Efficient Global Robustness Certification of Neural Networks via Interleaving Twin-Network Encoding

Zhilu Wang*, Chao Huang†, Qi Zhu*
*Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL, USA
Emails: zhilu.wang@u.northwestern.edu, qzhu@northwestern.edu
†Department of Computer Science, University of Liverpool, Liverpool, UK
Email: chao.huang2@liverpool.ac.uk

***Abstract*—The robustness of deep neural networks has received significant interest recently, especially when being deployed in safety-critical systems, as it is important to analyze how sensitive the model output is under input perturbations. While most previous works focused on the *local robustness* property around an input sample, the studies of the *global robustness* property, which bounds the maximum output change under perturbations over the entire input space, are still lacking. In this work, we formulate the global robustness certification for neural networks with ReLU activation functions as a mixed-integer linear programming (MILP) problem, and present an efficient approach to address it. Our approach includes a novel interleaving twin-network encoding scheme, where two copies of the neural network are encoded side-by-side with extra interleaving dependencies added between them, and an over-approximation algorithm leveraging relaxation and refinement techniques to reduce complexity. Experiments demonstrate the timing efficiency of our work when compared with previous global robustness certification methods and the tightness of our over-approximation. A case study of closed-loop control safety verification is conducted, and demonstrates the importance and practicality of our approach for certifying the global robustness of neural networks in safety-critical systems.**

I. Introduction

Deep neural networks (DNNs) are being applied to a wide range of tasks, such as perception, prediction, planning, control, and general decision making. While DNNs often show significant advantages in performance over traditional model-based methods, pressing concerns have been raised on the uncertain behaviors of DNNs under varying inputs, especially for safety-critical systems such as autonomous vehicles and robots. Some of the well-trained DNNs are found to be vulnerable to a small adversarial perturbation on their inputs [1]. The *robustness* metric of DNNs is defined to bound such uncertain behaviors when the input is perturbed. Specifically, the robustness of a DNN represents how much its output varies when its input has a bounded perturbation, where the perturbation can be either from random noises or due to malicious attacks.

Formal methods, such as Satisfiability Modulo Theories (SMT) [2], [3] and Mixed-Integer Linear Programming (MILP) [4], have been used to either find an adversarial example or certify the robustness (i.e., proving no adversarial example exists) around a given input sample, which is considered as *local robustness*. The efficiency is improved by certifying a conservative (sound but not complete) local robustness bound via over-approximated output range analysis methods [5], [6].

In safety-critical systems with DNNs involved, it is important to analyze the DNN robustness for evaluating system safety. As local robustness is defined locally to a given input sample, to evaluate system safety, we will need to apply local robustness certification *during runtime* for each input sample that the system encounters or may encounter. However, the complexity of local robustness certification, even with conservative over-approximation, is too high for runtime execution in most practical systems. Moreover, sometimes it is required to ensure system safety for future time horizon (e.g., to verify there is no collision for the next 3 seconds). As future input samples are unknown, local robustness certification cannot be applied.

Instead, we could address system safety by considering the problem of *global robustness*, which measures the worst-case DNN output change against input perturbation for all possible input samples. The definition of the global robustness is introduced in [2], with a proposed SMT-based tool Reluplex for certifying the exact global robustness. In [7], the global robustness for classification problems is formulated into MILP to find the maximum input perturbation that can preserve the classification result. In [8], an MILP-based exact global robustness certification method is proposed for logic ensemble classifier (a generalized-version of decision tree, not DNN).

A major challenge for exact global robustness certification is its complexity – while it can be run offline to facilitate safety analysis (since it considers the entire input space), the high complexity often makes it intractable even in offline computation. A number of methods have been proposed to tackle the complexity, however they often lack the deterministic guarantees needed for safety verification. For instance, dataset-based global robustness estimation approaches are proposed in [9], [10], which conduct local robustness certification for each sample in the dataset and find the worst case or the average. As the dataset cannot cover all possible input samples, the derived global robustness is just an estimation with no deterministic guarantees. Sampling-based methods [11] randomly sample the DNN inputs, evaluate the local robustness of the random-sampled inputs, and provide a probabilistic global robustness. Similarly, they cannot provide deterministic guarantees. Region-based robustness analyses [11], [12] divide the input space into regions, where all possible inputs in

This work is supported in part by NSF grants 1834701, 1839511, 1724341, 2038853, and ONR grant N00014-19-1-2496.

978-3-9819263-6-1/DATE22/©2022 EDAA

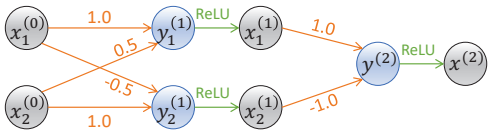

Fig. 1. An example neural network for illustration.

each certified-robust region have the same classification result. However, the number of regions could be extremely large for DNNs with high-dimensional inputs, making it hard to certify all regions and provide deterministic guarantees.

