversarial distortion [4] [8] [5] [7] [6] [3] for multi-layer perceptrons and other common neural network architectures. Many of these algorithms work in a similar fashion, where we derive an upper and lower bound by propagating constraints from each neural network layer to the next. The main issue that must be dealt with is that the non-linear nature of the activation function requires the loosening of constraints such that the bounds can be found via tractable computation. Furthermore, these bounds can be used during gradient descent to improve the robustness guarantees of said network.

In the context of this paper, we will base our work off the CROWN algorithm introduced by Zhang et. al. [8]. We will copy some of the equations and summarize the method below, though we encourage the reader to look at the original paper for deeper details.

Definition 3.1 (Linear bounds on activation function) *For the r-th neuron in k-th layer with pre-activation bounds $\mathbf{l}_r^{(k)}$,$\mathbf{u}_r^{(k)}$ and the activation function $\sigma(y)$, define two linear functions $h_{U,r}^{(k)}, h_{L,r}^{(k)} : \mathbb{R} \to \mathbb{R}$, $h_{U,r}^{(k)}(y) = \alpha_{U,r}^{(k)}(y + \beta_{U,r}^{(k)})$, $h_{L,r}^{(k)}(y) = \alpha_{L,r}^{(k)}(y + \beta_{L,r}^{(k)})$, such that $h_{L,r}^{(k)}(y) \leq \sigma(y) \leq h_{U,r}^{(k)}(y)$, $y \in [\mathbf{l}_r^{(k)}, \mathbf{u}_r^{(k)}]$, $\forall k \in [m-1], r \in [n_k]$ and $\alpha_{U,r}^{(k)}, \alpha_{L,r}^{(k)} \in \mathbb{R}^+, \beta_{U,r}^{(k)}, \beta_{L,r}^{(k)} \in \mathbb{R}$.*

Theorem 3.2 (Explicit output bounds of neural network f) *Given an m-layer neural network function $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$, there exists two explicit functions $f_j^L : \mathbb{R}^{n_0} \to \mathbb{R}$ and $f_j^U : \mathbb{R}^{n_0} \to \mathbb{R}$ such that $\forall j \in [n_m],\ \forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$, the inequality $f_j^L(\mathbf{x}) \leq f_j(\mathbf{x}) \leq f_j^U(\mathbf{x})$ holds true, where*

$$f_j^U(\mathbf{x}) = \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^{m} \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}), \quad f_j^L(\mathbf{x}) = \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x} + \sum_{k=1}^{m} \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}), \tag{4}$$

$$\mathbf{\Lambda}_{j,:}^{(k-1)} = \begin{cases} \mathbf{e}_j^\top & \text{if } k = m+1; \\ (\mathbf{\Lambda}_{j,:}^{(k)}\mathbf{W}^{(k)}) \odot \lambda_{j,:}^{(k-1)} & \text{if } k \in [m]. \end{cases} \quad \mathbf{\Omega}_{j,:}^{(k-1)} = \begin{cases} \mathbf{e}_j^\top & \text{if } k = m+1; \\ (\mathbf{\Omega}_{j,:}^{(k)}\mathbf{W}^{(k)}) \odot \omega_{j,:}^{(k-1)} & \text{if } k \in [m]. \end{cases}$$

and $\forall i \in [n_k]$,

$$\lambda_{j,i}^{(k)} = \begin{cases} \alpha_{U,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,i}^{(k)} > 0; \\ \alpha_{L,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,i}^{(k)} < 0; \\ 1 & \text{if } k = 0. \end{cases} \quad \omega_{j,i}^{(k)} = \begin{cases} \alpha_{L,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,i}^{(k)} > 0; \\ \alpha_{U,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,i}^{(k)} < 0; \\ 1 & \text{if } k = 0. \end{cases}$$

$$\mathbf{\Delta}_{i,j}^{(k)} = \begin{cases} \beta_{U,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,i}^{(k)} > 0; \\ \beta_{L,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,i}^{(k)} < 0; \\ 0 & \text{if } k = m. \end{cases} \quad \mathbf{\Theta}_{i,j}^{(k)} = \begin{cases} \beta_{L,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,i}^{(k)} > 0; \\ \beta_{U,i}^{(k)} & \text{if } k \in [m-1],\ \mathbf{\Lambda}_{j,i}^{(k)} < 0; \\ 0 & \text{if } k = m. \end{cases}$$

and $\odot$ is the Hadamard product.

Corollary 3.3 (Closed-form global bounds) *Given a data point $\mathbf{x_0} \in \mathbb{R}^{n_0}$, ℓ_p ball parameters $p \geq 1$ and $\epsilon > 0$. For an m-layer neural network function $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$, there exist two fixed values γ_j^L and γ_j^U such that $\forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ and $\forall j \in [n_m]$, $1/q = 1 - 1/p$, the inequality $\gamma_j^L \leq f_j(\mathbf{x}) \leq \gamma_j^U$ holds true, and*

$$\gamma_j^U = \epsilon\|\mathbf{\Lambda}_{j,:}^{(0)}\|_q + \mathbf{\Lambda}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^{m} \mathbf{\Lambda}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Delta}_{:,j}^{(k)}),\ \gamma_j^L = -\epsilon\|\mathbf{\Omega}_{j,:}^{(0)}\|_q + \mathbf{\Omega}_{j,:}^{(0)}\mathbf{x_0} + \sum_{k=1}^{m} \mathbf{\Omega}_{j,:}^{(k)}(\mathbf{b}^{(k)} + \mathbf{\Theta}_{:,j}^{(k)}). \tag{5}$$

Essentially this algorithm works in a recursive back-propagation fashion, in which we simply combine the linear bounds from the above layers of the network with the next layer until we get a full description of how our given model can perturb input. Note that in order to do this we must provide linear equations to upper and lower bound the possible outputs of the non-linear activations, represented by the α and β parameters. These can be different for each neuron, and can be set heuristically or included as a parameter to be optimized over through gradient descent. This is generally possible for most common non-linear activations. We encourage the readers to look at the CROWN [8] and DeepPoly [5] papers in order to gain a detailed understanding of how and why this is done.

