APPENDIX

A FOUR CASES OF NEURON INTERACTIONS

In Section 3, we discussed how to find neuron implications between two neurons. Since each ReLU neuron has two possible statuses (active or inactive), there are four kinds of possible implications. We list all four possibilities and their corresponding optimization formulation here:

1. An implicant neuron $z_{j_2}^{(i_2)}$ split to *inactive* case implies an improved *lower* bound of an implicated neuron $z_{j_1}^{(i_1)}$ (this is the case in (6) discussed in Section 3):

$$\begin{aligned} l^*_{\text{relaxed}} &:= \min_{\boldsymbol{x}} \underline{\mathbf{a}}^{(i_1,j_1)\top} \boldsymbol{x} + \underline{c}^{(i_1,j_1)} \\ \text{s.t. } \boldsymbol{x}_0 - \epsilon \le \boldsymbol{x} \le \boldsymbol{x}_0 + \epsilon; &\quad \underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} \le 0 \end{aligned} \tag{9}$$

2. An implicant neuron $z_{j_2}^{(i_2)}$ split to *active* case implies an improved *lower* bound of an implicated neuron $z_{j_1}^{(i_1)}$:

$$\begin{aligned} l^*_{\text{relaxed}} &:= \min_{\boldsymbol{x}} \underline{\mathbf{a}}^{(i_1,j_1)\top} \boldsymbol{x} + \underline{c}^{(i_1,j_1)} \\ \text{s.t. } \boldsymbol{x}_0 - \epsilon \le \boldsymbol{x} \le \boldsymbol{x}_0 + \epsilon; &\quad \overline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \overline{c}^{(i_2,j_2)} \ge 0 \end{aligned} \tag{10}$$

3. An implicant neuron $z_{j_2}^{(i_2)}$ split to *inactive* case implies an improved *upper* bound of an implicated neuron $z_{j_1}^{(i_1)}$:

$$\begin{aligned} l^*_{\text{relaxed}} &:= \max_{\boldsymbol{x}} \overline{\mathbf{a}}^{(i_1,j_1)\top} \boldsymbol{x} + \overline{c}^{(i_1,j_1)} \\ \text{s.t. } \boldsymbol{x}_0 - \epsilon \le \boldsymbol{x} \le \boldsymbol{x}_0 + \epsilon; &\quad \underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} \le 0 \end{aligned} \tag{11}$$

4. An implicant neuron $z_{j_2}^{(i_2)}$ split to *active* case implies an improved *upper* bound of an implicated neuron $z_{j_1}^{(i_1)}$:

$$\begin{aligned} l^*_{\text{relaxed}} &:= \max_{\boldsymbol{x}} \overline{\mathbf{a}}^{(i_1,j_1)\top} \boldsymbol{x} + \overline{c}^{(i_1,j_1)} \\ \text{s.t. } \boldsymbol{x}_0 - \epsilon \le \boldsymbol{x} \le \boldsymbol{x}_0 + \epsilon; &\quad \overline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \overline{c}^{(i_2,j_2)} \ge 0 \end{aligned} \tag{12}$$

Note that all four optimization problems have the same nature and can be solved using the same technique discussed in Section C.1. we will only use Eq. (6) as an example since all other three cases can be converted to Eq. (6). For instance, to solve Eq. (10), we can change the implicant constraint to $-\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} - \underline{c}^{(i_2,j_2)} \le 0$; to solve Eq. (11), we can change the max objective to $\min_{\boldsymbol{x}} -\underline{\mathbf{a}}^{(i_1,j_1)\top} \boldsymbol{x} - \underline{c}^{(i_1,j_1)}$.

B PROOFS

We now give the proofs for the two theorems in Section 3. Here Theorem 3.1 shows the possibility of solving a cheap linear programming problem to find bound implications, and Theorem 3.2 shows how we can reduce the number of optimization problems by filtering out some neurons that do not help find bound implications.

Theorem 3.1. *Given two unstable neurons and the LP formulations in Eq. (4), (5) and (6), the following holds: (1) The LP in Eq. (6) is always feasible; (2)* $l^*_{\text{no-imp}} \le l^*_{\text{relaxed}} \le l^*_{\text{imp}}$.

Proof. (1) To show that (6) is always feasible, we must show that there exists some $\boldsymbol{x}$ such that $\boldsymbol{x}_0 - \epsilon \le \boldsymbol{x} \le \boldsymbol{x}_0 + \epsilon$ also satisfies the other constraint $\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} \le 0$.

Given that the neuron $z_{j_2}^{(i_2)}$ is an unstable neuron, we know that $l_{j_2}^{(i_2)} \leq 0$. By definition of $l_{j_2}^{(i_2)}$, we know that $\min_{\boldsymbol{x}_0-\epsilon \leq \boldsymbol{x} \leq \boldsymbol{x}_0+\epsilon} \underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} \leq 0$, so such a $\boldsymbol{x}$ must exist to satisfy the constraint.