In this work, we propose an efficient certification approach to *over-approximate* the global robustness. The derived global robustness is sound and deterministic, the over-approximation is tight, and the approach can be used effectively for the purpose of safety verification. To achieve this, our approach introduces a novel network encoding structure, namely interleaving twin-network encoding, to compare two copies of the neural network side-by-side under different inputs, with extra interleaving dependencies added between them to improve efficiency. Our approach also includes over-approximation techniques based on network decomposition and LP (linear programming) relaxation, to further reduce the computation complexity. To the best of our knowledge, our approach is the *first global robustness over-approximation method that certifies the robustness among the entire input domain with sound and deterministic guarantee*. Experiments show that our approach is much more efficient and scalable than the exact global robustness methods such as Reluplex [2], with tight over-approximation – our approach can certify DNNs with more than 5k neurons in 5 hours while the exact methods cannot certify DNNs with more than 64 neurons in 1 day. To demonstrate the importance of global robustness and the practical application of our certification approach, a case study is conducted to verify the safety of a closed-loop control system with a vision-based perception DNN in the loop.

II. Efficient Global Robustness Certification

A. Problem Formulation

An n-layer neural network, formulated as $F : \mathbb{R}^{m_0} \to \mathbb{R}^{m_n}$ maps an m_0-dimension input $x^{(0)} \in \mathbb{R}^{m_0}$ into an m_n-dimension output $x^{(n)} \in \mathbb{R}^{m_n}$. In the neural network, the output of each layer i is denoted as $x^{(i)} \in \mathbb{R}^{m_i}$ with dimension m_i, and layer i maps the output from the previous layer $x^{(i-1)}$ into its output by $x^{(i)} = f^{(i)}(x^{(i-1)})$. $f^{(i)}$ is composed with a linear transformation $y^{(i)} = W^{(i)}x^{(i-1)} + b^{(i)}$ (e.g., fully-connected layer, convolution layer, average pooling or normalization), and (optionally) a ReLU activation function. The linear transformation result is denoted as intermediate variable $y^{(i)} \in \mathbb{R}^{m_i}$. The layer output $x^{(i)} = relu(y^{(i)}) = \max(y^{(i)}, 0)$ if it has ReLU activation function, and $x^{(i)} = y^{(i)}$ otherwise. An illustrating example is shown in Fig. 1, which is a 2-layer neural network that maps 2-dimension input $x^{(0)}$ into 1-dimension output $x^{(2)}$ with a 2-neuron hidden layer. For simplicity, the bias $b^{(i)}$ of each layer i is set to 0.

We consider the global robustness over the entire input domain X of the neural network, i.e., $\forall x^{(0)} \in X$. It measures the worst-case output variation when there is a small input perturbation for any possible input sample in the input domain X. Moreover, we focus on the input perturbation that is bounded in the form of L_∞ norm, and have the same global robustness definition as [2], [13].

Definition 1 (Global Robustness)**.** *The j-th output of neural network F is (δ, ε)-globally robust in the input domain X iff*

$$\forall x^{(0)}, \hat{x}^{(0)} \in X, \|\hat{x}^{(0)} - x^{(0)}\|_\infty \leq \delta \implies |\hat{x}_j^{(n)} - x_j^{(n)}| \leq \varepsilon$$

where $x^{(n)} = F(x^{(0)})$ and $\hat{x}^{(n)} = F(\hat{x}^{(0)})$.

In this work, we tackle the problem of *how robust the neural network is, i.e., how small ε can be for a given δ,* formally as:

Problem 1. *For a neural network F, given an input perturbation bound δ, determine the minimal output variation bound ε such that F is guaranteed to be (δ, ε)-globally robust.*

The work in [2] proposes to solve the global robustness problem by encoding two copies of the neural network side by side, as illustrated in the left part of Fig. 2. The two network copies take two separate inputs $x^{(0)}$ and $\hat{x}^{(0)}$ and produce outputs $x^{(n)}$ and $\hat{x}^{(n)}$. As a perturbation of $x^{(0)}$, $\hat{x}^{(0)}$ is restricted by the input perturbation $\Delta x^{(0)} = \hat{x}^{(0)} - x^{(0)}$, where $\|\Delta x^{(0)}\|_\infty \leq \delta$. The output variation bound ε is the bound of the output distance $\Delta x^{(n)} = \hat{x}^{(n)} - x^{(n)}$. Under this encoding, Problem 1 can be formulated as an optimization problem:

$$\begin{aligned} \varepsilon := \max \quad & |\hat{x}^{(n)} - x^{(n)}| \\ \text{s.t.} \quad & \hat{x}^{(n)} = F(\hat{x}^{(0)}),\ x^{(n)} = F(x^{(0)}) \\ & \hat{x}^{(0)}, x^{(0)} \in X,\ \|\hat{x}^{(0)} - x^{(0)}\|_\infty < \delta \end{aligned} \tag{1}$$