4 Derivation of GCN pre-activation bounds

Note that essentially the only difference between applying CROWN to a regular neural network and a GCN is the adjacency matrix. This means that the bounds we can produce for each layer will depend on the other data points in the input, whereas with the typical algorithm this is not the case. However, it turns out that having this sort of dependency makes defining the bounds in same form as the CROWN algorithm difficult.

As such, we will start by defining the pre-activation bounds of the network, $\mathbf{l}^{(k)}$ and $\mathbf{u}^{(k)}$. These bounds will be strictly looser than any bounds derived by a CROWN-like algorithm, but never the less will still provide some sort of guarantee and are an important to further minimizing our bounds.

Given an m-layer graph convolutional neural network function $f : \mathbb{R}^{N \times C_0} \to \mathbb{R}^{N \times C_m}$ for $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ and $\forall k \in [m-1]$, let the pre-activation inputs for the $[n,i]$-th neuron at layer k be $\mathbf{y}_{n,i}^{(k)} := [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(k-1)})]_{n,:}\mathbf{W}_{:,i}^{(m-1)}$. The $[n,j]$-th output of the neural network is the following:

$$f_{n,j}(\mathbf{x}) = \sum_{i=1}^{C_{m-1}} [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} \tag{6}$$

$$= \sum_{\mathbf{W}_{i,j}^{(m)} \geq 0} [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} + \sum_{\mathbf{W}_{i,j}^{(m)} < 0} [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} \tag{7}$$

Assume $y^{(m-1)}$ is bounded by two values, $\mathbf{l}^{(m-1)}, \mathbf{u}^{(m-1)}$. Therefore we have

$$\mathbf{l}^{(m-1)} \leq \mathbf{y}^{(m-1)} \leq \mathbf{u}^{(m-1)}.$$

Thus, if the associated weight $\mathbf{W}_{i,j}^{(m)}$ to the $[n,i]$-th neuron is non-negative, we have

$$[\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{l}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} \leq [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} \leq [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{u}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)}; \quad (8)$$

otherwise, we have

$$[\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{u}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} \leq [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} \leq [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{l}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)}. \quad (9)$$

under the assumption that σ is a non-decreasing function. Note that the elements of $\tilde{D}$ and $\tilde{A}$ are all positive, and as such will not affect the sign of these values.

Recursive bound. Lets compute $\mathbf{u}_{n,j}^{m}$, which is an upper bound of $f_{n,j}(\mathbf{x})$ at the output of the network. To compute this, (7), (8) and (9) are the key equations. Precisely, for the $\mathbf{W}_{i,j}^{(m)} \geq 0$ terms in (7), the upper bound is the right-hand-side (RHS) in (8); and for the $\mathbf{W}_{i,j}^{(m)} < 0$ terms in (7), the upper bound is the RHS in (9). Thus, we obtain:

$$\mathbf{u}_{n,j}^{(m)} = \sum_{\mathbf{W}_{i,j}^{(m)} \geq 0} [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{u}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} + \sum_{\mathbf{W}_{j,i}^{(m)} < 0} [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{l}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} \quad (10)$$

$$= \sum_{i=1}^{C_{m-1}} [\tilde{D}^{-1}\tilde{A}\sigma(\tilde{\mathbf{u}}_{:,:,j}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} \quad (11)$$

$$= \left[\tilde{D}^{-1}\tilde{A}\sigma(\tilde{\mathbf{u}}_{:,:,j}^{(m-1)})\mathbf{W}^{(m)}\right]_{n,j} \quad (12)$$

Where we define:

$$\tilde{\mathbf{u}}_{n,i,j}^{(m-1)} = \begin{cases} \mathbf{u}_{n,i}^{(m-1)} & \text{if } \mathbf{W}_{i,j}^{(m)} \geq 0; \\ \mathbf{l}_{n,i}^{(m-1)} & \text{if } \mathbf{W}_{i,j}^{(m)} < 0; \end{cases} \quad (13)$$

Note that we now have a recursive equation with which we can describe the upper bound. We can also apply this same derivation identically to the lower bound, except we select the lower bound when the corresponding weight is positive, and the upper when it is negative.

As such, we define the theorem as follows:

Theorem 4.1 (Pre-activation output bounds of GCN f) *Given an m-layer convolutional neural network function $f : \mathbb{R}^{N,C_0} \to \mathbb{R}^{N,C_m}$ with non-decreasing activation functions σ, there exists two matrices $\mathbf{l}^{(k)}$ and $\mathbf{u}^{(k)}$ such that $\forall k \in [m]$, $\forall j \in [C_k]$, $\forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$, the inequality $\mathbf{l}_{n,j}^{(k)} \leq f_{n,j}^{(k)}(\mathbf{x}) \leq \mathbf{u}_{n,j}^{(k)}$ holds true, where*

$$\mathbf{l}_{:,j}^{(k)} = \begin{cases} \tilde{D}^{-1}\tilde{A}\sigma(\tilde{\mathbf{l}}_{:,:,j}^{(k-1)})\mathbf{W}^{(k)} & \text{if } k \in [2,m] \\ \tilde{D}^{-1}\tilde{A}\tilde{\mathbf{l}}_{:,:,j}^{(k-1)}\mathbf{W}^{(k)} & \text{if } k = 1 \\ x - \epsilon & \text{if } k = 0 \end{cases}$$

$$\mathbf{u}_{:,j}^{(k)} = \begin{cases} \tilde{D}^{-1}\tilde{A}\sigma(\tilde{\mathbf{u}}_{:,:,j}^{(k-1)})\mathbf{W}^{(k)} & \text{if } k \in [2,m] \\ \tilde{D}^{-1}\tilde{A}\tilde{\mathbf{u}}_{:,:,j}^{(k-1)}\mathbf{W}^{(k)} & \text{if } k = 1 \\ x + \epsilon & \text{if } k = 0 \end{cases}$$

and $\forall n \in [N], \forall i \in [C_{k-1}]$

$$\tilde{\mathbf{l}}_{n,i,j}^{(k-1)} = \begin{cases} \mathbf{l}_{n,i}^{(k-1)} & \text{if } \mathbf{W}_{i,j}^{(k)} \geq 0; \\ \mathbf{u}_{n,i}^{(k-1)} & \text{if } \mathbf{W}_{i,j}^{(k)} < 0; \end{cases} \qquad \tilde{\mathbf{u}}_{n,i,j}^{(k-1)} = \begin{cases} \mathbf{u}_{n,i}^{(k-1)} & \text{if } \mathbf{W}_{i,j}^{(k)} \geq 0; \\ \mathbf{l}_{n,i}^{(k-1)} & \text{if } \mathbf{W}_{i,j}^{(k)} < 0; \end{cases}$$

where $f^k(\mathbf{x})$ is the pre-activation output of the network at layer k, and $f^m(\mathbf{x})$ corresponds to the output of the network.

