Improved Certified Adversarial Lower Bound Using Adaptive Relaxations

Akash Kumar
Department of Computer Science
Aalto University
Otaniemi, Espoo-603103, Finland
akash.kumar@aalto.fi

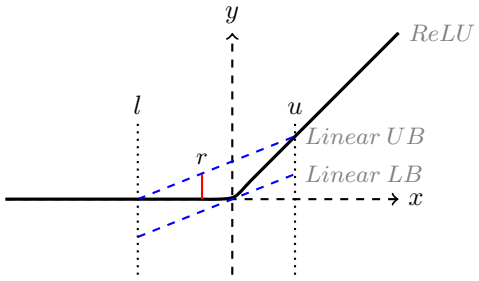
Abstract

In this work, we are trying to improve upon the works of Wong et al. [2] and Hsieh et al. [3]. They tried in their works to get a certified lower bound on the minimal adversarial bound using convex relaxations for ReLU networks. For a fixed normed ϵ-ball for an input instance x i.e. $\mathcal{B}(x, \epsilon)$, they strive to find linear approximations for ReLU activation functions. This convex relaxation is used to upper and lower bound the components of the logit(final) layer. Although the linear relaxation thus used, helps in finding certified lower bound on the minimal adversarial bounds efficiently, it is at the cost of large error in approximating the ReLU function. Our work addresses this issue and builds on the work based on convex relaxations. To minimize the error in approximating the ReLU activations, we use regulated softplus function which we call ReST. Our approximation is based on adaptive selection of ReST, Linear and Quadratic bounds for ReLU activations in such a manner that we obtain concave upper bound and convex lower bound for the component functions of the logit layer. Thus, for a targeted attack, we obtain a concave objective to be maximized till a threshold.

1 Relevant Previous Work

Recently, we have seen CROWN and how it gives certified lower bound on the minimal adversarial perturbation. Long story short, for the search space $B(x_0, \epsilon)$, they calculate the bounds on the activation functions (in particular ReLU) using linear approximations as shown below.
But you can notice that, this approximation, although harnessing the linear property of it, looses so much in terms of error. So the final approximated functions for each component function of the logit layer, thus calculated, will be loose as well.

$$f_i^L(x) \leq f_i \leq f_i^U(x) \tag{1}$$



Preprint.

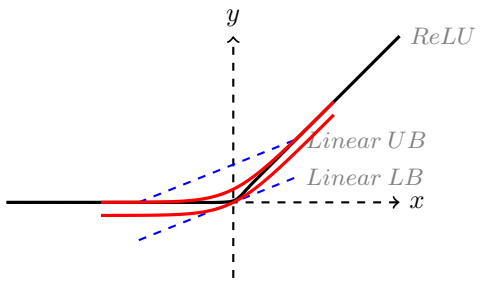
We notice that at the point r, the red vertical line denotes the error we make in approximating ReLU activation with a linear upper bound. Similarly, the same happens for linear lower bound.
This is where our work tries to minimize this error so that we get better bounds.

2 Regulated softplus : ReST

The idea to use regulated softplus (ReST) for ReLU has been there for a while but was first proposed in Hein et al [1] for robustness setting.

$$ReLU \longrightarrow \frac{1}{\alpha}\ln(1+e^{\alpha x}) \tag{2}$$

ReST has an interesting property that it bounds $ReLU$ and $\lim_{\alpha\to\infty}\ln(1+e^{\alpha x})$ = $ReLU$.



So, we make much lower error in approximating ReLU based on ReST.