Eq. (1) can be solved by mixed-integer linear programming (MILP), where the neural network F can be decomposed into a series of neuron-wise dependency. For the j-th neuron of layer i with ReLU activation, we have:

$$y_j^{(i)} = W_j^{(i)} x_{i-1} + b_j^{(i)}, \quad x_j^{(i)} = \max(y_j^{(i)}, 0) \tag{2}$$

$$\hat{y}_j^{(i)} = W_j^{(i)} \hat{x}^{(i-1)} + b_j^{(i)}, \quad \hat{x}_j^{(i)} = \max(\hat{y}_j^{(i)}, 0) \tag{3}$$

The $\max(x, 0)$ function for each ReLU activation can be linearized by Big-M method [7] with the introduction of a new integer (binary) variable to indication whether $y < 0$.

The complexity to solve this MILP is exponential to the number of ReLU neurons, making it too complex for most practical neural networks. In this work, to overcome this challenge, we present 1) a new interleaving twin-network encoding (ITNE) scheme that adds extra interleaving dependencies between the two copies of the neural network, and 2) two approximation techniques leveraging the new encoding scheme, namely *network decomposition (ND)* and *LP relaxation (LPR)*, to efficiently find an over-approximated sub-optimal solution $\bar{\varepsilon}$ of the output variation bound ε, where $\bar{\varepsilon} \geq \varepsilon$.

B. Interleaving Twin-Network Encoding

We call the neural network encoding scheme introduced in [2], shown in the left side of Fig. 2 with only input and output layers connected, the basic twin-network encoding (BTNE). In contrast, the new network encoding scheme we designed to

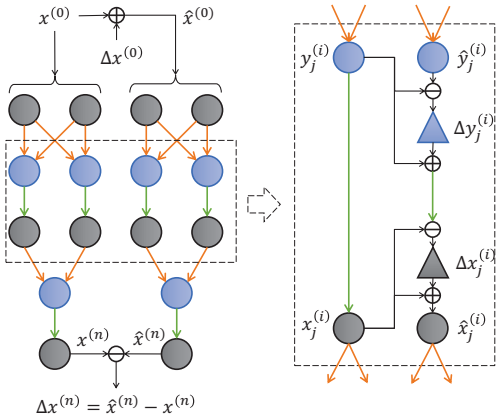

Fig. 2. Left: The basic twin-network encoding (BTNE) for global robustness certification. Right: the neuron-level interleaving twin-network encoding (ITNE) built upon the basic structure, where the hidden layer neurons are connected between the two copies with distance variables $\Delta y_j^{(i)}$ and $\Delta x_j^{(i)}$.

enable the over-approximation techniques for global robustness certification is shown in the right side of Fig. 2 and called the interleaving twin-network encoding (ITNE).

In ITNE, besides the connections between the input and output layers, interleaving connections are added for all hidden layer neurons between the two network copies. Specifically, for the j-th neuron of layer i, two variables, $\Delta y_j^{(i)}$ and $\Delta x_j^{(i)}$, are added to encode the distance of $y_j^{(i)}$ and $x_j^{(i)}$ between the two copies, where

$$\Delta y_j^{(i)} = \hat{y}_j^{(i)} - y_j^{(i)},$$

$$\Delta x_j^{(i)} = \hat{x}_j^{(i)} - x_j^{(i)} = relu(y_j^{(i)} + \Delta y_j^{(i)}) - relu(y_j^{(i)}).$$

These additional distance variables reflect how different the two network copies are during the forward propagation. And the non-linear mapping from $\hat{y}_j^{(i)}$ to $\hat{x}_j^{(i)}$ is substituted by the mapping between $\Delta y_j^{(i)}$ and $\Delta x_j^{(i)}$. These changes enable the usage of the over-approximation techniques introduced below.

C. Over-Approximation Techniques

Leveraging ITNE, we design two over-approximation techniques, network decomposition (ND) and LP relaxation (LPR), to improve the global robustness certification efficiency. The two techniques are inspired by the local robustness certification work in [6]. However, there are major differences between global and local robustness certification, given that the former considers the entire input space X while the latter only considers a given input $x^{(0)} \in X$, and our ND and LPR techniques are specifically designed for global robustness.

a) ITNE-based network decomposition (ND): The main idea of ND is to divide a neural network into sub-networks and decompose the entire optimization problem into smaller problems. The input of a sub-network is either the network input or the output of other sub-networks. Given the range of the input of a sub-network, the range of its output can be derived by solving an optimization problem of the sub-network, which can then be used as the input range of the next sub-network. As the complexity of MILP is exponential to the neural network size, using ND can significantly reduce the complexity of the optimization problem. For global robustness certification, we also consider two copies of each sub-network. However, instead of finding the output range of each sub-network copy, we look for the range of one sub-network copy and the range of the output distance, based on the ITNE.