(2) The objective l^*_{relaxed} and l^*_{imp} have one additional constraint compared to $l^*_{\text{no-imp}}$, so $l^*_{\text{no-imp}} \leq l^*_{\text{relaxed}}$ and $l^*_{\text{no-imp}} \leq l^*_{\text{imp}}$. It remains to prove $l^*_{\text{relaxed}} \leq l^*_{\text{imp}}$.

Define $S := \{\boldsymbol{x} : z_{j_2}^{(i_2)}(\boldsymbol{x}) \leq 0, \boldsymbol{x} \in \mathcal{C}\}$ and $\bar{S} := \{\boldsymbol{x} : \underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} \leq 0, \boldsymbol{x} \in \mathcal{C}\}$. Here $\mathcal{C}$ is the set $\boldsymbol{x}_0 - \epsilon \leq \boldsymbol{x} \leq \boldsymbol{x}_0 + \epsilon$. We claim that $S \subseteq \bar{S}$. This is true because given any $\boldsymbol{x}' \in \mathcal{C}$, if $z_{j_2}^{(i_2)}(\boldsymbol{x}') \leq 0$ (by definition, $\boldsymbol{x}' \in S$), we have $\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x}' + \underline{c}^{(i_2,j_2)} \leq z_{j_2}^{(i_2)}(\boldsymbol{x}') \leq 0$ since $\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x}' + \underline{c}^{(i_2,j_2)}$ is a linear lower bound of $z_{j_2}^{(i_2)}(\boldsymbol{x}')$ found by α-CROWN. Thus, $S \subseteq \bar{S}$.

Since $S \subseteq \bar{S}$, (6) has the equal or larger feasible region compared to (5), and thus $l^*_{\text{relaxed}} \leq l^*_{\text{imp}}$. □

Theorem 3.2. *Given two unstable neurons and the LP formulations in (4), (6), if the following condition holds:*

$$\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x}^* + \underline{c}^{(i_2,j_2)} \leq 0$$

where each element of $\boldsymbol{x}^*$ is chosen as (here the subscript k means the k-th element in a vector):

$$x_k^* := \begin{cases} x_{0,k} - \epsilon \cdot \underline{a}_k^{(i_1,j_1)}, & \text{if } \underline{a}_k^{(i_1,j_1)} > 0 \\ x_{0,k} + \epsilon \cdot \underline{a}_k^{(i_1,j_1)}, & \text{if } \underline{a}_k^{(i_1,j_1)} < 0 \\ x_{0,k} - \epsilon \cdot \text{sign}(\underline{a}_k^{(i_2,j_2)}), & \text{if } \underline{a}_k^{(i_1,j_1)} = 0 \end{cases}$$

then $l^*_{\text{relaxed}} = l^*_{\text{no-imp}}$, i.e., there is no improvement. Additionally, one of the two cases must happen:

1. $\left|\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x}_0 + \underline{c}^{(i_2,j_2)}\right| - \epsilon \|\underline{\mathbf{a}}^{(i_2,j_2)}\|_1 > 0$*, i.e., the relaxed implicant constraint does not have an intersection with the ℓ_∞ box $\mathcal{C}$;*
2. $\left|\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x}_0 + \underline{c}^{(i_2,j_2)}\right| - \epsilon \|\underline{\mathbf{a}}^{(i_2,j_2)}\|_1 \leq 0$ *but* $\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x}^* + \underline{c}^{(i_2,j_2)} \leq 0$.

Proof. The linear programming problem (4) has a closed form optimal solution:

$$x'_k := \begin{cases} x_{0,k} - \epsilon \cdot \underline{a}_k^{(i_1,j_1)}, & \text{if } \underline{a}_k^{(i_1,j_1)} > 0 \\ x_{0,k} + \epsilon \cdot \underline{a}_k^{(i_1,j_1)}, & \text{if } \underline{a}_k^{(i_1,j_1)} < 0 \\ \text{don't care}, & \text{if } \underline{a}_k^{(i_1,j_1)} = 0 \end{cases}$$

If this solution x'_k already satisfies the additional constraint $\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} \leq 0$ added in (6), this constraint is redundant and thus $l^*_{\text{relaxed}} = l^*_{\text{no-imp}}$.

For the indices k where $\underline{a}_k^{(i_1,j_1)} = 0$, since x'_k is unconstrained, we can choose x'_k that minimize $\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)}$ to satisfy the new constraint in (6) as much as possible, by setting $x'_k = x_{0,k} - \epsilon \cdot \text{sign}(\underline{a}_k^{(i_2,j_2)})$. So the setting of x_k^* gives an optimal solution of (4) that has the minimum violation of the new constraint in (6). If x_k^* satisfies the new implicant constraint, the constraint is redundant, and $l^*_{\text{relaxed}} = l^*_{\text{no-imp}}$.