This theorem allows us to derive the pre-activation bounds, and hence also output bounds, in a recursive manner.

5 Derivation of Relaxed CROWN bounds for GCN

We will now derive bounds for GCNs using an algorithm like CROWN [8]. It turns out that deriving these bounds in a similar fashion to CROWN is difficult, but we can accomplish it via additional relaxations on the non-linear activations.

Note that these proofs will rely on definitions similar to those of Definition 3.1, in which linear upper and lower bounds are found for a specific neuron's non-linear activation given a $\mathbf{l}^{(k)}$ and $\mathbf{u}^{(k)}$ (which Theorem 4.1 allows us to compute). We will not cover how to derive the α and β terms in these definitions as they are identical to the methods presented by various other papers [8] [5].

First we will formally define the theorem and corollary, then prove them below.

Definition 5.1 (Linear bounds on GCN activation function) *For the $[n, r]$-th neuron in the k-th layer of a GCN with pre-activation bounds $\mathbf{l}^{(k)}_{n,r}, \mathbf{u}^{(k)}_{n,r}$ and the activation function $\sigma(y)$, define two linear functions $h^{(k)}_{U,n,r}, h^{(k)}_{L,n,r} : \mathbb{R} \to \mathbb{R}$, $h^{(k)}_{U,n,r}(y) = \alpha^{(k)}_{U,n,r}(y + \beta^{(k)}_{U,n,r})$, $h^{(k)}_{L,n,r}(y) = \alpha^{(k)}_{L,n,r}(y + \beta^{(k)}_{L,n,r})$, such that $h^{(k)}_{L,n,r}(y) \leq \sigma(y) \leq h^{(k)}_{U,n,r}(y)$, $y \in [\mathbf{l}^{(k)}_{n,r}, \mathbf{u}^{(k)}_{n,r}]$, $\forall k \in [m-1], n \in [N], r \in [C_k]$ and $\alpha^{(k)}_{U,n,r}, \alpha^{(k)}_{L,n,r} \in \mathbb{R}^+, \beta^{(k)}_{U,n,r}, \beta^{(k)}_{L,n,r} \in \mathbb{R}$.*

Theorem 5.2 (Explicit output bounds of GCN f) *Given an m-layer GCN function $f : \mathbb{R}^{N \times C_0} \to \mathbb{R}^{N \times C_m}$, there exists two explicit functions $f^L_{n,j} : \mathbb{R}^{N \times C_0} \to \mathbb{R}$ and $f^U_{n,j} : \mathbb{R}^{N \times C_0} \to \mathbb{R}$ such that $\forall j \in [C_m]$, $\forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$, the inequality $f^L_{n,j}(\mathbf{x}) \leq f_{n,j}(\mathbf{x}) \leq f^U_{n,j}(\mathbf{x})$ holds true, where*

$$f^U_{n,j}(\mathbf{x}) = [\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1} [\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j}$$

$$f^L_{n,j}(\mathbf{x}) = [\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1} [\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j}$$

$$\mathbf{\Lambda}^{(k-1)} = \begin{cases} \mathbf{W}^{(m)} & \text{if } k = m+1; \\ \mathbf{W}^{(k-1)}(\hat{\lambda}^{(k-1)} \odot \mathbf{\Lambda}^{(k)}) & \text{if } k \in [2, m]. \end{cases}$$

$$\mathbf{\Omega}^{(k-1)} = \begin{cases} \mathbf{W}^{(m)} & \text{if } k = m+1; \\ \mathbf{W}^{(k-1)}(\hat{\omega}^{(k-1)} \odot \mathbf{\Omega}^{(k)}) & \text{if } k \in [2, m]. \end{cases}$$

$$\tilde{\mathbf{J}}^{(k-1)} = \begin{cases} \tilde{D}^{-1}\tilde{A} & \text{if } k = m+1; \\ (\tilde{D}^{-1}\tilde{A})\tilde{\mathbf{J}}^{k} & \text{if } k \in [2, m]. \end{cases}$$

and $\forall i \in [n_k]$,

$$\lambda^{(k-1)}_{n,i,j} = \begin{cases} \alpha^{(k-1)}_{U,n,i} & \text{if } \mathbf{\Lambda}^{(k)}_{i,j} \geq 0; \\ \alpha^{(k-1)}_{L,n,i} & \text{if } \mathbf{\Lambda}^{(k)}_{i,j} < 0. \end{cases} \qquad \omega^{(k-1)}_{n,i,j} = \begin{cases} \alpha^{(k-1)}_{L,n,i} & \text{if } \mathbf{\Omega}^{(k)}_{i,j} \geq 0; \\ \alpha^{(k-1)}_{U,n,i} & \text{if } \mathbf{\Omega}^{(k)}_{i,j} < 0. \end{cases}$$

$$\mathbf{\Delta}^{(k-1)}_{n,i,j} = \begin{cases} \beta^{(k-1)}_{U,n,i} & \text{if } \mathbf{\Lambda}^{(k)}_{i,j} \geq 0; \\ \beta^{(k-1)}_{L,n,i} & \text{if } \mathbf{\Lambda}^{(k)}_{i,j} < 0. \end{cases} \qquad \mathbf{\Theta}^{(k-1)}_{n,i,j} = \begin{cases} \beta^{(k-1)}_{L,n,i} & \text{if } \mathbf{\Omega}^{(k)}_{i,j} \geq 0; \\ \beta^{(k-1)}_{U,n,i} & \text{if } \mathbf{\Omega}^{(k)}_{i,j} < 0. \end{cases}$$

$$\hat{\lambda}^{(m-1)}_{i,j} = \begin{cases} \max(\alpha^{(k-1)}_{U,:,i}) & \text{if } \mathbf{\Lambda}^{(k)}_{i,j} \geq 0; \\ \min(\alpha^{(k-1)}_{L,:,i}) & \text{if } \mathbf{\Lambda}^{(k)}_{i,j} < 0; \end{cases} \qquad \hat{\omega}^{(k-1)}_{i,j} = \begin{cases} \min(\alpha^{(k-1)}_{L,:,i}) & \text{if } \mathbf{\Omega}^{(k)}_{i,j} \geq 0; \\ \max(\alpha^{(k-1)}_{U,:,i}) & \text{if } \mathbf{\Omega}^{(k)}_{i,j} < 0; \end{cases}$$