3 General Framework

Notations. For an m-layer neural network with an input vector $\mathbf{x} \in \mathbb{R}^{n_0}$, let the number of neurons in each layer be n_k, $\forall\ k \in [m]$, where $[i]$ denotes set $\{1, 2, ..., i\}$. Let the k-th layer weight matrix be $\mathbf{W}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$ and bias vector be $\mathbf{b}^{(k)} \in \mathbb{R}^{n_k}$ and let $\Phi_k(\mathbf{x}) = \sigma(\mathbf{W}^{(k)}\Phi_{k-1}(\mathbf{x}) + \mathbf{b}^{(k)})$, $k \in [m-1]$, where $\sigma(.)$ is the coordinate-wise activation function. In this work we emphasize on ReLU activation function: $\sigma(y) = \max(y, 0)$. Note that the input $\Phi_0(\mathbf{x}) = \mathbf{x}$, and the vector output of the NN is $f(\mathbf{x}) = \mathbf{W}^{(m)}\Phi_{m-1}(\mathbf{x}) + \mathbf{b}^{(m)}$. The j-th output element is denoted as $f_j(\mathbf{x}) = [\Phi_m(\mathbf{x})]_j$.

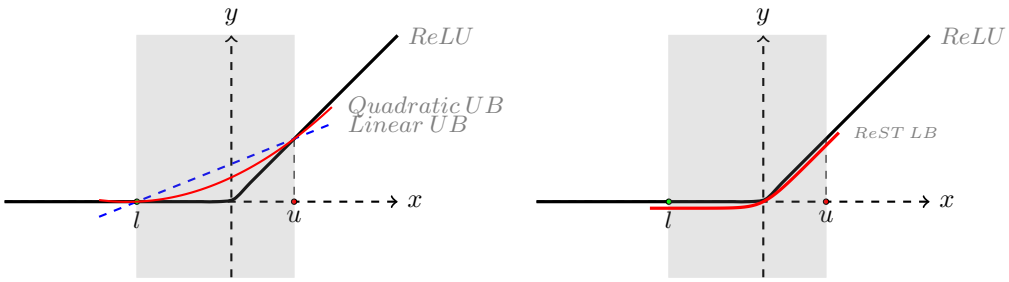
Input perturbation and pre-activation bounds. Let $\mathbf{x}_0 \in \mathbb{R}^{n_0}$ be a given data point, and let the perturbed input vector $\mathbf{x}$ be within an ϵ-bounded ℓ_p-ball centered at $\mathbf{x}_0$, i.e., $\mathbf{x} \in \mathbb{B}_p(\mathbf{x}_0, \epsilon)$, where $\mathbb{B}_p(\mathbf{x}_0, \epsilon) := \{\mathbf{x} |\ ||\mathbf{x} - \mathbf{x}_0||_p \leq \epsilon\}$. For the r-th neuron in the k-th layer, let its *pre-activation* input be $y_r^{(k)}$, where $y_r^{(k)} = \mathbf{W}_{r,:}^{(k)}\Phi_{k-1}(\mathbf{x}) + \mathbf{b}_r^{(k)}$ and $\mathbf{W}_{r,:}^{(k)}$ denotes the r-th row of matrix $\mathbf{W}^{(k)}$. When $\mathbf{x}_0$ is perturbed within a ϵ-bounded ℓ_p-ball, let $\mathbf{l}_r^{(k)}$, $\mathbf{u}_r^{(k)} \in \mathbb{R}$ be the pre-activation lower bound and upper bound of $y_r^{(k)}$, i.e. $\mathbf{l}_r^{(k)} \leq y_r^{(k)} \leq \mathbf{u}_r^{(k)}$.

4 Bounding ReLU

There are several ways to bound ReLU so that the output f can be bounded. In this section, we would demonstrate the way we are bounding.

4.1 Adaptive quadratic and linear upper bound and ReSt lower bound

Say for the r-th neuron in the k-th layer, we have obtained the corresponding pre-activation lower bound and upper bound of $y_r^{(k)}$, then as shown in figure below we can linearly and quadratically approximate the ReLU by above. Given that we can control the convexity of ReSt, we bound the ReLU by a ReST from below as shown.



Theorem 4.1 (Concave Upper Bound, Convex Lower Bound)**.** *For an m-layer neural network, denoted by $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$, there exists two explicit functions $f_i^L : \mathbb{R}^{n_0} \to \mathbb{R}$ and $f_i^U : \mathbb{R}^{n_0} \to \mathbb{R}$ such that $\forall i \in [n_m]$, $\forall \boldsymbol{x} \in \mathbb{B}_p(\boldsymbol{x}_0, \epsilon)$ the following inequality holds:*

$$f_i^L(\boldsymbol{x}) \leq f_i(\boldsymbol{x}) \leq f_i^U(\boldsymbol{x}) \tag{3}$$

where $f_i^L(\boldsymbol{x})$ and $f_i^U(\boldsymbol{x})$ are convex and concave functions respectively.