Now, we define the set $\hat{S} := \{\boldsymbol{x} : \underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} \leq 0\}$. Due to Theorem 3.1, the LP in (6) is always feasible, so $\hat{S} \cap \mathcal{C} \neq \emptyset$. Geometrically, the constraint $\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} \leq 0$ is redundant in two cases:

1. $\mathcal{C} \in \hat{S}$, or the ℓ_∞ box $\mathcal{C}$ does not intersect with the line $\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x} + \underline{c}^{(i_2,j_2)} = 0$. In this case, the ℓ_∞ distance from this line to the origin is greater than ϵ:

$$\frac{\left|\underline{\mathbf{a}}^{(i_2,j_2)\top} \boldsymbol{x}_0 + \underline{c}^{(i_2,j_2)}\right|}{\|\underline{\mathbf{a}}^{(i_2,j_2)}\|_1} > \epsilon$$

And this is the first case in this Theorem.

2. $\mathcal{C} \cap \hat{S} \neq \mathcal{C}$, when the line ${\underline{\mathbf{a}}^{(i_2,j_2)}}^\top \boldsymbol{x} + \underline{c}^{(i_2,j_2)} = 0$ cuts through $\mathcal{C}$ (ℓ_∞ distance is less than or equal to ϵ), however it does not remove the existing optimal solution for (4), i.e., ${\underline{\mathbf{a}}^{(i_2,j_2)}}^\top \boldsymbol{x}^* + \underline{c}^{(i_2,j_2)} \leq 0$.

This theorem allows us to filter out implicant neurons that have large ℓ_∞ distance to the origin first. Using the remaining implicant neurons, we further check each implicated unstable ReLU neuron to see if $\boldsymbol{x}^x$ satisfies the above constraint. This allows us to eliminate these pairs of neurons in calculation.

□

C ALGORITHMS

We show the full algorithms discussed in Section 3 in this section.

C.1 CLOSED-FORM SOLUTION OF THE RELAXED LP

In Section 3, we show that it is possible to use a fast optimization algorithm in $O(d_0 \log d_0)$ time, where d_0 is the neural network input dimension. Here we present this algorithm.

We solve the problem by adding a Lagrange multiplier ρ and derive a dual form of Eq. (6):

$$\begin{aligned}
&\min_{\boldsymbol{x}_0-\epsilon \leq \boldsymbol{x} \leq \boldsymbol{x}_0+\epsilon} \max_{\rho \geq 0} \ {\underline{\mathbf{a}}^{(i_1,j_1)}}^\top \boldsymbol{x} + \underline{c}^{(i_1,j_1)} + \rho({\underline{\mathbf{a}}^{(i_2,j_2)}}^\top \boldsymbol{x} + \underline{c}^{(i_2,j_2)}) \\
\geq &\max_{\rho \geq 0} \min_{\boldsymbol{x}_0-\epsilon \leq \boldsymbol{x} \leq \boldsymbol{x}_0+\epsilon} \ (\underline{\mathbf{a}}^{(i_1,j_1)} + \rho \underline{\mathbf{a}}^{(i_2,j_2)})^\top \boldsymbol{x} + \underline{c}^{(i_1,j_1)} + \rho \underline{c}^{(i_2,j_2)} \\
= &\max_{\rho \geq 0} \ (\underline{\mathbf{a}}^{(i_1,j_1)} + \rho \underline{\mathbf{a}}^{(i_2,j_2)})^\top \boldsymbol{x}_0 + \underline{c}^{(i_1,j_1)} + \rho \underline{c}^{(i_2,j_2)} - \|(\underline{\mathbf{a}}^{(i_1,j_1)} + \rho \underline{\mathbf{a}}^{(i_2,j_2)})\|_1 \cdot \epsilon \\
= &\max_{\rho \geq 0} \ -\|(\underline{\mathbf{a}}^{(i_1,j_1)} + \rho \underline{\mathbf{a}}^{(i_2,j_2)})\|_1 \cdot \epsilon + \rho({\underline{\mathbf{a}}^{(i_2,j_2)}}^\top \boldsymbol{x}_0 + \underline{c}^{(i_2,j_2)}) + \underline{\mathbf{a}}^{(i_1,j_1)} \boldsymbol{x}_0 + \underline{c}^{(i_1,j_1)}
\end{aligned} \tag{13}$$

Note that the inner minimization is solved in closed form using Hölder's inequality. The maximization problem over the dual variable ρ is a one-dimensional, non-smooth, piece-wise linear, and concave optimization problem and can be solved by checking super-gradients at the endpoints of all linear pieces. We list the solving procedure in Algorithm 1:

C.2 ALGORITHM OF BIG CONSTRUCTION AND UTILIZATION

In Algorithm 3, We will introduce how to construct BIG after we obtain the linear equation of all unstable neurons (can be calculated by CROWN or α-CROWN) in the verification process. Note that there are **For** loops Algorithm 3, the calculations inside them are independent and can be calculated in parallel though. In our experiments, we construct BIG efficiently by computing all implications only in four batches for four cases of neuron interactions.