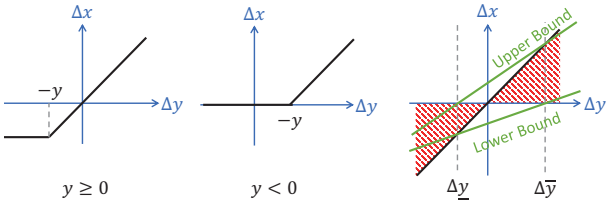

Fig. 3. Left: ReLU distance relation when $y \geq 0$; Middle: ReLU distance relation when $y < 0$; Right: LP-relaxation of ReLU distance relation. The ReLU distance relation for $\forall y \in \mathbb{R}$ lays in the shadowed area. Within the distance range $\Delta y \in [\Delta \underline{y}, \Delta \overline{y}]$, Δx is bounded by the lower and upper bounds.

More specifically, a w-layer sub-network with input $x^{(i-w)}$ and output $x_j^{(i)}$ $(i \geq w)$ is denoted as $F_w(x_j^{(i)})$. For any variable v, its range is denoted as $\overline{v} = [\underline{v}, \overline{v}]$. Under ITNE, for each sub-network $F_w(x_j^{(i)})$, given the input range $\overline{x}^{(i-w)}$ and input distance range $\Delta \overline{x}^{(i-w)}$, we can derive the output range $\overline{x}_j^{(i)}$ and output distance range $\Delta \overline{x}_j^{(i)}$ by solving the optimization problem in Eq. (1) for this sub-network.

b) ITNE-based LP relaxation (LPR): The idea of LPR is to relax the ReLU relation $x = \max(0, y)$ into linear constraints given the range of y as $[\underline{y}, \overline{y}]$. When $\overline{y} \leq 0$ or $\underline{y} \geq 0$, the ReLU relation degenerates to $x = 0$ or $x = y$. Otherwise, ReLU relation can be relaxed by three linear inequations:

$$x \geq 0, \quad x \geq y, \quad x \leq \frac{\overline{y}(y - \underline{y})}{\overline{y} - \underline{y}} \tag{4}$$

The ReLU relations of a neuron in the two network copies are formulated in Eqs. (2) and (3). In our work, instead of relaxing the ReLU relations in two network copies separately, we relax Eq. (2) by the original LPR in Eq. (4), and relax the ReLU distance relation based on ITNE:

$$\Delta x = relu(y + \Delta y) - relu(y). \tag{5}$$

For a given y, the relation between Δx and Δy is shown in the first two plots of Fig. 3. Thus, the shadowed area in the third plot of Fig. 3 can cover all possible $(\Delta x, \Delta y)$ mappings for $\forall y \in \mathbb{R}$. Given $\Delta y \in [\Delta \underline{y}, \Delta \overline{y}]$, the relation between Δx and Δy can be bounded by the linear lower and upper bounds as shown in the third plot of Fig. 3. Formally, the bound is

$$\frac{l(u - \Delta y)}{u - l} \leq \Delta x \leq \frac{u(\Delta y - l)}{u - l}, \tag{6}$$

where $l = \min(0, \Delta \underline{y})$ and $u = \max(0, \Delta \overline{y})$.

D. Illustrating example

Consider the example neural network in Fig. 1. Assume that the input perturbation bound as $\delta = 0.1$ and the input

domain as $x^{(0)} \in [-1,1]^2$. The example neural network can be decomposed into three sub-networks:

$$x_1^{(1)} = relu(y_1^{(1)}) = relu(x_1^{(0)} + 0.5x_2^{(0)})$$
$$x_2^{(1)} = relu(y_2^{(1)}) = relu(-0.5x_1^{(0)} + x_2^{(0)})$$
$$x^{(2)} = relu(y^{(2)}) = relu(x_1^{(1)} - x_2^{(1)}).$$

For local robustness, with input $x^{(0)} = [0,0]$, the certification processes for different techniques are shown in Fig. 4. The exact bound of $\hat{x}^{(2)}$ is derived by solving the MILP problem for the entire network. ND solves the MILP problems of the first two sub-networks to derive the bound of $\hat{x}^{(1)}$ and then derive the bound of $\hat{x}^{(2)}$ by solving the MILP problem of the third sub-network. For LPR, given $\hat{y}_1^{(1)}, \hat{y}_2^{(1)}, \hat{y}^{(2)} \in [-0.15, 0.15]$, all ReLU relations can be relaxed based on Eq. (4), and the bound of $\hat{x}^{(2)}$ can be derived by solving the relaxed LP problem. As we can see, ND and LPR can provide tight (1.2x and 1.15x) over-approximations for local robustness.

