8. Appendix

8.1. Experiment Setups

In this subsection, we present some experiment setups in detail. Concretely, we list the sources of networks, the size of the test set, and the initial perturbation range in Table 3. We evaluate methods on 10 inputs for all CNNs in Table 1 and 2. As CIFAR_Conv_MaxPool has low accuracy and low robustness, the size of the test set is 50 and the initial perturbation range is 0.00005. Testing on 10 inputs can sufficiently evaluate the performance of verification methods, as it is shown that the average certified results of 1000 inputs are similar to 10 images [4].

8.2. Additional Experiments

In this subsection, we conduct some additional experiments to further illustrate (I) the advantage of the block-wise tightness over the neuron-wise tightness, (II) the time efficiency of MaxLin compared to other BaB-based verification tools, (III) the performance of MaxLin using multi-neuron abstraction techniques, and (IV) the performance of MaxLin on PointNets.

We list the sources of networks for additional experiments in Table 3. We evaluate methods on 100 inputs for ERAN benchmark (CIFAR_Conv_MaxPool) and 100 inputs for PointNets. The perturbation range ϵ is 0.005 for PointNets and is 0.0007, 0.0008, 0.0009, 0.0010, or 0.0011 for CIFAR_Conv_MaxPool. We follow the metrics used in the baseline methods. As for the effectiveness, we use certified accuracy, the percentage of the successfully verified inputs against the perturbation range, to evaluate the tightness of methods. We also use the average per-example verified time as the metric for time efficiency. In additional experiments II and III, we compare MaxLin with three state-of-the-art verification techniques, which use the BaB and multi-neuron abstraction technique to enhance the precision. Concretely, the baselines are MN-BaB [12], α,β-CROWN [41, 48, 50] (VNN-COMP 2021 [2] and 2022 [32] winner), and ERAN. In additional experiment IV, we evaluate the performance of MaxLin and three single-neuron abstraction methods(DeepPoly [37], 3DCertify [27], and Ti-Lin [45]) on PointNets.

8.2.1 Results (I): Advantages Over The Neuron-wise Tightest Method

To further illustrate the advantage of the block-wise tightness (MaxLin) over the neuron-wise tightness(Ti-Lin), we analyze the verified interval of the output neurons of the last layer. In Figure 4, we compare the results computed by MaxLin and Ti-Lin on CIFAR_Conv_MaxPool and MNIST_LeNet_Tanh, whose actvations are ReLU and S-shaped, respectively. For the output neurons(10 labels), we use $(\boldsymbol{l}, \boldsymbol{u})$ and $(\boldsymbol{l'}, \boldsymbol{u'})$ to represent the output bound of Ti-Lin and MaxLin testing on 100 inputs, respectively. We use the red and blue dots to represent $\boldsymbol{u} - \boldsymbol{u'}$ and $\boldsymbol{l} - \boldsymbol{l'}$, respectively. The x-axis represents the output neuron index(label), and the y-axis represents the deviations between the lower and upper bounds of the intervals.

As the activation of CIFAR_Conv_MaxPool is ReLU, the lower bounds of the ouput neurons are mostly zero and thus

Table 3. The additional experimental setup and source of neural networks used in experiments. ϵ_0 is the initial perturbation range in Algorithm 1. The third column represents the size of the input test set.

Dataset	Network	Size	ϵ_0	Source
MNIST	Conv_MaxPool	10	0.005	ERAN
	CNN, 4 layers	10	0.005	CNN-Cert
	CNN, 5 layers	10	0.005	
	CNN, 6 layers	10	0.005	
	CNN, 7 layers	10	0.005	
	CNN, 8 layers	10	0.005	
	LeNet_ReLU	10	0.005	
	LeNet_Sigmoid	10	0.005	
	LeNet_Tanh	10	0.005	
	LeNet_Atan	10	0.005	
CIFAR-10	Conv_MaxPool	50	0.00005	ERAN
	CNN, 4 layers	10	0.005	CNN-Cert
	CNN, 5 layers	10	0.005	
	CNN, 6 layers	10	0.005	
	CNN, 7 layers	10	0.005	
	CNN, 8 layers	10	0.005	
Tiny ImageNet	CNN, 7 layers	10	0.005	
ModelNet40	16p_natural	100	0.005	3DCertify
	32p_natural	100	0.005	
	64p_natural	100	0.005	
	128p_natural	100	0.005	
	256p_natural	100	0.005	