D BOUND IMPLICATIONS WITH ADDITIONAL SPLIT CONSTRAINTS

To solve Eq. (8), we leverage k Lagrange multipliers $\boldsymbol{\rho}_i$, where $i \in [1, 2, .., k]$ to derive the dual form solution:

$$\begin{aligned}
\max_{\boldsymbol{\rho} \geq 0} \min_{\boldsymbol{x}} \ {\underline{\mathbf{a}}^{(i_1,j_1)}}^\top \boldsymbol{x} + \underline{c}^{(i_1,j_1)} + \sum_{k=1}^{M} \boldsymbol{\rho}_i({\underline{\mathbf{a}}^{(\boldsymbol{m}[k],\boldsymbol{n}[k])}}^\top \boldsymbol{x} + \underline{c}^{(\boldsymbol{m}[k],\boldsymbol{n}[k])}) \\
\text{s.t. } \boldsymbol{x}_0 - \epsilon \leq \boldsymbol{x} \leq \boldsymbol{x}_0 + \epsilon
\end{aligned} \tag{14}$$

For the ℓ_∞-norm constraint of $\boldsymbol{x}$, the inner minimization has a closed-form solution:

Algorithm 1 The closed-form solution of Eq. (6).

Inputs: $\underline{\mathbf{a}}^{(i_1,j_1)}$, $\underline{c}^{(i_1,j_1)}$, $\underline{\mathbf{a}}^{(i_2,j_2)}$, $\underline{c}^{(i_2,j_2)}$, $\boldsymbol{x}_0$, ϵ
Outputs: The optimal solution of Eq. (6): l^*_{relaxed}
$\boldsymbol{q} \leftarrow -\underline{\mathbf{a}}^{(i_1,j_1)} / \underline{\mathbf{a}}^{(i_2,j_2)}$
$\boldsymbol{I} \leftarrow \text{argsort}(\boldsymbol{q})$ ▷ Dominates time complexity
$\underline{\mathbf{a}}^{(i_2,j_2)}_{\text{sorted}} \leftarrow \{\underline{\mathbf{a}}^{(i_2,j_2)}_{\boldsymbol{I}_1} \cdot \epsilon, \underline{\mathbf{a}}^{(i_2,j_2)}_{\boldsymbol{I}_2} \cdot \epsilon, ..., \underline{\mathbf{a}}^{(i_2,j_2)}_{\boldsymbol{I}_{d_0}} \cdot \epsilon\}$ ▷ Sort $\underline{\mathbf{a}}^{(i_2,j_2)}$ by index I and scale by ϵ
$\left(\underline{\mathbf{a}}^{(i_2,j_2)}_{-}\right)_i \leftarrow -\sum_{k=1}^{i}(|\underline{\mathbf{a}}^{(i_2,j_2)}_{\text{sorted}}|_k),\ i \in [d_0]$ ▷ Cumulative sum of $|\underline{\mathbf{a}}^{(i_2,j_2)}_{\text{sorted}}|$ by its length d_0
$\underline{\mathbf{a}}^{(i_2,j_2)}_{+} \leftarrow \underline{\mathbf{a}}^{(i_2,j_2)}_{-} - \underline{\mathbf{a}}^{(i_2,j_2)}_{-d_0}$ ▷ Shift $\underline{\mathbf{a}}^{(i_2,j_2)}_{-}$ to positive range
$\nabla\underline{\mathbf{a}} \leftarrow \underline{\mathbf{a}}^{(i_2,j_2)}_{+} + \underline{\mathbf{a}}^{(i_2,j_2)}_{-} + (\underline{\mathbf{a}}^{(i_2,j_2)^\top}\boldsymbol{x}_0 + \underline{c}^{(i_2,j_2)})$ ▷ Calculate super-gradient
$i^* \leftarrow i$ where $\nabla\underline{\mathbf{a}}_i = 0$ ▷ Find the index of super-gradient is 0
$\rho^* \leftarrow \max(\boldsymbol{q}_{\boldsymbol{I}_{i^*}}, 0)$ ▷ Find the best ρ and introduce it to Eq. 13
$l^*_{\text{relaxed}} \leftarrow -\|(\underline{\mathbf{a}}^{(i_1,j_1)} + \rho^*\underline{\mathbf{a}}^{(i_2,j_2)})\|_1 \cdot \epsilon + \rho^*(\underline{\mathbf{a}}^{(i_2,j_2)^\top}\boldsymbol{x}_0 + \underline{c}^{(i_2,j_2)}) + \underline{\mathbf{a}}^{(i_1,j_1)}\boldsymbol{x}_0 + \underline{c}^{(i_1,j_1)}$
$l^*_0 \leftarrow -\|\underline{\mathbf{a}}^{(i_1,j_1)}\|_1 \cdot \epsilon + \underline{c}^{(i_2,j_2)} + \underline{\mathbf{a}}^{(i_1,j_1)}\boldsymbol{x}_0 + \underline{c}^{(i_1,j_1)}$ ▷ Objective when $\rho = 0$
if $l^*_{\text{relaxed}} < l^*_0$ **then**
    $l^*_{\text{relaxed}} \leftarrow l^*_0$ ▷ Compare to l^*_0, which is an additional end point
Return: l^*_{relaxed}

Algorithm 2 Filter Top-K constraints by distance to $\boldsymbol{x}_0$.