By adaptively choosing linear or "slow-growing"-quadratic upper bound and ReST lower bound,

Definition 4.1 (Slow-growing quadratic function)**.** Assume $g : \mathbb{R}^{n_0} \to \mathbb{R}$ to be a concave function, then we say a quadratic function $Q_g(\mathbf{x}) = a\mathbf{x}^2 + b\mathbf{x} + c$ to be *slow-growing* w.r.t to g if $a > 0$ and $Q_g \circ g$ remains concave.

Lemma 4.2 (Existence of a slow-growing function)**.**

Theorem 4.3. *Let f be convex and β-smooth on $\mathbb{B}_p(x_0, \epsilon)$ then gradient descent with $\eta = \frac{1}{\beta}$ satisfies*

$$f(x_t) - f(x^*) \leq \frac{2\beta\epsilon^2}{t-1} \tag{4}$$

5 Bounds

In this section, we'll prove Theorem 4.1 and give the bounds for the activation functions and the m-layer neural network. We approximate the neural network in a forward manner by evaluating the bounds of the neurons of the intial layers unlike CROWN which does it recursively from the final layer itself.
We'll show that for each layer k, we obtain the following bounds on the pre-activations of the k^{th} layer.

$$f_{L,i}^{(k)}(\mathbf{x}) \leq y_i^{(k)} \leq f_{U,i}^{(k)}(\mathbf{x}) \tag{5}$$

Upper Bound. For the m-layer network $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$, j^{th} component f_j can be written using the post-activations of the $(m-1)$-layer as follows:

$$\begin{aligned}
f_j(\mathbf{x}) &= W_{j,:}^{(m)} \Phi_{m-1}(\mathbf{x}) + b_j^m \\
&= \sum_{i=1}^{n_{m-1}} W_{j,i}^{(m)} [\Phi_{m-1}(\mathbf{x})]_i + b_j^m \\
&= \sum_{W_{j,i}^{(m)} \geq 0} W_{j,i}^{(m)} [\Phi_{m-1}(\mathbf{x})]_i + \sum_{W_{j,i}^{(m)} < 0} W_{j,i}^{(m)} [\Phi_{m-1}(\mathbf{x})]_i + b_j^m \\
&= \underbrace{\sum_{W_{j,i}^{(m)} \geq 0} W_{j,i}^{(m)} \sigma(\mathbf{y}_i^{(m-1)})}_{\text{positive weights}} + \underbrace{\sum_{W_{j,i}^{(m)} < 0} W_{j,i}^{(m)} \sigma(\mathbf{y}_i^{(m-1)})}_{\text{negative weights}} + b_j^m \\
&\leq \underbrace{\sum_{W_{j,i}^{(m)} \geq 0} W_{j,i}^{(m)} h_{U,i}^{(m-1)}(f_{U,i}^{(m-1)}(\mathbf{x}))}_{\text{product of concave functions}} + \underbrace{\sum_{W_{j,i}^{(m)} < 0} W_{j,i}^{(m)} h_{L,i}^{(m-1)}(f_{U,i}^{(m-1)}(\mathbf{x}))}_{\text{negative product of convex functions}} + b_j^m
\end{aligned} \tag{6}$$

If $\forall\ i$, $f_{U,i}^{(k)}(\mathbf{x})i$ and $f_{L,i}^{(k)}(\mathbf{x})$ are concave and convex functions respectively then $h_{U,i}^{(m-1)}(f_{U,i}^{(k)}(\mathbf{x}))$ and $h_{L,i}^{(m-1)}(f_{L,i}^{(k)}(\mathbf{x}))$ are concave and convex as well. That follows because $\forall i, m \quad h_{U,i}^{(m-1)}$ and $h_{L,i}^{(m-1)}$ are concave and convex functions respectively. So, notice that sum in the last inequality is a concave function.