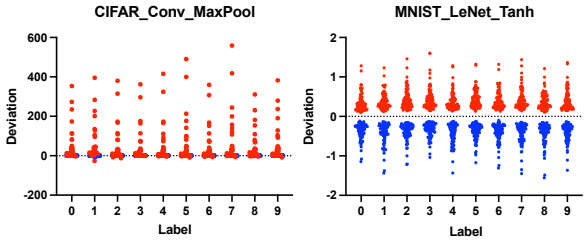

Figure 4. Visualization of the output intervals verified by MaxLin and Ti-Lin. $(\boldsymbol{l}, \boldsymbol{u})$ and $(\boldsymbol{l'}, \boldsymbol{u'})$ represent the output bound of Ti-Lin and MaxLin testing on 100 inputs, respectively. Red and blue dots represent $\boldsymbol{u} - \boldsymbol{u'}$ and $\boldsymbol{l} - \boldsymbol{l'}$, respectively.

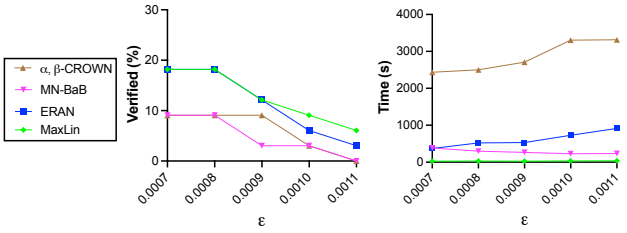

Figure 5. Certified accuracy(%) and average per-example verification time(s) on CIFAR_Conv_MaxPool tested by MN-BaB, α,β-CROWN, ERAN, and MaxLin(using single-neuron techniques for ReLU).

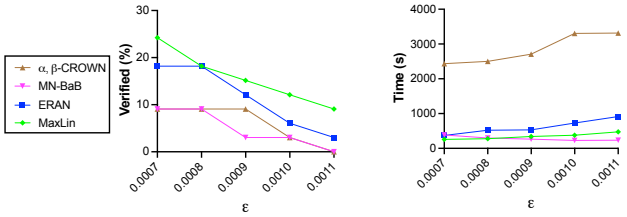

Figure 6. Certified accuracy(%) and average per-example verification time(s) on CIFAR_Conv_MaxPool tested by MN-BaB, α,β-CROWN, ERAN, and MaxLin(using multi-neuron techniques for ReLU).

the deviation of lower bounds is zero. Except for this case, most $u - u'$ are larger than zero and $l - l'$ are smaller than zero in Figure 4. It reveals that the block-wise tightest upper linear bounds could bring tighter output intervals than the neuron-wise tightest linear bounds. Consequently, MaxLin could certify much larger robustness bounds than Ti-Lin in Table 1.

8.2.2 Results (II): Performance Using Single-neuron Abstraction for ReLU

We conduct additional experiments to present the time efficiency of the single-neuron abstraction technique, which is used by MaxLin in Section 5. We compare MaxLin(single-neuron abstraction for ReLU) with three state-of-the-art verification techniques(MN-BaB [12], α,β-CROWN [41, 48], and ERAN using multi-neuron abstraction) on CIFAR_Conv_MaxPool. The results are presented in Figure 5. In terms of time efficiency, MaxLin can accelerate the computation process with up to $14.1, 96.5$, and $23.8\times$ compared to MN-BaB, α, β-CROWN, and ERAN, respectively. Although MaxLin uses the single-neuron abstraction technique for ReLU, MaxLin still can enhance precision with up to 9.1, 9.1, and 3.0% improvement compared to MN-BaB, α, β-CROWN, and ERAN , respectively. These results demonstrate that the single-neuron abstraction technique has the potential of verifying large models and other complex models, such as PointNets(results in Subsection 8.2.4)

8.2.3 Results (III): Performance Using Multi-neuron Abstraction for ReLU

As ERAN framework, atop which MaxLin is built, not only supports single-neuron abstraction but also integrates the multi-neuron abstraction for ReLU. We compare MaxLin using multi-neuron abstraction for ReLU to MN-BaB, α, β-CROWN, and ERAN using multi-neuron abstraction on CIFAR_Conv_MaxPool. The results are shown in Figure 6. The results show that MaxLin has higher certified accuracy with up to 15.2, 15.2, 6.1% improvement compared to MN-BaB, α, β-CROWN, and ERAN, respectively. In terms of time efficiency, MaxLin has similar time cost to MN-BaB and has up to 9.5 and 1.9 $\times$ speedup compared to α, β-CROWN and ERAN, respectively. These results show that if using multi-neuron abstraction, MaxLin also could have higher certified accuracy and less time cost than these verification tools.