Inputs: constraints of all unstable neurons $\underline{\mathbf{a}}, \overline{\mathbf{a}} \in \mathbb{R}^{N \times d_0}$ and $\underline{c}, \overline{c} \in \mathbb{R}^N$, where N is number of unstable neurons and d_0 is the length of the model input $\boldsymbol{x}_0$; perturbation size ϵ, K, $\underline{\mathbf{S}} = \emptyset, \overline{\mathbf{S}} = \emptyset$
function TOPKFILTERING($\underline{\mathbf{a}}, \overline{\mathbf{a}}, \underline{c}, \overline{c}, \boldsymbol{x}_0, \epsilon, K$)
    for $p = 1$ to N **do**
        $\underline{\mathbf{S}} \cup |\underline{\mathbf{a}}^{(i_p,j_p)^\top}\boldsymbol{x}_0 + \underline{c}^{(i_p,j_p)}| - \epsilon \cdot ||\underline{\mathbf{a}}^{(i_p,j_p)}||_1$
        $\overline{\mathbf{S}} \cup |\overline{\mathbf{a}}^{(i_p,j_p)^\top}\boldsymbol{x}_0 + \overline{c}^{(i_p,j_p)}| - \epsilon \cdot ||\overline{\mathbf{a}}^{(i_p,j_p)}||_1$
    ▷ Sort from nearest to farthest distance
    $\underline{\mathbf{S}} = \texttt{argsort}(\underline{d}_p)[:K]$ ▷ Indices of Top-K inactive implicant constraints
    $\overline{\mathbf{S}} = \texttt{argsort}(\overline{d}_p)[:K]$ ▷ Indices of Top-K active implicant constraints
return $\underline{\mathbf{S}}, \overline{\mathbf{S}}$

$$\max_{\boldsymbol{\rho}\geq 0} \underline{\mathbf{a}}^{(i_1,j_1)^\top}\boldsymbol{x}_0 + \underline{c}^{(i_1,j_1)} - \|\underline{\mathbf{a}}^{(i_1,j_1)}\|_1 \cdot \epsilon + \sum_{k=1}^{M} \boldsymbol{\rho}_i\underline{\mathbf{a}}^{(\boldsymbol{m}[k],\boldsymbol{n}[k])^\top}\boldsymbol{x}_0 + \boldsymbol{\rho}_i\underline{c}^{(\boldsymbol{m}[k],\boldsymbol{n}[k])} - \|\boldsymbol{\rho}_i\underline{\mathbf{a}}^{(\boldsymbol{m}[k],\boldsymbol{n}[k])}\|_1 \cdot \epsilon \tag{15}$$

Then, we can easily solve $\boldsymbol{\rho}$ by projected gradient descent using an optimizer like Adam (Diederik et al., 2014).

E EXPERIMENTAL DETAILS

E.1 VERIHARD BENCHMARK

The **VeriHard** benchmark includes models from existing benchmarks, such as MNIST-A-Adv, CIFAR-A-Adv, CIFAR-A-Mix from SDP-FO (Dathathri et al., 2020), and CIFAR100-small, CIFAR100-medium, CIFAR100-large, and TinyImageNet-medium from the neural networks competition (VNN-COMP (Bak et al., 2021; Müller et al., 2022a)). **VeriHard** applies a careful selection of

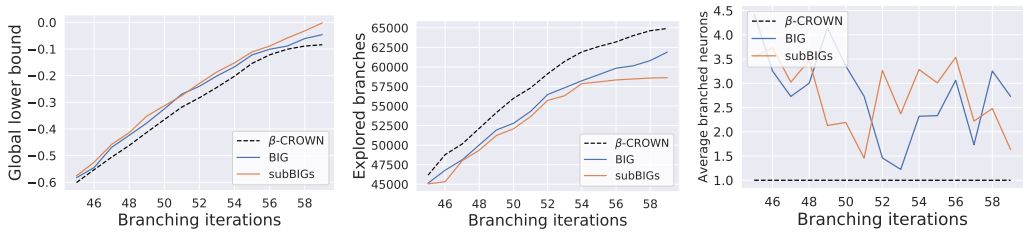

Figure 4: {**(Left)** Global lower bounds; **(Middle)** Total explored branches; **(Right)** Average used bound implications} trends along with branching iterations on one specific instance in **VeriHard**-MNIST-A-Adv benchmark. More than one neurons are branched per iteration due to BIG, and the bounds and the number explored branches both improve noticeably.