We define the upper bound on $f_j(\mathbf{x})$ by :

$$f_{U,j} = \sum_{W_{j,i}^{(m)} \geq 0} W_{j,i}^{(m)} h_{U,i}^{(m-1)}(f_{U,i}^{(m-1)}(\mathbf{x})) + \sum_{W_{j,i}^{(m)} < 0} W_{j,i}^{(m)} h_{L,i}^{(m-1)}(f_{L,i}^{(m-1)}(\mathbf{x})) + b_j^m \tag{7}$$

Thus, we show that $f_{U,j}$ is a concave function.

Lower Bound. Similar to equation (6), we obtain a lower bound $f_{L,j}$ for f_j which is convex and has the following form.

$$f_{L,j} = \sum_{W_{j,i}^{(m)} \geq 0} W_{j,i}^{(m)} h_{L,i}^{(m-1)}(f_{L,i}^{(m-1)}(\mathbf{x})) + \sum_{W_{j,i}^{(m)} < 0} W_{j,i}^{(m)} h_{U,i}^{(m-1)}(f_{U,i}^{(m-1)}(\mathbf{x})) + b_j^m \tag{8}$$

Note that $f_{L,i}^{(m-1)}(\mathbf{x})$ and $f_{U,i}^{(m-1)}(\mathbf{x})$ can be recursively unfolded in terms of $f_{L,:}^{(m-2)}(\mathbf{x})$ and $f_{U,:}^{(m-2)}(\mathbf{x})$.

Lemma 5.1. For each layer k and neuron i, $f_{L,i}^{(k)}(\mathbf{x})$ is convex and $f_{U,i}^{(k)}(\mathbf{x})$ is concave.

Proof. We use induction on the number of layers k. For base case (k = 1), it is trivial as by definition, $f_{U,i}^{(1)}(\mathbf{x}) = f_{U,i}^{(1)}(\mathbf{x}) = W_{i,:}^{(1)}\mathbf{x} + b_i^{(1)}$ are linear affine functions and thus are both concave and convex. Now, if we assume the lemma for $k = r$, then for $k = r + 1$ the corresponding terms $f_{U,i}^{(r+1)}(\mathbf{x})$ and $f_{L,i}^{(r+1)}(\mathbf{x})$ can be written in terms of $f_{L,:}^{(r)}(\mathbf{x})$ and $f_{U,:}^{(r)}(\mathbf{x})$ as in (7) and (8) and thus the lemma follows. □

6 Algorithms

In this section, we'll talk about the algorithm we use to calculate the bounding functions and respective bounds of each neuron.

Observation. We note that for any layer k, the weights matrix $W^{(k)}$ can be written as the sum of two matrices $W_{\text{positive}}^{(k)}$ and $W_{\text{negative}}^{(k)}$ as

$$W^{(k)} = W_{\text{positive}}^{(k)} + W_{\text{negative}}^{(k)}\ , \text{ where}$$

$$W_{\text{positive},ij}^{(k)} = \begin{cases} W_{ij}^{(k)}, & \text{if } W_{ij}^{(k)} > 0 \\ 0, & \text{otherwise} \end{cases},\ W_{\text{negative}}^{(k)} = \begin{cases} W_{ij}^{(k)}, & \text{if } W_{ij}^{(k)} < 0 \\ 0, & \text{otherwise} \end{cases}$$

Now, we can re-write (7) and (8) as

$$f_{U,j} = W_{\text{positive}}^{(m)} \bar{h}^{(m-1)}(f_U^{(m-1)}(\mathbf{x})) + W_{\text{negative}}^{(k)} \bar{h}^{(m-1)}(f_L^{(m-1)}(\mathbf{x})) + b^{(m)} \tag{9}$$

$$f_{L,j} = W_{\text{positive}}^{(m)} \bar{h}^{(m-1)}(f_L^{(m-1)}(\mathbf{x})) + W_{\text{negative}}^{(k)} \bar{h}^{(m-1)}(f_U^{(m-1)}(\mathbf{x})) + b^{(m)} \tag{10}$$

Gradient Descent approximations for bounding functions. For the targeted convex minimization problem with the objective $(f_{L,l}(\mathbf{x}) - f_{U,k}(\mathbf{x}))$ where l and k are predicted and target labels, we use gradient descent to find the minimum.