8.2.4 Results (IV): Performance on PointNets

3D point cloud models are widely used and achieve great success in some safety-critical domains, such as autonomous driving. It is of vital importance to provide a provable robustness guarantee to models before deployed.

As 3DCertify is a robustness verifier for point cloud models, MaxLin, which is built atop the 3DCertify framework, can be extended to certify the robustness of 3D Point Cloud models against point-wise l_∞ perturbation and 3D transformation. Here, we demonstrate that MaxLin is not only useful beyond image classification models but also performs well on other models. To that end, we show certification results against point-wise l_∞ perturbation on seven PointNets for the ModelNet40 [44] dataset in Table 4. The PointNet whose inputs' point number is k is denoted as kp_natural. As Ti-Lin is the best state-of-the-art tool built on the CNN-Cert framework, we integrate its linear bounds for MaxPool into the 3DCertify framework as one baseline. The generation way of the test set is the same as 3DCertify and all experiments use the same random subset of 100 objects from the ModelNet40 [44] dataset. The certification results represent the percentage of verified robustness properties and the perturbation range is 0.005. The perturbation is in l_∞ norm and is measured by Hausdroff distance.

As for tightness, the certification results in Table 4 show that MaxLin outperforms other tools in all cases in terms of tightness. It is reasonable that the results for 16p_natural and 256_natural certified by 3DCertify and MaxLin are the same, as the perturbation range is not large enough to dis-

Table 4. Averaged certified results and runtime on PointNet on the ModelNet40 datasets tested by DeepPoly, 3DCertify, Ti-Lin, and MaxLin, where 16p_natural represents the PointNet model is naturally trained and the number of its point input is 16.

Dataset	Network	Certified accuracy (%)				Average Runtime(second)				Speedup
		DeepPoly	3DCertify	Ti-Lin	MaxLin	DeepPoly	3DCertify	Ti-Lin	MaxLin	vs. 3DCertify
ModelNet40	16p_natural	72.73	**74.03**	**74.03**	**74.03**	10.37	18.22	12.47	10.61	**1.72**
	32p_natural	54.88	58.54	58.54	**64.63**	17.88	34.79	21.24	18.06	**1.93**
	64p_natural	32.56	40.70	36.05	**47.67**	35.85	96.04	42.96	36.63	**2.62**
	128p_natural	4.55	11.36	2.27	**14.77**	81.46	207.04	110.16	85.90	**2.41**
	256p_natural	1.12	**4.49**	1.12	**4.49**	178.34	494.70	248.19	199.88	**2.47**

tinguish the tightness of these tools. As for efficiency, in Table 4, MaxLin has up to 2.62 $\times$ speedup compared with 3DCertify and is slightly faster than Ti-Lin. MaxLin has almost the same time consumption as DeepPoly. In summary, these results demonstrate that our fine-grained linear approximation can help improve both the tightness and efficiency of robustness verification of other models beyond the image classification domain.

8.3. Complexity analysis of MaxLin

For a K-layer convolutional network, we assume that the k-th layer has n_k neurons and the filter size is $k \times k$. The time complexity of backsubstitution is $\mathcal{O}(K \times \max n_k^3)$ [49] and the backsubstitution process will be repeated $K-1$ times to verify one input perturbed within a certain perturbation range. Therefore, the time complexity of MaxLin is $\mathcal{O}(K^2 \times \max {n_k}^3)$.

8.4. Proof of Theorem 1

We prove the correctness of linear bounds in Theorem 1 as follows.

Proof. **Upper linear bound:**

Case 1:

When $(l_i = l_{max}) \wedge (l_i \geq u_j)$, $u(x_1, \cdots, x_n) = x_i$ and $f(x_1, \cdots, x_n) = x_i$. Therefore,

$$\begin{aligned} u(x_1, \cdots, x_n) - f(x_1, \cdots, x_n) =& x_i - x_i \\ =& 0 \end{aligned}$$

Case 2:

Otherwise, $f(x_1, \cdots, x_n) = max\{x_1, \cdots, x_n\}$

If $f(x_1, \cdots, x_n) = x_i$,

$$\begin{aligned} & u(x_1, \cdots, x_n) - f(x_1, \cdots, x_n) \\ =& \frac{u_i - u_j}{u_i - l_i}(x_i - l_i) + u_j - x_i \\ =& \frac{l_i - u_j}{u_i - l_i} x_i - \frac{u_j - l_i}{u_i - l_i} u_i \\ =& \frac{u_j - l_i}{u_i - l_i}(u_i - x_i) \\ \geq& 0 \end{aligned}$$