perturbation size ϵ for each instance, aiming to increase the verification challenge while ensuring solvability. We will illustrate our ϵ configuration algorithm as follows:

We start by determining the unknown range $[\epsilon_l, \epsilon_r]$ of each instance. Given an instance with an initial $\epsilon_l = 0$, we increment ϵ_l step by step with step-size $\alpha = 0.01$, applying CROWN verification after each step, until the vanilla CROWN verifier fails to verify the instance at the current ϵ_l. Conversely, we initialize ϵ_r to 1.0, decreasing it by the same step size 0.01 until PGD attack succeed with perturbation bound ϵ_r. In this case, any ϵ configuration within the range $[\epsilon_l, \epsilon_r]$ enforces verification using the BaB process.

For determining the final ϵ, we employ β-CROWN as our reference verifier. Binary search on ϵ is used to assess the feasibility of verifying the instance within a timeout t, which is randomly sampled from the interval $[0.8T, 1.5T]$ (T is the final timeout designated for the instance). The resulting ϵ will make β-CROWN verifier running time to be as close to the given timeout t as possible.

In **VeriHard** dataset, for MNIST and CIFAR10 instances, the actual timeout T is set to 200 seconds, whereas for CIFAR100 and TinyImageNet instances, T is set to 250 seconds. We crafted 100 instances from each benchmark, assigning the respective ϵ to each instance following the algorithm mentioned before. We will release our benchmark in the standard VNNLIB format.

E.2 Experimental Setup

Our implementation is based on the open-source α, β-CROWN verifier by integrating BIG and subBIGs construction and branching utilization code into that. For BIG and subBIGs construction, we select top-K implicant neurons to derive our paired bound implications as mentioned in Theorem 3.2 with $K = 1000$. Specifically for subBIGs, we use first $M = 8$ splits by BaB and apply a 50-steps gradient descent with Adam optimizer to optimize our objective $l^*_{\text{relaxed-multi}}$ on up to $2^M = 256$ unsolved subproblems with initial learning rate as 0.1 and its decay factor as 0.99. Throughout our experiments, we employ Filtered Smart Branching (FSB) (De Palma et al., 2021a) as our branching heuristics and apply Adam optimizer to optimize both α and β for 20 iterations during verification process. The initial learning rate is set to be 0.1 for α optimization and 0.05 for β, while the corresponding decay ratio is set to be 0.995 for α and 0.98 for β. All our experiments are conducted on one NVIDIA A100 GPU device (80G memory). Timeout for classic benchmark is aligned with prior work: MNIST-CNN-A-Adv (200s), CIFAR10-CNN-A-Adv (200s), CIFAR10-CNN-A-Mix (200s), MNIST-ConvSmall (180s), CIFAR10-ConvSmall (180s). For **VeriHard** benchmark, we set timeout to be 200 seconds for MNIST and CIFAR10 instances and 250 second for CIFAR100 and TinyImageNet instances.

F Visualization of BIG-enhanced BaB process

In this section, we provide more figures showing the bounds improvements, the number of branches, and the average number of branched neurons per BaB iteration, on different benchmarks and data-points.

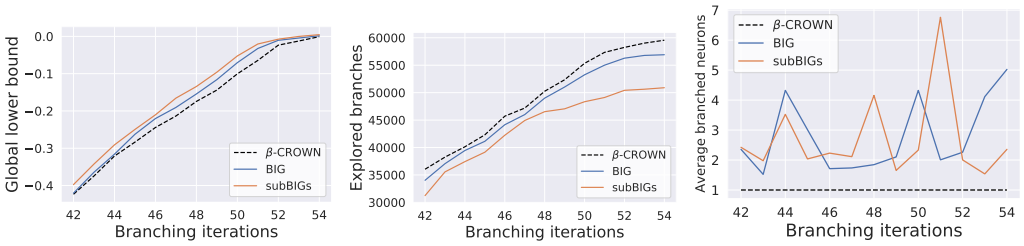

Figure 5: {**(Left)** Global lower bounds; **(Middle)** Total explored branches; **(Right)** Average used bound implications} trends along with branching iterations on one specific instance in **VeriHard**-CIFAR10-A-Adv benchmark.