For global robustness, under BTNE, there are no distance variables for hidden neurons. Thus, ND and LPR can only be applied to each individual network copy. ND will only find the range of $x^{(1)}$ and $\hat{x}^{(1)}$ and for the third sub-network, the distance information between two network copies is lost, resulting in a 7.5x over-approximation of the range of $\Delta x^{(2)}$. The LPR is based on the bounds $y^{(1)}, \hat{y}^{(1)} \in [-1.5, 1.5]^2$, and $y^{(2)}, \hat{y}^{(2)} \in [-1.5, 1.5]$, resulting in a 10.9x over-approximation. On the other hand, under ITNE, the ITNE-based ND can first find the range of $x^{(1)}$ and $\Delta x^{(1)}$ and then derive the range of $\Delta x^{(2)}$, which is only 1.5x of the exact one. Given $\Delta y^{(1)} \in [-0.15, 0.15]^2$ and $\Delta y^{(2)} \in [-0.3, 0.3]$, the ITNE-based LPR derives a tight 1.38x over-approximation of the range of $\Delta x^{(2)}$. This shows that combing ITNE with ND and LPR significantly improves the approximation tightness over BTNE.

E. Efficient Global Robustness Over-Approximation Algorithm

Finally, we design our global robustness certification algorithm by leveraging the ITNE encoding as well as the ND and LPR techniques, as shown in Algorithm 1. Note that the MILP problems for sub-networks from ND are relaxed by LPR. The pre-assigned window size W is defined as the desired depth of sub-networks. For LPR, besides the range of the input and distance, $\underline{y}^{(i-k)}$ and $\Delta\underline{y}^{(i-k)}$ for $\forall k \in [0, w-1]$ are also needed. To evaluate $\underline{y}_j^{(i)}$ and $\Delta\underline{y}_j^{(i)}$, we can consider a w-layer sub-network with $y_j^{(i)}$ as the output (i.e., no ReLU activation at the output layer), denoted as $F_w(y_j^{(i)})$. Formally, we have two types of sub-network optimization problems: $LpRelaxY$ and $LpRelaxX$, which is respectively encoded (lines 7 and 9) as ITNE of sub-network $F_w(y_j^{(i)})$ and $F_w(x_j^{(i)})$ (decomposed from network F in lines 6 and 9). Both problems require the sub-network input range $\underline{x}^{(i-w)}$, $\Delta\underline{x}^{(i-w)}$ and the ranges for hidden neurons, i.e., $\underline{y}^{(i-k)}$ and $\Delta\underline{y}^{(i-k)}, \forall k \in [1, w-1]$ as prerequisites, while $LpRelaxX$ (line 11) also needs $\underline{y}_j^{(i)}, \Delta\underline{y}_j^{(i)}$, which is derived from $LpRelaxY$ (line 8). Therefore, the ranges need to be evaluated layer by layer, by treating these hidden layer neurons as outputs of sub-networks $F_w(y_j^{(i)})$ and $F_w(x_j^{(i)})$. In such a way, when evaluating the ranges of certain layer, all ranges of previous layers have been derived.

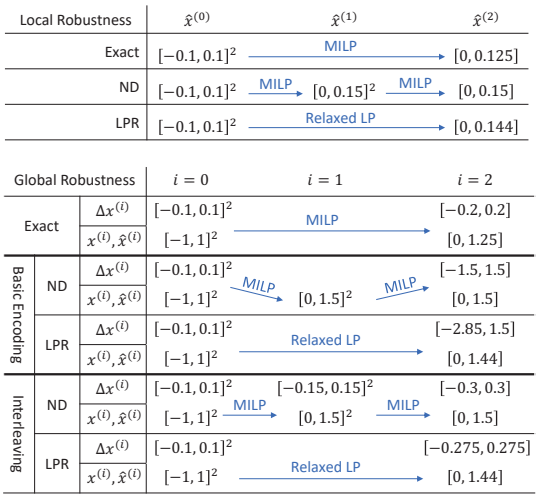

Fig. 4. Illustrating example: The local and global robustness certification processes of exact MILP, network decomposition (ND) and LP relaxation (LPR). For global robustness, the ND and LPR for both basic and interleaving twin-network encoding (i.e., BTNE and ITNE) are illustrated.

Algorithm 1 Global Robustness Certification Algorithm

Input: Neural Network F, window size W, refine param r
Input: input domain X, input perturbation bound δ
Output: output variation bound $\bar{\varepsilon}$

