Reachability analysis of neural networks using mixed monotonicity

Pierre-Jean Meyer

▶ **To cite this version:**

Pierre-Jean Meyer. Reachability analysis of neural networks using mixed monotonicity. IEEE Control Systems Letters, 2022, 6, pp3068-3073. 10.1109/LCSYS.2022.3182547 . hal-03708688

HAL Id: hal-03708688
https://hal.science/hal-03708688v1

Submitted on 29 Jun 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reachability analysis of neural networks using mixed monotonicity

Pierre-Jean Meyer

***Abstract*— This paper presents a new reachability analysis approach to compute interval over-approximations of the output set of feedforward neural networks with input uncertainty. We adapt to neural networks an existing mixed-monotonicity method for the reachability analysis of dynamical systems and apply it to each partial network within the main network. This ensures that the intersection of the obtained results is the tightest interval over-approximation of the output of each layer that can be obtained using mixed-monotonicity on any partial network decomposition. Unlike other tools in the literature focusing on small classes of piecewise-affine or monotone activation functions, the main strength of our approach is its generality: it can handle neural networks with any Lipschitz-continuous activation function. In addition, the simplicity of our framework allows users to very easily add unimplemented activation functions, by simply providing the function, its derivative and the global argmin and argmax of the derivative. Our algorithm is compared to five other interval-based tools (Interval Bound Propagation, RELUVAL, NEURIFY, VERINET, CROWN) on both existing benchmarks and two sets of small and large randomly generated networks for four activation functions (ReLU, TanH, ELU, SiLU).**

***Index Terms*— Neural network, uncertain systems**

I. INTRODUCTION

ARTIFICIAL intelligence and particularly neural networks are quickly spreading in various fields including safety-critical applications such as autonomous driving [1]. With it comes a growing need for replacing verification methods based on statistical testing [2] by more formal methods able to provide safety guarantees of the satisfaction of desired properties on the output of the neural network [3]. Such methods are also useful for the verification of a closed-loop system with a neural network controller [4], [5].

When focusing on isolated neural networks, the rapidly growing field of formal verification has been categorized into three main types of objectives [3]: *counter-example result* to find an input-output pair that violates the desired property; *adversarial result* to determine the maximum allowed disturbance that can be applied to a nominal input while preserving the properties of the nominal output; *reachability result* to evaluate (exactly or through over-approximations) the set of all output values reachable by the network when given a bounded input set. While the first two types of results often rely on solving optimization problems [3], [6], reachability results (which are considered in this paper) naturally lean toward developing new reachability analysis approaches to be applied layer by layer in the network. The current reachability methods in the literature consider various set representations to over-approximate the network's output set with static intervals [5], symbolic interval equations and linear relaxations of activation functions (RELUVAL [7], NEURIFY [8], VERINET [9], CROWN [10]) or polyhedra and zonotopes in the tool libraries NNV [11] and ERAN [12].

The main weakness of existing neural network verifiers is that most of them are limited to a very small class of activation functions. The large majority of verifiers only handles piecewise-affine activation to focus on the most popular ReLU function [7], [8], [13], [14]. A handful of tools cover other popular activations, such as S-shaped functions sigmoid and hyperbolic tangent [9], [11], [12]. Several other tools claiming to handle general activation functions are either implicitly restricted to monotone functions in their theory or implementation [15], [16], or their claimed generality rather refers to a mere compatibility with general activation functions but they require the user to provide their own implementation of how to handle these new functions [10].

Although ReLU or sigmoid networks often perform well, the focus on these activation functions in the literature is also strongly tied to the fact that most current optimization and verification tools cannot handle more general activations. Proposing a new approach to deal with these general activation functions is thus an important step towards opening the field of formal verification of neural networks to new non-monotone activations that have already been shown to outperform classical ReLU or sigmoid networks (see e.g. Swish/SiLU [17], Mish [18], PFLU and FPFLU [19]).