Corollary 5.3 (GCN closed-form global bounds) *Given a data point $\mathbf{x_0} \in \mathbb{R}^{n_0}$, ℓ_p ball parameters $p \geq 1$ and $\epsilon > 0$. For an m-layer GCN function $f : \mathbb{R}^{N \times C_0} \to \mathbb{R}^{N \times C_m}$, there exist two fixed values $\gamma^L_{n,j}$ and $\gamma^U_{n,j}$ such that $\forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ and $\forall j \in [C_m]$, $1/q = 1 - 1/p$, the inequality $\gamma^L_{n,j} \leq f_{n,j}(\mathbf{x}) \leq \gamma^U_{n,j}$ holds true, and*

$$\gamma^U_{n,j} = \epsilon \|\mathbf{\Lambda}^{(1)}_{:,j}\|_q + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1} [\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j}$$

$$\gamma^L_{n,j} = -\epsilon\|\mathbf{\Omega}^{(1)}_{:,j}\|_q + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j}$$

Note that this definition, theorem, and corollary closely match those those provided by CROWN [8].

5.1 Proof of Theorem 5.2

Given an m-layer graph convolutional neural network function $f : \mathbb{R}^{N\times C_0} \to \mathbb{R}^{N\times C_m}$ with pre-activation bounds $\mathbf{l}^{(k)}$ and $\mathbf{u}^{(k)}$ for $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$ and $\forall k \in [m-1]$, let the pre-activation inputs for the $[n,i]$-th neuron at layer $m-1$ be $\mathbf{y}^{(m-1)}_{n,i} := [\tilde{D}^{-1}\tilde{A}\Phi_{m-2}(\mathbf{x})]_{n,:}\mathbf{W}^{(m-1)}_{:,i}$, where $\Phi_{m-1} = \sigma(\mathbf{y}^{(m-1)})$. The $[n,j]$-th output of the neural network is the following:

$$\begin{aligned} f_{n,j}(\mathbf{x}) &= \sum_{i=1}^{C_{m-1}}[\tilde{D}^{-1}\tilde{A}\Phi_{m-1}(\mathbf{x})]_{n,i}\mathbf{W}^{(m)}_{i,j} && (14)\\ &= \sum_{i=1}^{C_{m-1}}[\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j} \\ &= \sum_{\mathbf{W}^{(m)}_{i,j}\geq 0}[\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j} + \sum_{\mathbf{W}^{(m)}_{i,j}<0}[\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j} && (15)\end{aligned}$$

Assuming the activation function $\sigma(y)$ is bounded by two linear functions $h^{(m-1)}_U, h^{(m-1)}_L$ in Definition 5.1, we have

$$h^{(m-1)}_L(\mathbf{y}^{(m-1)}) \leq \sigma(\mathbf{y}^{(m-1)}) \leq h^{(m-1)}_U(\mathbf{y}^{(m-1)}).$$

Thus, if the associated weight $\mathbf{W}^{(m)}_{i,j}$ to the $[n,i]$-th neuron is non-negative, we have

$$[\tilde{D}^{-1}\tilde{A}h^{(m-1)}_L(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j} \leq [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j} \leq [\tilde{D}^{-1}\tilde{A}h^{(m-1)}_U(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j}; \quad (16)$$

otherwise, we have

$$[\tilde{D}^{-1}\tilde{A}h^{(m-1)}_U(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j} \leq [\tilde{D}^{-1}\tilde{A}\sigma(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j} \leq [\tilde{D}^{-1}\tilde{A}h^{(m-1)}_L(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}^{(m)}_{i,j}. \quad (17)$$

Note that the elements of $\tilde{D}$ and $\tilde{A}$ are all positive, and as such will not affect the sign of these values.

Upper bound. Let $f^{U,m-1}_{n,j}(\mathbf{x})$ be an upper bound of $f_{n,j}(\mathbf{x})$. To compute $f^{U,m-1}_{n,j}(\mathbf{x})$, (15), (16) and (17) are the key equations. Precisely, for the $\mathbf{W}^{(m)}_{i,j} \geq 0$ terms in (15), the upper bound is the right-hand-side (RHS) in (16); and for the $\mathbf{W}^{(m)}_{i,j} < 0$ terms in (15), the upper bound is the RHS in

(17). Thus, we obtain:

$$
\begin{aligned}
& f_{n,j}^{U,m-1}(\mathbf{x}) \\
&= \sum_{\mathbf{W}_{i,j}^{(m)} \geq 0} [\tilde{D}^{-1}\tilde{A}h_U^{(m-1)}(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} + \sum_{\mathbf{W}_{j,i}^{(m)} < 0} [\tilde{D}^{-1}\tilde{A}h_L^{(m-1)}(\mathbf{y}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} && (18) \\
&= \sum_{\mathbf{W}_{i,j}^{(m)} \geq 0} [\tilde{D}^{-1}\tilde{A}\alpha_U^{(m-1)} \odot (\mathbf{y}^{(m-1)} + \beta_U^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} + \sum_{\mathbf{W}_{j,i}^{(m)} < 0} [\tilde{D}^{-1}\tilde{A}\alpha_L^{(m-1)} \odot (\mathbf{y}^{(m-1)} + \beta_L^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} && (19) \\
&= \sum_{i=1}^{C_{m-1}} [\tilde{D}^{-1}\tilde{A}\lambda_{:,:,j}^{(m-1)} \odot (\mathbf{y}^{(m-1)} + \mathbf{\Delta}_{:,:,j}^{(m-1)})]_{n,i}\mathbf{W}_{i,j}^{(m)} && (20) \\
&= \sum_{i=1}^{C_{m-1}} \sum_{a=1}^{N} [\tilde{D}^{-1}\tilde{A}]_{n,a}\lambda_{a,i,j}^{(m-1)}(\mathbf{y}_{a,i}^{(m-1)} + \mathbf{\Delta}_{a,i,j}^{(m-1)})\mathbf{W}_{i,j}^{(m)} && (21) \\
&= \sum_{i=1}^{C_{m-1}} \sum_{a=1}^{N} [\tilde{D}^{-1}\tilde{A}]_{n,a}\lambda_{a,i,j}^{(m-1)}\mathbf{y}_{a,i}^{(m-1)}\mathbf{W}_{i,j}^{(m)} + \sum_{i=1}^{C_{m-1}} \sum_{a=1}^{N} [\tilde{D}^{-1}\tilde{A}]_{n,a}\lambda_{a,i,j}^{(m-1)}\mathbf{\Delta}_{a,i,j}^{(m-1)}\mathbf{W}_{i,j}^{(m)} && (22) \\
&= \sum_{i=1}^{C_{m-1}} \sum_{a=1}^{N} [\tilde{D}^{-1}\tilde{A}]_{n,a}\lambda_{a,i,j}^{(m-1)}[\tilde{D}^{-1}\tilde{A}\Phi_{m-2}(\mathbf{x})\mathbf{W}^{(m-1)} + \mathbf{b}^{(m-1)}]_{a,i}\mathbf{W}_{i,j}^{(m)} \\
&\quad + \left[[\tilde{D}^{-1}\tilde{A}(\lambda_{:,:,j}^{(m-1)} \odot \mathbf{\Delta}_{:,:,j}^{(m-1)})\mathbf{W}^{(m)}]_{n,j} \right], && (23) \\
&= \sum_{i=1}^{C_{m-1}} \sum_{a=1}^{N} [\tilde{D}^{-1}\tilde{A}]_{n,a}\lambda_{a,i,j}^{(m-1)} \left[[\tilde{D}^{-1}\tilde{A}]_{a,:}\Phi_{m-2}(\mathbf{x})\mathbf{W}_{:,i}^{(m-1)} \right] \mathbf{W}_{i,j}^{(m)} + \tilde{\mathbf{b}}_{n,j}^{(m-1)} && (24) \\
&= \sum_{i=1}^{C_{m-1}} \sum_{a=1}^{N} [\tilde{D}^{-1}\tilde{A}]_{n,a}\lambda_{a,i,j}^{(m-1)} \sum_{k=1}^{C_{m-2}} \sum_{b=1}^{N} \left[[\tilde{D}^{-1}\tilde{A}]_{a,b}\Phi_{m-2}(\mathbf{x})_{b,k}\mathbf{W}_{k,i}^{(m-1)} \right] \mathbf{W}_{i,j}^{(m)} + \tilde{\mathbf{b}}_{n,j}^{(m-1)} && (25) \\
&= \sum_{k=1}^{C_{m-2}} \sum_{b=1}^{N} \sum_{i=1}^{C_{m-1}} \sum_{a=1}^{N} [\tilde{D}^{-1}\tilde{A}]_{n,a}\lambda_{a,i,j}^{(m-1)}[\tilde{D}^{-1}\tilde{A}]_{a,b}\Phi_{m-2}(\mathbf{x})_{b,k}\mathbf{W}_{k,i}^{(m-1)}\mathbf{W}_{i,j}^{(m)} + \tilde{\mathbf{b}}_{n,j}^{(m-1)} && (26)
\end{aligned}
$$

From (18) to (19), we replace $h_U^{(m-1)}(\mathbf{y}^{(m-1)})$ and $h_L^{(m-1)}(\mathbf{y}^{(m-1)})$ by their definitions; from (19) to (20), we use variables $\lambda^{(m-1)}$ and $\mathbf{\Delta}^{(m-1)}$ to denote the slopes in front of $\mathbf{y}^{(m-1)}$ and the intercepts in the parentheses:

$$
\lambda_{n,i,j}^{(m-1)} = \begin{cases} \alpha_{U,n,i}^{(m-1)} & \text{if } \mathbf{W}_{i,j}^{(m)} \geq 0; \\ \alpha_{L,n,i}^{(m-1)} & \text{if } \mathbf{W}_{i,j}^{(m)} < 0; \end{cases} \tag{27}
$$

$$
\mathbf{\Delta}_{n,i,j}^{(m-1)} = \begin{cases} \beta_{U,n,i}^{(m-1)} & \text{if } \mathbf{W}_{i,j}^{(m)} \geq 0; \\ \beta_{L,n,i}^{(m-1)} & \text{if } \mathbf{W}_{i,j}^{(m)} < 0. \end{cases} \tag{28}
$$

For the rest of the equations, we are simply unrolling matrix multiplications by adding summations and rearranging variables. We absorb all terms not related to $\mathbf{x}$ into an equivalent bias term $\tilde{\mathbf{b}}_{n,j}^{(m-1)}$, replace $y^{(m-1)}$ with it's definition, and further unroll the expression by replacing matrix multiplies with summations. Then lastly we move the summations all to the left side via the distributive property.

Note that at this point, it becomes nearly impossible decompose these terms into something that resembles our original expression (which allows us to recursively unroll this expression with respect to $\mathbf{x}$). The summation across the a index is impossible to perform, as we have one term with a column index of a, but two terms with row indexes of a. (Similar logic can also be applied to the i summation.) We will resolve this issue by providing a further relaxation on the $\lambda^{(m-1)}$ term.