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{x}_t - \eta \boldsymbol{\nabla}(f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)) \\ \mathbf{y} &= \boldsymbol{\nabla}^\top (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)).(\mathbf{x} - \mathbf{x}_t) + (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)) \end{aligned} \tag{11}$$

Notice that at the t^{th} step, the hyperplane defined by second equation in (11) lower bounds $(f_{L,l}(\mathbf{x}) - f_{U,k}(\mathbf{x}))$ by the implication of separating hyperplane theorem. Now,

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{B}_p(x_0,\epsilon)} \mathbf{y} &= \min_{\mathbf{x}\in\mathbb{B}_p(x_0,\epsilon)} \boldsymbol{\nabla}^\top (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)).(\mathbf{x} - \mathbf{x}_t) + (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)) \\
&= \min_{\mathbf{r}\in\mathbb{B}_p(0,\epsilon)} \boldsymbol{\nabla}^\top (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)).(\mathbf{r} + \mathbf{x}_0 - \mathbf{x}_t) + (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)) \\
&= \min_{\mathbf{r}\in\mathbb{B}_p(0,\epsilon)} \boldsymbol{\nabla}^\top (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)).\mathbf{r} + \left[\boldsymbol{\nabla}^\top (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)).(\mathbf{x}_0 - \mathbf{x}_t) + (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t))\right] \\
&= ||\boldsymbol{\nabla}(f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t))||_q.\epsilon + \left[\boldsymbol{\nabla}^\top (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t)).(\mathbf{x}_0 - \mathbf{x}_t) + (f_{L,l}(\mathbf{x}_t) - f_{U,k}(\mathbf{x}_t))\right]
\end{aligned}$$

7 Suboptimality

In this section, we'll show the sub-optimality of the current algorithm. In comparison to CROWN, we would show that the bounds obtained are slightly weaker.

We would work on a 3-layered network with the notations similar to CROWN. Let us denote the $\lambda_j^{(k)}$ (sign of $W_{c,i}^{(3)}$ for all i) for the bounds (upper or lower) for the j^{th} neuron in the k^{th} layer. Thus from (7), we can write for some class c:

$$\begin{aligned}
f_{U,c} &= \sum_{W_{c,i}^{(3)}\geq 0} W_{c,i}^{(3)} h_{U,i}^{(2)}(f_{U,i}^{(2)}(\mathbf{x})) + \sum_{W_{j,i}^{(3)}<0} W_{c,i}^{(3)} h_{L,i}^{(2)}(f_{L,i}^{(2)}(\mathbf{x})) + b_c^{(3)} \\
&= \sum_i W_{c,i}^{(3)} (\alpha_i^{(2),\lambda_i^{(2)}} W_i^{(2)} \bar{\Phi}^{(1)}(x) + \beta_i^{(2),\lambda_i^{(2)}}) + b_c^{(3)} \\
&= \sum_i W_{c,i}^{(3)} (\alpha_i^{(2),\lambda_i^{(2)}} \sum_j W_{ij}^{(2)} \bar{\Phi}_j^{(1)}(x) + \beta_i^{(2),\lambda_i^{(2)}}) + b_c^{(3)} \\
&= \sum_j (\sum_i W_{c,i}^{(3)} \alpha_i^{(2),\lambda_i^{(2)}} W_{ij}^{(2)}).\bar{\Phi}_j^{(1)}(x) + C_0 + b_c^{(3)}
\end{aligned} \tag{12}$$

Now, according to the ReSt approach, we write the same bound as follows (we denote the logit layer upper bound as $\bar{f}_{U,c}$ in the case ReSt approach):

$$\bar{f}_{U,c} = \sum_i \sum_j W_{c,i}^{(3)} \alpha_i^{(2),\lambda_i^{(2)}} W_{ij}^{(2)}.\bar{\Phi}_j^{(1),sgn(W_{ij}^{(2)}),\lambda_j^{(2)}}(x) + C_0 + b_c^{(3)} \tag{13}$$

In the case of ReSt, there is a difference in how we pick the bound on the activation functions which depends on the weights of the immediate next layer, where in CROWN it is dependent on the "cumulative" weights as shown. In (13), we note that upper or lower bound on $\bar{\Phi}_j^{(1)}$ depends on sign of $W_{ij}^{(2)}$ and $\lambda_j^{(2)}$ which is shown in the table below.