If $f(x_1, \cdots, x_n) = x_j$,

$$\begin{aligned} & u(x_1, \cdots, x_n) - f(x_1, \cdots, x_n) \\ =& \frac{u_i - u_j}{u_i - l_i}(x_i - l_i) + u_j - x_j \\ \geq& 0 \end{aligned}$$

Otherwise, we assume $f(x_1, \cdots, x_n) = x_q$, where $q \neq i, j$ and $q \in [n]$.

$$\begin{aligned} & u(x_1, \cdots, x_n) - f(x_1, \cdots, x_n) \\ =& \frac{u_i - u_j}{u_i - l_i}(x_i - l_i) + u_j - x_q \\ \geq& \frac{u_i - u_j}{u_i - l_i}(x_i - l_i) + u_j - u_q \\ \geq& 0 \end{aligned}$$

Lower linear bound:

$$\begin{aligned} f(x_1, \cdots, x_n) =& max(x_1, \cdots, x_n) \\ \geq& x_j \\ =& l(x_1, \cdots, x_n) \end{aligned}$$

This completes the proof. □

8.5. Proof of Theorem 2

First, we prove minimizing the volume of the over-approximation zone of the linear bounds $U_b^{k+1}(\cdot)$ and $L_b^{k+1}(\cdot)$ for non-linear block is equivalent to minimizing $U_b^{k+1}(m^{k-1})$ and $L_b^{k+1}(m^{k-1})$, respectively.

Theorem 3. *Minimizing*

$$\iint_{\boldsymbol{x^{k-1}}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}(U_b^{k+1}(\boldsymbol{x^{k-1}})-L_b^{k+1}(\boldsymbol{x^{k-1}}))d\boldsymbol{x^{k-1}}$$

is equivalent to minimizing $U_b^{k+1}(\boldsymbol{m^{k-1}})$ *and* $L_b^{k+1}(\boldsymbol{m^{k-1}})$

The proof of Theorem 3 is as follows.

Proof. First, Minimizing

$$\iint_{\boldsymbol{x^{k-1}}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}(U_b^{k+1}(\boldsymbol{x^{k-1}})-L_b^{k+1}(\boldsymbol{x^{k-1}}))d\boldsymbol{x^{k-1}}$$

is equivalent to minimizing the

$$\iint_{\boldsymbol{x^{k-1}}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}(U_b^{k+1}(\boldsymbol{x^{k-1}})-f^{k-1}(\boldsymbol{x^{k-1}}))d\boldsymbol{x^{k-1}}$$

and

$$\iint_{\boldsymbol{x^{k-1}}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}(f^{k-1}(\boldsymbol{x^{k-1}})-L_b^{k+1}(\boldsymbol{x^{k-1}}))d\boldsymbol{x^{k-1}}$$

, respectively.

Therefore, it is equivalent to minimize the $\iint_{\boldsymbol{x^{k-1}}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}U_b^{k+1}(\boldsymbol{x^{k-1}})d\boldsymbol{x^{k-1}}$ and $\iint_{\boldsymbol{x^{k-1}}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}(-L_b^{k+1}(\boldsymbol{x^{k-1}}))d\boldsymbol{x^{k-1}}$, respectively.

Because $U_b^{k+1}(\boldsymbol{x^{k-1}})$ is a linear combination of $\boldsymbol{x^{k-1}}$. Without loss of generality, we assume $U_b^{k+1}(\boldsymbol{x^{k-1}}) = \sum_{q\in[n_{k-1}]}A_{u,q}^{k+1}x_q^{k-1}+B_u^{k+1}$. Then,

$$\begin{aligned}
&\iint_{\boldsymbol{x}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}U_b^{k+1}(\boldsymbol{x^{k-1}})d\boldsymbol{x}\\
=&\iint_{\boldsymbol{x}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}(\sum_{q\in[n_{k-1}]}A_{u,q}^{k+1}x_q+B_u^{k+1})d\boldsymbol{x}\\
=&\Pi_{i=1}^{n_{k-1}}(u_i^{k-1}-l_i^{k-1})(\sum_{q\in[n_{k-1}]}A_{u,q}^{k+1}\frac{u_q^{k-1}+l_q^{k-1}}{2}+B_u^{k+1})\\
=&\Pi_{i=1}^{n_{k-1}}(u_i^{k-1}-l_i^{k-1})U_b^{k+1}(\boldsymbol{m^{k-1}})
\end{aligned}$$

where $\boldsymbol{m^{k-1}} = (\frac{u_1^{k-1}+l_1^{k-1}}{2},\cdots,\frac{u_{n_{k-1}}^{k-1}+l_{n_{k-1}}^{k-1}}{2})$, because u_q^{k-1}, l_q^{k-1}, $q\in[n_{k-1}]$ are constant, and the minimize target has been transformed into minimizing $U_b^{k+1}(\boldsymbol{m})$.