Contributions: This paper presents a novel method for the reachability analysis of feedforward neural networks with general activation functions. The proposed approach adapts the existing mixed-monotonicity reachability method for dynamical systems [20] to be applicable to neural networks so that we can obtain an interval over-approximation of the set of all the network's outputs that can be reached from a given input set. Since our reachability method is also applicable to any partial network within the main neural network and always returns a sound over-approximation of the last layer's output set, intersecting the interval over-approximations from several partial networks ending at the same layer can only tighten the approximation while preserving the soundness of the results. To take full advantage of this, we propose an algorithm that applies our new mixed-monotonicity reachability method

Pierre-Jean Meyer is with Univ Gustave Eiffel, COSYS-ESTAS, F-59666 Villeneuve d'Ascq, France
pierre-jean.meyer@univ-eiffel.fr

to all $L(L+1)/2$ partial networks contained within the considered L-layer network. This ensures that the resulting interval over-approximation is the tightest output bound of the network obtainable using mixed-monotonicity reachability analysis on any partial network decomposition, at the cost of a computational complexity of $O(L^2)$.

Mixed-monotonicity reachability analysis is applicable to any system whose Jacobian matrix is bounded on any bounded input set [20]. In the case of neural networks combining linear transformations and nonlinear activation functions, the above requirement implies that our approach is applicable to any network with Lipschitz-continuous activation function. Since extracting the bounds of the Jacobian matrix of a system (or of the derivative of an activation function in our case) is not always straightforward, for the sake of self-containment of the paper we provide a method to automatically obtain these bounds for a (still very general) sub-class of activation functions whose derivative can be defined as a 3-piece piecewise-monotone function. Apart from the binary step and Gaussian function, all activation functions we could find in the literature (including all non-monotone functions reviewed in [19]) belong to this class.

Although most of the tools cited above provide a two-part verification framework for neural networks (one reachability part to compute output bounds of the network, and one part to iteratively refine the network's domain until a definitive answer can be given to the verification problem), this paper focuses on providing a novel method for the first reachability step only. The proposed approach can thus be seen either as a preliminary step towards the development of a larger verification framework for neural networks when combined with an iterative splitting of the input domain, or as a stand-alone tool for the reachability analysis of neural networks as is used for example in the analysis of closed-loop systems with a sample-and-hold neural network controller [5].

In summary, the main contributions of this paper are:

- a novel approach to soundly bound the output of neural networks using mixed-monotonicity reachability;
- a method that is compatible with any Lipschitz-continuous activation function;
- a framework that allows to easily add new activation functions (the user only needs to provide the activation function, its derivative, and the global $\arg\min$ and $\arg\max$ of the derivative).

The paper is organized as follows. Section II defines the considered problem and provides useful preliminaries for the main algorithm, such as the definition of mixed-monotonicity reachability for a neural network, and how to automatically obtain local bounds on the activation function derivative. Section III presents the main algorithm applying mixed-monotonicity reachability to all partial networks within the given network. Finally, Section IV compares our novel reachability approach to the bounding methods of 5 other interval-based optimization-free tools [5], [7], [8], [9], [10] on both existing benchmarks [21] and randomly generated networks, to highlight the complementarity with these tools and the cases when our approach outperforms them.

II. Preliminaries

Given $\underline{x}, \overline{x} \in \mathbb{R}^n$ with $\underline{x} \leq \overline{x}$, the interval $[\underline{x}, \overline{x}] \subseteq \mathbb{R}^n$ is the set $\{x \in \mathbb{R}^n \mid \forall i \in \{1, \dots, n\},\ \underline{x}_i \leq x_i \leq \overline{x}_i\}$.