1: $\overline{x}^{(0)} = X$
2: $\Delta\overline{x}^{(i)} = [-\delta, \delta]^{m_0}$
3: **for** $i = 1 : n$ **do**
4: $\quad w = \max(i, W)$
5: $\quad$ **for** $j = 1 : m_i$ **do**
6: $\quad\quad F_w(y_j^{(i)}) \leftarrow NetDecompose(F, y_j^{(i)}, w)$
7: $\quad\quad e_y = InterleaveTwinNetEncode(F_w(y_j^{(i)}))$
8: $\quad\quad (\underline{y}_j^{(i)}, \Delta\underline{y}_j^{(i)}) \leftarrow LpRelaxY(e_y, r)$
9: $\quad\quad F_w(x_j^{(i)}) \leftarrow NetDecompose(F, x_j^{(i)}, w)$
10: $\quad\quad e_x = InterleaveTwinNetEncode(F_w(x_j^{(i)}))$
11: $\quad\quad (\underline{x}_j^{(i)}, \Delta\underline{x}_j^{(i)}) \leftarrow LpRelaxX(e_x, r)$
12: $\quad$ **end for**
13: **end for**
14: $\bar{\varepsilon} \leftarrow \max(|\Delta\underline{x}^{(n)}|, |\Delta\overline{x}^{(n)}|)$

Selective Refinement: While LPR can remove all integer variables in the MILP formulation to reduce the complexity, such extreme over-approximation may be too inaccurate. Thus, we try to selectively refine a limited number of neurons, by *not* relaxing their ReLU relations. This is similar to the layer-level refinement idea in [6], but with a focus on global robustness. Specifically, a score of LPR is evaluated for each neuron as the worst-case inaccuracy, and the top r neurons will be refined. The score is defined as the maximum distance between the lower and upper bound of LPR, i.e., the score is $-\overline{y}\underline{y}/(\overline{y} - \underline{y})$

TABLE I
NEURAL NETWORK SETTING AND EXPERIMENTAL RESULTS.

Dataset	ID	Neurons	t_R	t_M	t_{our}	ε	$\overline{\varepsilon}$ (ours)
Auto MPG	1	8	2s	0.1s	0.3s	0.0583	0.0657
	2	12	130s	0.2s	0.4s	0.0527	0.0722
	3	16	8h	0.8s	1s	0.0496	0.0653
	4	32	$>$ 24h	74s	5s	0.0481	0.0673

Dataset	ID	Neurons	Layers	t_{our}	$\underline{\varepsilon}$	$\overline{\varepsilon}$ (ours)
Auto MPG	5	64	FC:3	50s	0.0452	0.0731
MNIST	6	1416	Conv:1 FC:2	4.8h	0.347	0.578
					0.300	0.572
	7	3872	Conv:2 FC:2	3.3h	0.453	0.874
					0.420	0.723
	8	5824	Conv:3 FC:2	3.5h	0.519	1.521
					0.407	1.175

In Table I, 'Layers' are the type and number of layers; 'Neurons' is the total number of hidden neurons; t_R, t_M, t_{our} are the certification time of exact MILP, Reluplex, and our approach, respectively; ε, $\underline{\varepsilon}$, $\overline{\varepsilon}$ are the exact (derived by Reluplex or MILP), under-approximated (derived by projected gradient descent (PGD) among dataset), and over-approximated (derived by our approach) output variation bounds, respectively.

for Eq. (4), and $\max(|\Delta\underline{y}|, |\Delta\overline{y}|)$ for Eq. (6). The number of neurons for refinement r is a parameter that can be set in LPR.

Complexity Analysis: MILP's complexity is polynomial to the number of variables and exponential to the number of integer variables. In our approach, for each MILP, the number of variables is linear to the number of neurons in the sub-network, and the number of integer variables is linear to the number of neurons selected for refinement. The number of MILPs to solve is linear to the total number of neurons.

III. EXPERIMENTAL RESULTS

We first evaluate our algorithm on various DNNs and compare its results with exact global robustness (when available) and an under-approximated global robustness. Then, we demonstrate the application of our approach in a case study of safety verification for a vision-based robotic control system, and show the importance of efficient global robustness certification for safety-critical systems that involve neural networks.

We implement our efficient global robustness certification algorithm in Python. All MILP/LP problems in the algorithm are solved by Gurobi [14]. All the experiments are conducted on a platform with a 4-core 3.6 GHz Intel Xeon CPU and 16 GB memory. All neural networks are modeled in Tensorflow.

A. Comparison with Other Methods

We conduct experiments on a set of DNNs with different sizes. The DNNs are trained on two datasets: the Auto MPG dataset [15] for vehicle fuel consumption prediction, and the MNIST handwritten digits classification dataset [16]. Besides the fully-connected (FC) output layer, the Auto MPG DNNs contain 2 FC hidden layers with the number of hidden neurons ranging from 8 to 64. MNIST DNNs contain 1 to 3 convolutional layers followed by 1 FC hidden layer, with 1472 to 7176 hidden neurons. We certify the output variation bound ε based on the input perturbation bound δ, where $\delta = 0.001$ for Auto MPG DNNs and $\delta = 2/255$ for MNIST DNNs.

The network settings and the comparison among different approaches are shown in Table I. For our approach, window size $W = 2$ for Auto MPG DNNs, and $W = 3$ for MNIST DNNs. Half neurons are refined in Auto MPG DNNs and 30 neurons in each layer are refined in MNIST DNNs. We present 2 outputs of MNIST (out of 10) in Table I, and others show similar differences between different methods.