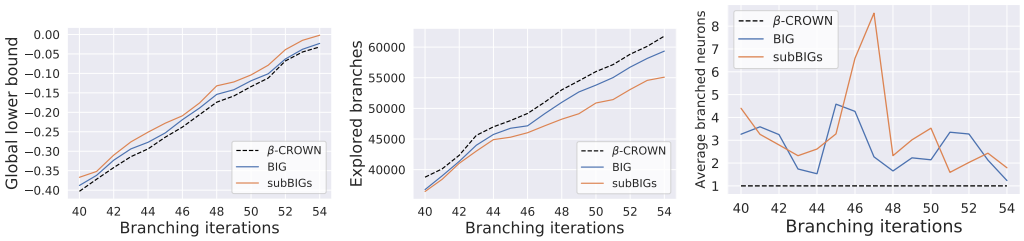

Figure 6: {**(Left)** Global lower bounds; **(Middle)** Total explored branches; **(Right)** Average used bound implications} trends along with branching iterations on one specific instance in **VeriHard**-CIFAR10-A-Mix benchmark.

Algorithm 3 Construct BIG

Inputs: constraints of all unstable neurons $\underline{\mathbf{a}}, \overline{\mathbf{a}} \in \mathbb{R}^{N \times d_0}$ and $\underline{c}, \overline{c} \in \mathbb{R}^N$, where N is number of unstable neurons and d_0 is the length of the model input $\boldsymbol{x}_0$; perturbation size ϵ, K
Outputs: BIG: $\mathbf{G}(\mathbf{V}, \mathbf{E})$
$\underline{\mathbf{S}}, \overline{\mathbf{S}} \leftarrow$ TOPKFILTERING($\underline{\mathbf{a}}, \overline{\mathbf{a}}, \underline{c}, \overline{c}, \boldsymbol{x}_0, \epsilon, K$) ▷ Get indices of Top-$K$ constraints from Alg. 2
for $p = 1$ to N **do** ▷ Iterate implicant neurons
  for $q = 1$ to N **do** ▷ Iterate implicated neurons
    if $p \in \underline{\mathbf{S}}$ **then**
      ▷ Consider inactive implicant constraint to improve lower bound

$$x_k^* := \begin{cases} x_{0,k} - \epsilon \cdot \underline{a}_k^{(i_q,j_q)}, & \text{if } \underline{a}_k^{(i_q,j_q)} > 0 \\ x_{0,k} + \epsilon \cdot \underline{a}_k^{(i_q,j_q)}, & \text{if } \underline{a}_k^{(i_q,j_q)} < 0 \\ x_{0,k} - \epsilon \cdot \text{sign}(\underline{a}_k^{(i_p,j_p)}), & \text{if } \underline{a}_k^{(i_q,j_q)} = 0 \end{cases}$$

      if $\underline{\mathbf{a}}^{(i_p,j_p)\top} x^* + \underline{c}^{(i_p,j_p)} > 0$ **then** ▷ Useful implicant constraint
        $l^*_{\text{relaxed}} \leftarrow$ Solve Eq. 6 with the implicated neurons: $\underline{\mathbf{a}}^{(i_q,j_q)\top} \boldsymbol{x} + \underline{c}^{(i_q,j_q)}$
        if $l^*_{\text{relaxed}} \geq 0$ **then** ▷ $z_{j_q}^{(i_q)}$ become to active
          $\mathbf{E} \leftarrow \mathbf{E} \cup (Q_{j_p}^{(i_p)}, P_{j_q}^{(i_q)})$ ▷ add edge $(Q_{j_p}^{(i_p)}, P_{j_q}^{(i_q)})$ to the graph
      ▷ Consider inactive implicant constraint to improve upper bound

$$x_k^* := \begin{cases} x_{0,k} - \epsilon \cdot \overline{a}_k^{(i_q,j_q)}, & \text{if } \overline{a}_k^{(i_q,j_q)} > 0 \\ x_{0,k} + \epsilon \cdot \overline{a}_k^{(i_q,j_q)}, & \text{if } \overline{a}_k^{(i_q,j_q)} < 0 \\ x_{0,k} - \epsilon \cdot \text{sign}(\underline{a}_k^{(i_p,j_p)}), & \text{if } \underline{a}_k^{(i_q,j_q)} = 0 \end{cases}$$

      if $\underline{\mathbf{a}}^{(i_p,j_p)\top} x^* + \underline{c}^{(i_p,j_p)} > 0$ **then** ▷ Useful implicant constraint
        $l^*_{\text{relaxed}} \leftarrow$ Solve Eq. 10 with the implicated neurons: $\overline{\mathbf{a}}^{(i_q,j_q)\top} \boldsymbol{x} + \overline{c}^{(i_q,j_q)}$
        if $l^*_{\text{relaxed}} \leq 0$ **then** ▷ $z_{j_q}^{(i_q)}$ become to inactive
          $\mathbf{E} \leftarrow \mathbf{E} \cup (Q_{j_p}^{(i_p)}, Q_{j_q}^{(i_q)})$ ▷ add edge $(Q_{j_p}^{(i_p)}, Q_{j_q}^{(i_q)})$ to the graph
    if $p \in \overline{\mathbf{S}}$ **then**
      ▷ Consider active implicant constraint to improve lower bound