A. Problem definition

Consider the L-layer feedforward neural network

$$x^{(i)} = \Phi(W^{(i)} x^{(i-1)} + b^{(i)}),\ \forall i \in \{1, \dots, L\} \qquad (1)$$

where $x^{(i)} \in \mathbb{R}^{n_i}$ is the output vector of layer i, $x^{(0)}$ and $x^{(L)}$ being the input and output of the neural network, respectively. We assume that the network is pre-trained and all weight matrices $W^{(i)} \in \mathbb{R}^{n_i \times n_{i-1}}$ and bias vectors $b^{(i)} \in \mathbb{R}^{n_i}$ are known. The function Φ is defined as the componentwise application of a scalar and Lipschitz-continuous activation function. For simplicity of presentation, the activation function Φ is assumed to be identical for all layers, but note that the proposed approach in this paper is compatible with having different activation functions for each layer (or even for each node of the same layer). In the literature, the activation function of the last layer is often omitted since a monotone activation (the most commonly used) would only change the output values but not their relative comparison (which is considered for classification problems). Since our approach introduces the ability to handle non-monotone activation, our network definition (1) needs to allow for the activation of all layers.

Some verification problems on neural networks aim to measure the robustness of the network with respect to input variations. Since the output set of the network cannot always be computed exactly, we rely on over-approximating it by a simpler set representation, such as an interval. We can then solve the verification problem using this interval: if the desired property is satisfied on the over-approximation, then it is also satisfied on the real output set of the network. In this paper, we focus on the problem of computing an interval over-approximation of the output set of the network when its input is taken in a known bounded set, as formalized below.

Problem 1: Given the L-layer neural network (1) and the interval input set $[\underline{x^0}, \overline{x^0}] \subseteq \mathbb{R}^{n_0}$, find an interval $[\underline{x^L}, \overline{x^L}] \subseteq \mathbb{R}^{n_L}$ over-approximating the output set of (1):

$$\{x^{(L)} \mid x^{(0)} \in [\underline{x^0}, \overline{x^0}]\} \subseteq [\underline{x^L}, \overline{x^L}].$$

Naturally, our secondary objective is to ensure that the computed interval over-approximation is as tight as possible in order to minimize the number of false negative results in the subsequent verification process.

B. Mixed-monotonicity reachability

In this paper, we solve Problem 1 by iteratively computing over-approximation intervals of the output of each layer, until the last layer of the network is reached. These over-approximations are obtained by adapting to neural networks existing methods for reachability analysis of dynamical systems. More specifically, we rely on the mixed-monotonicity approach in [20] to over-approximate the reachable set of any discrete-time system $x^+ = f(x)$ with Lipschitz-continuous vector field $f : \mathbb{R}^n \to \mathbb{R}^n$. Since the neural network (1) is

instead defined as a function $y = f(x)$ with input $x = x^{(0)}$ and output $y = x^{(L)}$ of different dimensions, we present below the generalization of the mixed-monotonicity method from [20] to any Lipschitz-continuous function.

Consider the function $y = f(x)$ with input $x \in [\underline{x}, \overline{x}] \subseteq \mathbb{R}^{n_x}$, output $y \in \mathbb{R}^{n_y}$ and Lipschitz-continuous function $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$. The Lipschitz-continuity assumption ensures that the derivative f' (also called Jacobian matrix in the paper) is bounded.

Proposition 2: Given an interval $[\underline{J}, \overline{J}] \subseteq \mathbb{R}^{n_y \times n_x}$ bounding the derivative $f'(x)$ for all $x \in [\underline{x}, \overline{x}]$, denote the center of the interval as J^*. For each output dimension $i \in \{1, \dots, n_y\}$, define input vectors $\underline{\xi^i}, \overline{\xi^i} \in \mathbb{R}^{n_x}$ and row vector $\alpha^i \in \mathbb{R}^{1 \times n_x}$ such that for all $j \in \{1, \dots, n_x\}$,

$$(\underline{\xi^i}_j, \overline{\xi^i}_j, \alpha^i_j) = \begin{cases} (\underline{x}_j, \overline{x}_j, \min(0, \underline{J}_{ij})) & \text{if } J^*_{ij} \geq 0, \\ (\overline{x}_j, \underline{x}_j, \max(0, \overline{J}_{ij})) & \text{if } J^*_{ij} \leq 0. \end{cases}$$