Let us define a new term, $\hat{\lambda}^{(m-1)}$:

$$\hat{\lambda}_{i,j}^{(m-1)} = \begin{cases} \max(\alpha_{U,:,i}^{(m-1)}) & \text{if } \mathbf{W}_{i,j}^{(m)} \geq 0; \\ \min(\alpha_{L,:,i}^{(m-1)}) & \text{if } \mathbf{W}_{i,j}^{(m)} < 0; \end{cases}$$

Essentially we are just taking the loosest non-linear relaxation along all nodes in the graph we are operating on, which is allowed because this will only loosen our upper bound. Using this definition and following from (26):

$$\leq \sum_{k=1}^{C_{m-2}} \sum_{b=1}^{N} \sum_{i=1}^{C_{m-1}} [\tilde{D}^{-1}\tilde{A}\tilde{D}^{-1}\tilde{A}]_{n,b} \hat{\lambda}_{i,j}^{(m-1)} \Phi_{m-2}(\mathbf{x})_{b,k} \mathbf{W}_{k,i}^{(m-1)} \mathbf{W}_{i,j}^{(m)} + \tilde{\mathbf{b}}_{n,j}^{(m-1)} \tag{29}$$

$$= \sum_{k=1}^{C_{m-2}} \sum_{i=1}^{C_{m-1}} [\tilde{D}^{-1}\tilde{A}\tilde{D}^{-1}\tilde{A}\Phi_{m-2}(\mathbf{x})]_{n,k} \mathbf{W}_{k,i}^{(m-1)} \hat{\lambda}_{i,j}^{(m-1)} \mathbf{W}_{i,j}^{(m)} + \tilde{\mathbf{b}}_{n,j}^{(m-1)} \tag{30}$$

$$= \sum_{k=1}^{C_{m-2}} [\tilde{D}^{-1}\tilde{A}\tilde{D}^{-1}\tilde{A}\Phi_{m-2}(\mathbf{x})]_{n,k} \mathbf{W}_{k,:}^{(m-1)} (\hat{\lambda}_{:,j}^{(m-1)} \odot \mathbf{W}_{:,j}^{(m)}) + \tilde{\mathbf{b}}_{n,j}^{(m-1)} \tag{31}$$

$$= \sum_{k=1}^{C_{m-2}} [\tilde{D}^{-1}\tilde{A}\tilde{D}^{-1}\tilde{A}\Phi_{m-2}(\mathbf{x})]_{n,k} [\mathbf{W}^{(m-1)} (\hat{\lambda}^{(m-1)} \odot \mathbf{W}^{(m)})]_{k,j} + \tilde{\mathbf{b}}_{n,j}^{(m-1)} \tag{32}$$

$$= \sum_{k=1}^{C_{m-2}} [\tilde{\mathbf{J}}^{(m-1)}\Phi_{m-2}(\mathbf{x})]_{n,k} [\mathbf{\Lambda}^{(m-1)}]_{k,j} + \tilde{\mathbf{b}}_{n,j}^{(m-1)} \tag{33}$$

Where we define:

$$\begin{aligned} \mathbf{\Lambda}_{k,j}^{(m-1)} &= [\mathbf{W}^{(m-1)} (\hat{\lambda}^{(m-1)} \odot \mathbf{W}^{(m)})]_{k,j} \\ \tilde{\mathbf{J}}^{(m-1)} &= \tilde{D}^{-1}\tilde{A}\tilde{D}^{-1}\tilde{A} \\ \tilde{\mathbf{b}}_{n,j}^{(m-1)} &= [\tilde{D}^{-1}\tilde{A}(\lambda_{:,:,j}^{(m-1)} \odot \mathbf{\Delta}_{:,:,j}^{(m-1)})\mathbf{W}^{(m)} + \mathbf{b}^{(m)}]_{n,j} \end{aligned}$$

Notice now that we've combined the terms to the left and right of $\Phi_{m-2}(\mathbf{x})$ into $\tilde{\mathbf{J}}^{m-1}$ and $\mathbf{\Lambda}^{(m-1)}$ respectively, $f_{n,j}^{U,m-1}(\mathbf{x})$ in (18) and $f_{n,j}(\mathbf{x})$ in (14) are in the same form. (With the exception of the bias term, but the bias is simply a constant that can be absorbed into the following iteration's equivalent bias.) Thus, we can repeat the above procedure again to obtain an upper bound of $f_{n,j}^{U,m-1}(\mathbf{x})$, i.e. $f_{n,j}^{U,m-2}(\mathbf{x})$:

$$\begin{aligned} \mathbf{\Lambda}^{(m-2)} &= \mathbf{W}^{(m-2)} (\hat{\lambda}^{(m-2)} \odot \mathbf{\Lambda}^{(m-1)}) \\ \tilde{\mathbf{J}}^{(m-2)} &= (\tilde{D}^{-1}\tilde{A})\tilde{\mathbf{J}}^{(m-1)} \\ \tilde{\mathbf{b}}^{(m-2)} &= \tilde{\mathbf{J}}^{m-1} (\lambda_{:,:,j}^{(m-2)} \odot \mathbf{\Delta}_{:,:,j}^{(m-2)}) \mathbf{\Lambda}^{(m-1)} + \tilde{\mathbf{b}}^{(m-1)} \end{aligned}$$

$$\lambda_{n,i,j}^{(m-2)} = \begin{cases} \alpha_{U,n,i}^{(m-2)} & \text{if } \mathbf{\Lambda}_{i,j}^{(m-1)} \geq 0; \\ \alpha_{L,n,i}^{(m-2)} & \text{if } \mathbf{\Lambda}_{i,j}^{(m-1)} < 0; \end{cases}$$

$$\mathbf{\Delta}_{n,i,j}^{(m-2)} = \begin{cases} \beta_{U,n,i}^{(m-2)} & \text{if } \mathbf{\Lambda}_{i,j}^{(m-1)} \geq 0; \\ \beta_{L,n,i}^{(m-2)} & \text{if } \mathbf{\Lambda}_{i,j}^{(m-1)} < 0. \end{cases}$$

$$\hat{\lambda}_{i,j}^{(m-2)} = \begin{cases} \max(\alpha_{U,:,i}^{(m-2)}) & \text{if } \mathbf{\Lambda}_{i,j}^{(m-1)} \geq 0; \\ \min(\alpha_{L,:,i}^{(m-2)}) & \text{if } \mathbf{\Lambda}_{i,j}^{(m-1)} < 0; \end{cases}$$