$W_{c,i}^{(3)}$	$W_{ij}^{(2)}$	Bound on $\bar{\Phi}_j^{(1),sgn(W_{ij}^{(2)}),\lambda_j^{(2)}}(x)$
+	+	U
+	-	L
-	+	L
-	-	U

Table 1: Bounds of Activations for ReST

Now, we can compare (12) and (13) as follows. Assume that for some arbitrary j, $\bar{\Phi}_j^{(1)}$ is positive (or similarly can be shown for negative value) as determined by CROWN. Then, we can shown that each product term in (12) i.e. $W_{c,i}^{(3)} \alpha_i^{(2),\lambda_i^{(2)}} W_{ij}^{(2)}.\bar{\Phi}_j^{(1)}(x)$ is smaller (greater in negative case) than corresponding term in (13) i.e. $W_{c,i}^{(3)} \alpha_i^{(2),\lambda_i^{(2)}} W_{ij}^{(2)}.\bar{\Phi}_j^{(1),sgn(W_{ij}^{(2)}),\lambda_j^{(2)}}(x)$.

$W_{c,i}^{(3)}$	$W_{ij}^{(2)}$	Bound on $\bar{\Phi}_j^{(1),sgn(W_{ij}^{(2)}),\lambda_j^{(2)}}(x)$	$W_{c,i}^{(3)}\alpha_i^{(2),\lambda_i^{(2)}}W_{ij}^{(2)}$	CROWN vs ReSt
+	+	U	+	Both positive (Equal)
+	-	L	-	Both neg.: CROWN*upper $<$ ReSt*lower
-	+	L	-	Both neg.: CROWN*upper $<$ ReSt*lower
-	-	U	+	Both positive (Equal)

Table 2: Comparison CROWN vs ReST

Since, we are computing the upper bound Table 2 shows that, CROWN computes a tighter upper bound than ReSt. Similarly, for lower bounds similar tables can be drawn and we find tighter lower bound using CROWN.

8 Non-Linear Softplus Approximations for CROWN Bounds

From CROWN, we can see that iteratively they find the bounds for the j^{th} component of the logit layer, f_j in the following form: $f_j(\mathbf{x}) \leq f_j^{U,m-1}(\mathbf{x}) \leq f_j^{U,m-2}(\mathbf{x}) \leq .... \leq f_j^{U,1}(\mathbf{x})$

Let us denote $\mathbf{y}_j^m$ as the j^{th} pre-activations for the m^{th}-layer (as computed by f or model). Now assume that $\bar{\mathbf{y}}_j^m$ and $\widetilde{\mathbf{y}}_j^m$ are the j^{th} pre-activations for the m^{th}-layer as computed by CROWN approximations and non-linear approximation which we will outline.
For convex optimization, we aim to obtain concave upper bound and convex lower bound. Thus, for every layer in a forward propagating manner, we would approximate functions compared to linear approximations such that those properties hold.