$$x_k^* := \begin{cases} x_{0,k} - \epsilon \cdot \underline{a}_k^{(i_q,j_q)}, & \text{if } \underline{a}_k^{(i_q,j_q)} > 0 \\ x_{0,k} + \epsilon \cdot \underline{a}_k^{(i_q,j_q)}, & \text{if } \underline{a}_k^{(i_q,j_q)} < 0 \\ x_{0,k} + \epsilon \cdot \text{sign}(\overline{a}_k^{(i_p,j_p)}), & \text{if } \underline{a}_k^{(i_q,j_q)} = 0 \end{cases}$$

      if $\overline{\mathbf{a}}^{(i_p,j_p)\top} x^* + \overline{c}^{(i_p,j_p)} < 0$ **then** ▷ Useful implicant constraint
        $l^*_{\text{relaxed}} \leftarrow$ Solve Eq. 11 with the implicated neurons: $\underline{\mathbf{a}}^{(i_q,j_q)\top} \boldsymbol{x} + \underline{c}^{(i_q,j_q)}$
        if $l^*_{\text{relaxed}} \geq 0$ **then** ▷ $z_{j_q}^{(i_q)}$ become to inactive
          $\mathbf{E} \leftarrow \mathbf{E} \cup (P_{j_p}^{(i_p)}, Q_{j_q}^{(i_q)})$ ▷ add edge $(P_{j_p}^{(i_p)}, Q_{j_q}^{(i_q)})$ to the graph
      ▷ Consider active implicant constraint to improve upper bound

$$x_k^* := \begin{cases} x_{0,k} - \epsilon \cdot \overline{a}_k^{(i_q,j_q)}, & \text{if } \overline{a}_k^{(i_q,j_q)} > 0 \\ x_{0,k} + \epsilon \cdot \overline{a}_k^{(i_q,j_q)}, & \text{if } \overline{a}_k^{(i_q,j_q)} < 0 \\ x_{0,k} + \epsilon \cdot \text{sign}(\overline{a}_k^{(i_p,j_p)}), & \text{if } \overline{a}_k^{(i_q,j_q)} = 0 \end{cases}$$

      if $\overline{\mathbf{a}}^{(i_p,j_p)\top} x^* + \overline{c}^{(i_p,j_p)} < 0$ **then** ▷ Useful implicant constraint
        $l^*_{\text{relaxed}} \leftarrow$ Solve Eq. 12 with the implicated neurons: $\overline{\mathbf{a}}^{(i_q,j_q)\top} \boldsymbol{x} + \overline{c}^{(i_q,j_q)}$
        if $l^*_{\text{relaxed}} \leq 0$ **then** ▷ $z_{j_q}^{(i_q)}$ become to active
          $\mathbf{E} \leftarrow \mathbf{E} \cup (P_{j_p}^{(i_p)}, P_{j_q}^{(i_q)})$ ▷ add edge $(P_{j_p}^{(i_p)}, P_{j_q}^{(i_q)})$ to the graph
Return: $\mathbf{G}(\mathbf{V}, \mathbf{E})$

Algorithm 4 Generate subproblems in one BaB iteration with BIG.

Inputs: $\mathbf{G}(\mathbf{V}, \mathbf{E})$, selected unstable neuron $\boldsymbol{z}_j^{(i)}$
Outputs: BIG-generated subproblems.
$\mathbf{S}_P \leftarrow \{P_j^{(i)}\}, \mathbf{S}_Q \leftarrow \{Q_j^{(i)}\}$ ▷ traverse BIG starting from node $P_j^{(i)}$ and $Q_j^{(i)}$
while $\mathbf{S}_P$ has unvisited nodes **do**
    Pick a unvisited node $X_{j'}^{(i')}$ out from $\mathbf{S}_p$
    for $e = (X_{j'}^{(i')}, Y_{j^*}^{(i^*)}) \in \mathbf{E}$ **do**
        $\mathbf{S}_P \leftarrow \mathbf{S}_P \cup \{Y_{j^*}^{(i^*)}\}$
    Mark node $X_{j'}^{(i')}$ as visited
while $\mathbf{S}_Q$ has unvisited nodes **do**
    Pick a unvisited node $X_{j'}^{(i')}$ out from $\mathbf{S}_Q$
    for $e = (X_{j'}^{(i')}, Y_{j^*}^{(i^*)}) \in \mathbf{E}$ **do**
        $\mathbf{S}_Q \leftarrow \mathbf{S}_Q \cup \{Y_{j^*}^{(i^*)}\}$
    Mark node $X_{j'}^{(i')}$ as visited
Add splits $z_{j'}^{(i')} \geq 0$ (or $z_{j'}^{(i')} \leq 0$) from nodes in $\mathbf{S}_P$ to subproblem $z_j^{(i)} \geq 0$
Add splits $z_{j'}^{(i')} \leq 0$ (or $z_{j'}^{(i')} \leq 0$) from nodes in $\mathbf{S}_Q$ to subproblem $z_j^{(i)} \leq 0$