and repeat again iteratively until obtain the final upper bound $f_{n,j}^{U,1}(\mathbf{x})$, where $f_{n,j}(\mathbf{x}) \leq f_{n,j}^{U,m-1}(\mathbf{x}) \leq f_{n,j}^{U,m-2}(\mathbf{x}) \leq \ldots \leq f_{n,j}^{U,1}(\mathbf{x})$. We let $f_{n,j}^{U}(\mathbf{x})$ denote the final upper bound $f_{n,j}^{U,1}(\mathbf{x})$, and we have

$$f_{n,j}^{U}(\mathbf{x}) = [\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\lambda_{:,:,j}^{(k)} \odot \mathbf{\Delta}_{:,:,j}^{(k)})\mathbf{\Lambda}^{(k+1)}]_{n,j}$$

$$\mathbf{\Lambda}^{(k-1)} = \begin{cases} \mathbf{W}^{(m)} & \text{if } k = m+1; \\ \mathbf{W}^{(k-1)}(\hat{\lambda}^{(k-1)} \odot \mathbf{\Lambda}^{(k)}) & \text{if } k \in [2, m]. \end{cases}$$

$$\tilde{\mathbf{J}}^{(k-1)} = \begin{cases} \tilde{D}^{-1}\tilde{A} & \text{if } k = m+1; \\ (\tilde{D}^{-1}\tilde{A})\tilde{\mathbf{J}}^{(k)} & \text{if } k \in [2, m]. \end{cases}$$

and $\forall i \in [n_k]$,

$$\lambda_{n,i,j}^{(k-1)} = \begin{cases} \alpha_{U,n,i}^{(k-1)} & \text{if } \mathbf{\Lambda}_{i,j}^{(k)} \geq 0; \\ \alpha_{L,n,i}^{(k-1)} & \text{if } \mathbf{\Lambda}_{i,j}^{(k)} < 0. \end{cases} \qquad \mathbf{\Delta}_{n,i,j}^{(k-1)} = \begin{cases} \beta_{U,n,i}^{(k-1)} & \text{if } \mathbf{\Lambda}_{i,j}^{(k)} \geq 0; \\ \beta_{L,n,i}^{(k-1)} & \text{if } \mathbf{\Lambda}_{i,j}^{(k)} < 0. \end{cases}$$

$$\hat{\lambda}_{i,j}^{(k-1)} = \begin{cases} \max(\alpha_{U,:,i}^{(k-1)}) & \text{if } \mathbf{\Lambda}_{i,j}^{(k)} \geq 0; \\ \min(\alpha_{L,:,i}^{(k-1)}) & \text{if } \mathbf{\Lambda}_{i,j}^{(k)} < 0; \end{cases}$$

Lower bound. The above derivations of upper bound can be applied similarly to deriving lower bounds of $f_{n,j}(\mathbf{x})$, when the main difference is now we need to use the LHS of (16) and (17) (rather than RHS when deriving upper bound) to bound the two terms in (15). Similarly in (29) where we define $\hat{\lambda}^{(k)}$, we must take reverse the bounds we use depending on the weight's sign. Thus, following the same procedure in deriving the upper bounds, we can iteratively unwrap the activation functions and obtain a final lower bound $f_{n,j}^{L,1}(\mathbf{x})$, where $f_{n,j}(\mathbf{x}) \geq f_{n,j}^{L,m-1}(\mathbf{x}) \geq f_{n,j}^{L,m-2}(\mathbf{x}) \geq \ldots \geq f_{n,j}^{L,1}(\mathbf{x})$. Let $f_{n,j}^{L}(\mathbf{x}) = f_{n,j}^{L,1}(\mathbf{x})$, we have:

$$f_{n,j}^{L}(\mathbf{x}) = [\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega_{:,:,j}^{(k)} \odot \mathbf{\Theta}_{:,:,j}^{(k)})\mathbf{\Omega}^{(k+1)}]_{n,j}$$

$$\mathbf{\Omega}^{(k-1)} = \begin{cases} \mathbf{W}^{(m)} & \text{if } k = m+1; \\ \mathbf{W}^{(k-1)}(\hat{\omega}^{(k-1)} \odot \mathbf{\Omega}^{(k)}) & \text{if } k \in [2, m]. \end{cases}$$

$$\tilde{\mathbf{J}}^{(k-1)} = \begin{cases} \tilde{D}^{-1}\tilde{A} & \text{if } k = m+1; \\ (\tilde{D}^{-1}\tilde{A})\tilde{\mathbf{J}}^{(k)} & \text{if } k \in [2, m]. \end{cases}$$

and $\forall i \in [n_k]$,

$$\omega_{n,i,j}^{(k-1)} = \begin{cases} \alpha_{L,n,i}^{(k-1)} & \text{if } \mathbf{\Omega}_{i,j}^{(k)} \geq 0; \\ \alpha_{U,n,i}^{(k-1)} & \text{if } \mathbf{\Omega}_{i,j}^{(k)} < 0. \end{cases} \qquad \mathbf{\Theta}_{n,i,j}^{(k-1)} = \begin{cases} \beta_{L,n,i}^{(k-1)} & \text{if } \mathbf{\Omega}_{i,j}^{(k)} \geq 0; \\ \beta_{U,n,i}^{(k-1)} & \text{if } \mathbf{\Omega}_{i,j}^{(k)} < 0. \end{cases}$$

$$\hat{\omega}_{i,j}^{(k-1)} = \begin{cases} \min(\alpha_{L,:,i}^{(k-1)}) & \text{if } \mathbf{\Omega}_{i,j}^{(k)} \geq 0; \\ \max(\alpha_{U,:,i}^{(k-1)}) & \text{if } \mathbf{\Omega}_{i,j}^{(k)} < 0; \end{cases}$$

$\lambda_{j,i}^{(k)}$ and $\omega_{j,i}^{(k)}$ only differs in the conditions of selecting $\alpha_{U,i}^{(k)}$ or $\alpha_{L,i}^{(k)}$; similarly for $\mathbf{\Delta}_{i,j}^{(k)}$ and $\mathbf{\Theta}_{i,j}^{(k)}$.