Based on CROWN,

$$\begin{aligned}
f_j^{U,2}(\mathbf{x}) = \sum_{r=1}^{n_1} \widetilde{W}_{j,r}^{(2)}[\Phi_1(\mathbf{x})]_r + \widetilde{b}_j^{(2)} &= \sum_{r=1}^{n_1} \widetilde{W}_{j,r}^{(2)}[\sigma^{\lambda_r^{(1)}}(\mathbf{y}_r^{(1)})]_r + \widetilde{b}_j^{(2)} \\
&= \sum_{r=1}^{n_1}\left(\sum_{k=1}^{n_2} \widetilde{W}_{j,k}^{(3)}\alpha_k^{\lambda_k^{(2)}}W_{k,r}^{(2)}\right)[\sigma^{\lambda_r^{(1)}}(\mathbf{y}_r^{(1)})]_r + \widetilde{b}_j^{(2)} \\
&= \sum_{k=1}^{n_2} \widetilde{W}_{j,k}^{(3)}\alpha_k^{\lambda_k^{(2)}}\sum_{r=1}^{n_1}\left(W_{k,r}^{(2)}[\sigma^{\lambda_r^{(1)}}(\mathbf{y}_r^{(1)})]_r\right) + \widetilde{b}_j^{(2)} \\
&= \sum_{k=1}^{n_2} \widetilde{W}_{j,k}^{(3)}\left(\alpha_k^{\lambda_k^{(2)}}\sum_{r=1}^{n_1} W_{k,r}^{(2)}[\sigma^{\lambda_r^{(1)}}(\mathbf{y}_r^{(1)})]_r + \beta_k^{\lambda_k^{(2)}}\right) + \widetilde{b}_j^{(3)}
\end{aligned} \tag{14}$$

Similarly, non-linear approximation can be written replacing $\sigma^{\lambda_r^{(1)}}$'s with $\widetilde{\sigma}^{\lambda_r^{(1)}}$ which is non-linear approximation. Thus, new non-linear upper bound $\widetilde{f}_j^{U,2}(\mathbf{x}) = \sum_{k=1}^{n_2} \widetilde{W}_{j,k}^{(3)}\left(\alpha_k^{\lambda_k^{(2)}}\sum_{r=1}^{n_1} W_{k,r}^{(2)}[\widetilde{\sigma}^{\lambda_r^{(1)}}(\mathbf{y}_r^{(1)})]_r + \beta_k^{\lambda_k^{(2)}}\right) + \widetilde{b}_j^{(3)}$. It is clear that $f_j^{U,3}(\mathbf{x}) \leq \widetilde{f}_j^{U,2}(\mathbf{x}) \leq f_j^{U,2}(\mathbf{x})$.

Notice that $\widetilde{W}_{j,r}^{(2)}$ fixes the direction (upper or lower) in which approximations would be applied in the 2nd layer. Thus, we can't say anything about the curvature of $\sum_{r=1}^{n_1} W_{k,r}^{(2)}[\sigma^{\lambda_r^{(1)}}(\mathbf{y}_r^{(1)})]_r$ unless we use non-linear approximations in a certain way.

8.1 Upper or Lower Boundedness

We noticed that, $f_j^{U,3}(\mathbf{x}) \leq \widetilde{f}_j^{U,2}(\mathbf{x}) \leq f_j^{U,2}(\mathbf{x})$ holds. Now, we have to work with $\sum_{r=1}^{n_1} W_{k,r}^{(2)}[\sigma^{\lambda_r^{(1)}}(\mathbf{y}_r^{(1)})]_r$ as input for the third layer for each k. But the input in case of $f_j^{U,3}(\mathbf{x})$ is $\sum_{r=1}^{n_1} W_{k,r}^{(2)}[\sigma(\mathbf{y}_r^{(1)})]_r$. Now it is not clear how the two sums $\sum_{r=1}^{n_1} W_{k,r}^{(2)}[\sigma(\mathbf{y}_r^{(1)})]_r$ and $\sum_{r=1}^{n_1} W_{k,r}^{(2)}[\sigma^{\lambda_r^{(1)}}(\mathbf{y}_r^{(1)})]_r$ are related (i.e. if one is bigger or smaller than the other). Thus, now if we apply the non-linear approximation again in the case of $f_j^{U,3}(\mathbf{x})$ and $\widetilde{f}_j^{U,2}(\mathbf{x})$, it becomes unclear to quantify how small $\widetilde{f}_j^{U,3}(\mathbf{x})$ is turning out to be.

References

[1] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 2266–2276. Curran Associates, Inc., 2017.

[2] Eric Wong and J. Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope, 2018.

[3] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 4939–4948. Curran Associates, Inc., 2018.