Therefore, minimizing $\iint_{\boldsymbol{x^{k-1}}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}U_b^{k+1}(\boldsymbol{x^{k-1}})$ is equivalent to minimize $U_b^{k+1}(\boldsymbol{m^{k-1}})$

Similarly, minimizing $-\iint_{\boldsymbol{x^{k-1}}\in[\boldsymbol{l^{k-1}},\boldsymbol{u^{k-1}}]}L_b^{k+1}(\boldsymbol{x^{k-1}})$ is equivalent to minimize $-L_b^{k+1}(\boldsymbol{m^{k-1}})$ □

Based on Theorem 3, we prove Theorem 2 as follows.

Proof. When $l<0\wedge u>0$, the linear bounds of ReLU used in our approach are the same as [4, 37, 49], which are the provable neuron-wise tightest upper linear bounds [37]. It is:

$$u(x)=\frac{u}{u-l}(x-l) \tag{2}$$

First, we prove the upper linear bound is the block-wise tightest. Without loss of generality, we assume ReLU is at the k-th layer. We use u^{k+1}, l^{k+1} and u_M^{k+1}, l_M^{k+1} to denote other linear bounds and our linear bounds for the MaxPool function, respectively. As the slope of the MaxPool linear bounds is always non-negative, the global upper and lower linear bounds of the ReLU+MaxPool block are:

$$\begin{aligned}
&u^{k+1}(x_1^k,\cdots,x_n^k)\\
\le&u^{k+1}(u^k(x_1^{k-1}),\cdots,u^k(x_n^{k-1}))\\
=:&U_b^{k+1}(x_1^{k-1},\cdots,x_n^{k-1})\\
&l^{k+1}(x_1^k,\cdots,x_n^k)\\
\le&l^{k+1}(l^k(x_1^{k-1}),\cdots,l^k(x_n^{k-1}))\\
=:&L_b^{k+1}(x_1^{k-1},\cdots,x_n^{k-1})
\end{aligned}$$

where we use $U_b^{k+1}(\cdot)$ and $L_b^{k+1}(\cdot)$ to denote the global upper and lower linear bounds of the ReLU+MaxPool block, respectively.

If $U_b^{k+1}(m_1^{k-1},\cdots,m_n^{k-1})$ reaches its minimum, the upper linear bound of Theorem 1 are the provable block-wise tightest.

Upper linear bound:

$$\begin{aligned}
&U_b^{k+1}(m_1^{k-1},\cdots,m_n^{k-1})\\
=&u^{k+1}(u^k(m_1^{k-1}),\cdots,u^k(m_n^{k-1}))\\
=&u^{k+1}(\frac{u_1^{k-1}(m_1^{k-1}-l_1^{k-1})}{u_1^{k-1}-l_1^{k-1}},\cdots,\frac{u_n^{k-1}(m_n^{k-1}-l_n^{k-1})}{u_n^{k-1}-l_n^{k-1}})\\
=&u^{k+1}(\frac{u_1^{k-1}(\frac{u_1^{k-1}-l_1^{k-1}}{2})}{u_1^{k-1}-l_1^{k-1}},\cdots,\frac{u_n^{k-1}(\frac{u_1^{k-1}-l_1^{k-1}}{2})}{u_n^{k-1}-l_n^{k-1}})\\
=&u^{k+1}(\frac{u_1^{k-1}}{2},\cdots,\frac{u_n^{k-1}}{2})\\
=&\frac{1}{2}u^{k+1}(u_1^{k-1},\cdots,u_n^{k-1})\\
=&\frac{1}{2}u^{k+1}(u_1^k,\cdots,u_n^k)\\
\ge&\frac{1}{2}max\{u_1^k,\cdots,u_n^k\}\\
=&\frac{1}{2}u_M^{k+1}(u_1^k,\cdots,u_n^k)
\end{aligned}$$

This means $u_M^{k+1}(\cdot)$ is the block-wise tightest. □