5.2 Proof of Corollary 5.3

Definition 5.4 (Dual norm) *Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. The associated dual norm, denoted as $\|\cdot\|_*$, is defined as*

$$\|\mathbf{a}\|_* = \{\sup_{\mathbf{y}} \mathbf{a}^\top \mathbf{y} \mid \|\mathbf{y}\| \leq 1\}.$$

Global upper bound. Our goal is to find a *global* upper and lower bound for the m-th layer GCN output $f_{n,j}(\mathbf{x}), \forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$. By Theorem 5.2, for $\mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$, we have $f^L_{n,j}(\mathbf{x}) \leq f_{n,j}(\mathbf{x}) \leq f^U_{n,j}(\mathbf{x})$ and $f^U_{n,j}(\mathbf{x}) = [\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j}$. Thus define $\gamma^U_{n,j} := \max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f^U_{n,j}(\mathbf{x})$, and we have

$$f_{n,j}(\mathbf{x}) \leq f^U_{n,j}(\mathbf{x}) \leq \max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f^U_{n,j}(\mathbf{x}) = \gamma^U_{n,j},$$

since $\forall \mathbf{x} \in \mathbb{B}_p(\mathbf{x_0}, \epsilon)$. In particular,

$$\begin{aligned}
\max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f^U_{n,j}(\mathbf{x}) &= \max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} \left[[\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j} \right] \\
&= \left[\max_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} [\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Lambda}^{(1)}]_{n,j} \right] + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j} \quad (34) \\
&= \epsilon \left[\max_{\mathbf{y}\in\mathbb{B}_p(\mathbf{0},1)} [\tilde{\mathbf{J}}^{(1)}\mathbf{y}\mathbf{\Lambda}^{(1)}]_{n,j} \right] + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j} \quad (35) \\
&= \epsilon \left[\max_{\mathbf{y}'\in\mathbb{B}_p(\mathbf{0},1)} \mathbf{y}'_{n,:}\mathbf{\Lambda}^{(1)}_{:,j} \right] + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j} \quad (36) \\
&= \epsilon\|\mathbf{\Lambda}^{(1)}_{:,j}\|_q + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j}. \quad (37)
\end{aligned}$$

From (34) to (35), let $\mathbf{y} := \frac{\mathbf{x}-\mathbf{x_0}}{\epsilon}$, and thus $\mathbf{y} \in \mathbb{B}_p(\mathbf{0}, 1)$. From (35) to (36), we utilize the fact that $\tilde{\mathbf{J}}$ is a product of row and column normalized matrices. As such, when multiplied with $\mathbf{y}$, it is essentially taking a weighted average of multiple p-norm vectors, and as such the result will never exceed the $\mathbb{B}_p(\mathbf{0}, 1)$ ball. From (36) to (37), apply Definition 5.4 and use the fact that ℓ_q norm is dual of ℓ_p norm for $p, q \in [1, \infty]$.

Global lower bound. Similarly, let $\gamma^L_{n,j} := \min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f^L_{n,j}(\mathbf{x})$, we have

$$f_{n,j}(\mathbf{x}) \geq f^L_{n,j}(\mathbf{x}) \geq \min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f^L_{n,j}(\mathbf{x}) = \gamma^L_{n,j}.$$

Since $f^L_{n,j}(\mathbf{x}) = [\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j}$, we can derive $\gamma^L_{n,j}$ (similar to the derivation of $\gamma^U_{n,j}$) below:

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} f^L_{n,j}(\mathbf{x}) &= \min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} \left[[\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j} \right] \\
&= \left[\min_{\mathbf{x}\in\mathbb{B}_p(\mathbf{x_0},\epsilon)} [\tilde{\mathbf{J}}^{(1)}\mathbf{x}\mathbf{\Omega}^{(1)}]_{n,j} \right] + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j} \\
&= -\epsilon \left[\max_{\mathbf{y}\in\mathbb{B}_p(0,1)} -[\tilde{\mathbf{J}}^{(1)}\mathbf{y}\mathbf{\Omega}^{(1)}]_{n,j} \right] + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j} \\
&= -\epsilon \left[\max_{\mathbf{y}'\in\mathbb{B}_p(0,1)} -\mathbf{y}'_{n,:}\mathbf{\Omega}^{(1)}_{:,j} \right] + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j} \\
&= -\epsilon\|\mathbf{\Omega}^{(1)}_{:,j}\|_q + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j}.
\end{aligned}$$

Thus, we have

$$\text{(global upper bound)} \quad \gamma^U_{n,j} = \epsilon\|\mathbf{\Lambda}^{(1)}_{:,j}\|_q + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Lambda}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\lambda^{(k)}_{:,:,j} \odot \mathbf{\Delta}^{(k)}_{:,:,j})\mathbf{\Lambda}^{(k+1)}]_{n,j}$$

$$\text{(global lower bound)} \quad \gamma^L_{n,j} = -\epsilon\|\mathbf{\Omega}^{(1)}_{:,j}\|_q + [\tilde{\mathbf{J}}^{(1)}\mathbf{x_0}\mathbf{\Omega}^{(1)}]_{n,j} + \sum_{k=1}^{m-1}[\tilde{\mathbf{J}}^{(k+1)}(\omega^{(k)}_{:,:,j} \odot \mathbf{\Theta}^{(k)}_{:,:,j})\mathbf{\Omega}^{(k+1)}]_{n,j}$$

6 Conclusion

We propose methods with which to provide perturbation bounds for graph convolutional networks, and developed code to test more advanced robustness method on training models.

This work will continue into future quarters. Firstly, we wish to train models using the code developed. We want to run experiments such as training to minimize CROWN-produced bounds, making the non-linear bound's slope a differentiable parameter to be altered via training, and implement fast-lip [7] to see if minimzing the local-lipshitz constant also helps with improving robustness. Furthermore, we would like to implement the GCN algorithm derived in this paper, and evaluate it's effectivness on improving GCN robustness.

References

[1] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[2] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[3] J. Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017.

[4] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10802–10813. Curran Associates, Inc., 2018.

[5] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL):41:1–41:30, January 2019.

[6] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. *CoRR*, abs/1809.08098, 2018.

[7] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018.

[8] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *CoRR*, abs/1811.00866, 2018.