For small networks (DNN-1 to DNN-5), we compare our over-approximated output variation bound $\overline{\varepsilon}$ with the exact bound ε solved by Reluplex [2] (latest version that is integrated in the tool Marabou [17] is used) and the MILP encoding in Eq. (1). We can see that the runtime of Reluplex t_R and MILP t_M quickly increases with respect to neural network size. None of them can get a solution within 24 hours for DNN-5 (with only 64 hidden neurons). From DNN-1 to DNN-4, our algorithm can derive an over-approximated global robustness with much slower runtime increase, while only have about 13% to 40% over-approximation over the exact global robustness.

Starting from DNN-5, the two exact certification methods cannot find a solution in reasonable time. There is also no other work in the literature that can derive a sound and deterministic global robustness. To assess how good our over-approximated results are for larger networks, we evaluate an under-approximation of the global robustness, inspired by [9]. Specifically, for each data sample in the dataset, we leverage projected gradient descent (PGD) [18] to look for an adversarial example in the input perturbation bound that maximizes the output variation, and treat the maximal output variation among the entire dataset as an *under-approximated* output variation bound $\underline{\varepsilon}$. The exact global robustness should be between the under-approximation $\underline{\varepsilon}$ derived from this dataset-wise PGD, and the over-approximation $\overline{\varepsilon}$ derived by our certification algorithm. The experiments from DNN-6 to DDN-8 demonstrate that **our method can provide meaningful over-approximation (less than 3x of the under-approximation) for DNNs with more than 5000 hidden neurons within 5 hours**.

B. Case Study on Control Safety Verification

For control systems that use neural networks for perception, a critical and yet challenging question is whether the system can remain safe when there is perturbation/disturbance to the perception neural network input. In [13], the authors formulate this as a design-time safety assurance problem based on global robustness (although did not provide a solution), i.e., if a global robustness bound ϵ can be derived for an input perturbation bound δ, such ϵ can be viewed as the state estimation error bound in control and leveraged for control safety verification.

In this section, leveraging our approach for bounding global robustness, we conduct a case study for safety verification of control systems that use neural networks for perception. We consider an advanced cruise control (ACC) scenario where an ego vehicle is following a reference vehicle. The ego vehicle is equipped with an RGB camera, and a DNN is designed to estimate the distance from the reference vehicle based on the camera image. We assume that the captured image may be slightly perturbed. A feedback controller takes the estimated distance, along with the ego vehicle speed, to generate the control input, which is the acceleration of the ego vehicle.

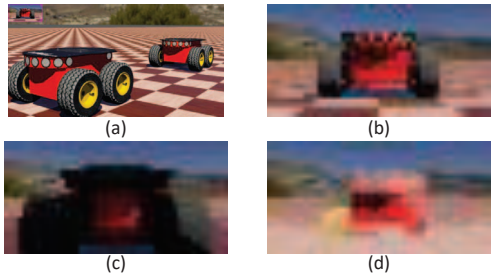

Fig. 5. (a) Webots simulation for case study, where ego vehicle (left) follows reference vehicle (right); (b) An example image captured by ego vehicle; (c) Lower bound of DNN input space; (d) Upper bound of DNN input space.

We model this example in the tool Webots [19] (Fig. 5). The ego vehicle is safe if distance $d \in [0.5, 1.9]$ and speed $v_e \in [0.1, 0.7]$. The reference vehicle speed v_r is randomly adjusted within $[0.2, 0.6]$. The sampling period is 100ms. The camera takes RGB images with resolution 24×48. A 5-layer (3 convolutional and 2 FC) DNN is trained with 100k pre-captured images. The system dynamics is modeled as:

$$x[k+1] = \begin{bmatrix} 1 & -0.1 \\ 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} -0.005 \\ 0.1 \end{bmatrix} u[k] + \begin{bmatrix} 1 \\ 0 \end{bmatrix} w_1[k] + w_2[k]$$

where the system state $x = [d - 1.2, v_e - 0.4]^\top$ contains the normalized distance and speed. $w_1 = 0.4 - v_r$ is the external disturbance, where v_r is bounded within $[0.2, 0.6]$ as previously mentioned. $w_2 = [w_d, w_v]^\top$ is the inaccuracy of this system dynamics model. The control input follows the feedback control law $u = K\hat{x}$, where $K = [0.3617, -0.8582]$. $\hat{x}$ is the estimated system state, and the state estimation error is $\Delta x = \hat{x} - x$. The estimation error of ego vehicle speed v_e is assumed as 0. The bound of w_d, w_v are profiled based on simulations, and bounded as $|w_d| \leq 5e-4$ and $|w_v| \leq 3e-5$. The estimation error for distance contains two parts $\Delta d = \Delta d_1 + \Delta d_2$, where Δd_1 is the error coming from the DNN model inaccuracy, while Δd_2 is the output variation caused by input perturbation.

The DNN model inaccuracy Δd_1 is determined by the worst-case model inaccuracy among the dataset and set as $|\Delta d_1| \leq 0.0730$. The DNN output variation Δd_2 is the focus of this study and can be bounded by our global robustness certification algorithm. Specifically, the input perturbation in this study is bounded by $\delta = 2/255$ (Fig. 5 illustrates the upper and lower bound of each pixel for the input space). Then, using our approach, we can derive a certified output variation bound $|\Delta d_2| \leq \bar{\epsilon} = 0.0568$. Combined with Δd_1, we have $|\Delta d| \leq 0.0730 + 0.0568 = 0.1298$.

With bounds on Δd and other variables, the vehicle control safety can be verified based on the computation of control invariant set, similarly as in [20]. Through such invariant set based verification, we find that as long as the distance estimation error Δd is within $[-0.14, 0.14]$, the system will always be safe. Since our global robustness certification bounds $\Delta d \leq 0.1298$, we can assert that our ACC system is safe with this DNN design under the assumed perturbation bound.

We also deploy our DNN model in Webots and add adversarial perturbation on the input images by Fast Gradient Sign Method (FGSM) [21]. With more than 1000 minutes of simulation, we find that when the perturbation bound is $\delta = 2/255$ as assumed, the distance estimation error Δd never exceeds the $[-0.14, 0.14]$ bound and the system is always safe (as expected since it was formally proven). When the perturbation bound δ is increased to $5/255$, Δd sometimes exceeds the bound, although unsafe system state is not observed. When δ is increased to $10/255$, about 17% of the simulations encounter unsafe states. This shows the impact of input perturbation on system safety and the importance of our global robustness analysis.

IV. Conclusion

We present an efficient certification algorithm to provide a sound and deterministic global robustness for neural networks with ReLU activation. Experiments demonstrate that our approach is much more efficient and scalable than exact global robustness certification approaches while providing tight over-approximation, and a case study further demonstrates the application of our approach in safety-critical systems. Our future work will continue improving the approach's efficiency, possibly via parallelization on multi-core platforms, to enable its usage for larger-size perception neural networks.

References

[1] B. Biggio, I. Corona, D. Maiorca *et al.*, "Evasion attacks against machine learning at test time," in *ECML PKDD*, 2013, pp. 387–402.
[2] G. Katz, C. Barrett, D. L. Dill *et al.*, "Reluplex: An efficient smt solver for verifying deep neural networks," in *CAV*, 2017, pp. 97–117.
[3] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *CAV*, 2017, pp. 3–29.
[4] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *arXiv*, 2017.
[5] G. Singh, T. Gehr, M. Püschel *et al.*, "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, Jan. 2019.
[6] C. Huang, J. Fan, X. Chen *et al.*, "Divide and slide: Layer-wise refinement for output range analysis of deep neural networks," *TCAD*, vol. 39, 2020.
[7] C.-H. Cheng, G. Nührenberg, and H. Ruess, "Maximum resilience of artificial neural networks," in *ATVA*, 2017, pp. 251–268.
[8] Y. Chen, S. Wang, Y. Qin *et al.*, "Learning security classifiers with verified global robustness properties," 2021. [Online]. Available: https://arxiv.org/abs/2105.11363
[9] W. Ruan, M. Wu, Y. Sun *et al.*, "Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance," in *IJCAI*, 2019.
[10] O. Bastani, Y. Ioannou, L. Lampropoulos *et al.*, "Measuring neural net robustness with constraints," in *NeurIPS*, vol. 29, 2016.
[11] R. Mangal, A. V. Nori, and A. Orso, "Robustness of neural networks: A probabilistic and practical approach," in *ICSE-NIER*, 2019, pp. 93–96.
[12] D. Gopinath, G. Katz, C. S. Păsăreanu *et al.*, "Deepsafe: A data-driven approach for assessing robustness of neural networks," in *ATVA*, 2018.
[13] Z. Wang, C. Huang, Y. Wang *et al.*, "Bounding perception neural network uncertainty for safe control of autonomous systems," in *DATE*, 2021.
[14] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com
[15] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[16] Y. Lecun, L. Bottou, Y. Bengio *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, 1998.
[17] G. Katz, D. A. Huang, D. Ibeling *et al.*, "The marabou framework for verification and analysis of deep neural networks," in *CAV*, 2019.
[18] A. Madry, A. Makelov, and L. S. andothers, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.
[19] Webots, "Commercial mobile robot simulation software." [Online]. Available: http://www.cyberbotics.com
[20] C. Huang, S. Xu, Z. Wang *et al.*, "Opportunistic intermittent control with safety guarantees for autonomous systems," in *DAC*, 2020, pp. 1–6.
[21] I. J. Goodfellow, J. Shlens *et al.*, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.