Then for all $x \in [\underline{x}, \overline{x}]$ and $i \in \{1, \dots, n_y\}$, we have:

$$f_i(x) \in \left[f_i(\underline{\xi^i}) - \alpha^i(\underline{\xi^i} - \overline{\xi^i}), f_i(\overline{\xi^i}) + \alpha^i(\underline{\xi^i} - \overline{\xi^i}) \right].$$

Intuitively, the output bounds are obtained by computing for each output dimension the images for two diagonally opposite vertices of the input interval, then expanding these bounds with an error term when the bounds on the derivative f' spans both negative and positive values. Proposition 2 can thus provide an interval over-approximation of the output set of any function as long as bounds on the derivative f' are known. Obtaining such bounds for a neural network is made possible by computing local bounds on the derivative of its activation functions, as detailed in Section II-C.

C. Local bounds of activation functions

Proposition 2 can be applied to any neural network whose activation functions are Lipschitz continuous since such functions have a bounded derivative. However, to avoid requiring the users to manually provide these bounds for each different activation function they want to use, we instead provide a framework to automatically and easily compute local bounds for a very general class of functions describing most popular activation functions and their derivatives.

Let $\mathbb{R}_\infty = \mathbb{R} \cup \{-\infty, +\infty\}$ and consider the scalar function $\varphi : \mathbb{R}_\infty \to \mathbb{R}_\infty$, where $\varphi(x) \in \{-\infty, +\infty\}$ only when $x \in \{-\infty, +\infty\}$. In this paper, we focus on 3-piece piecewise-monotone functions for which there exist $\underline{z}, \overline{z} \in \mathbb{R}_\infty$ such that φ is:

- non-increasing on $(-\infty, \underline{z}]$ until reaching its global minimum $\min_{x \in \mathbb{R}_\infty} \varphi(x) = \varphi(\underline{z})$;
- non-decreasing on $[\underline{z}, \overline{z}]$ until reaching its global maximum $\max_{x \in \mathbb{R}_\infty} \varphi(x) = \varphi(\overline{z})$;
- and non-increasing on $[\overline{z}, +\infty)$.

When $\underline{z} = -\infty$ (resp. $\overline{z} = +\infty$), the first (resp. last) monotone segment is squeezed into a singleton at infinity and can thus be ignored.

Although this description may appear restrictive, we should note that the vast majority of activation functions as well as their derivatives belong to this class of functions. In particular, this is the case for (but not restricted to) popular piecewise-affine activation functions (identity, ReLU, leaky ReLU), monotone functions (hyperbolic tangent, arctangent, sigmoid, SoftPlus, ELU) as well as many non-monotone activation functions (SiLU, GELU, HardSwish, Mish, REU, PFLU, FPFLU, which are all reviewed or introduced in [19]).

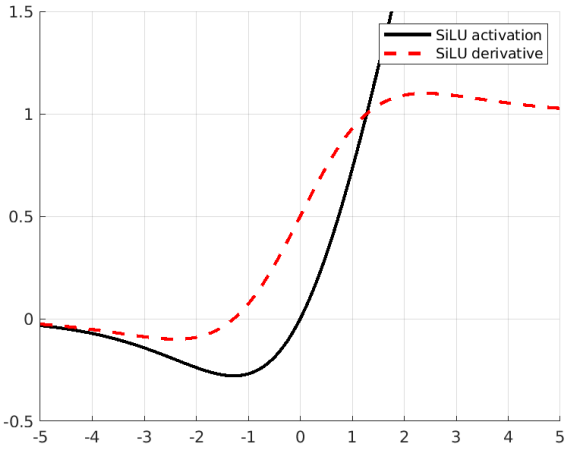

Fig. 1. SiLU activation function (black line) and its derivative (red dashed).

Example 1: Two examples of such functions are provided in Figure 1. The first one (in black) is the Sigmoid Linear Unit (SiLU, also called *Swish* [17]) activation function $\Phi(x) = x/(1+e^{-x})$, whose global $\arg\min$ is $\underline{z} = -1.2785$. Its global $\arg\max$ is $\overline{z} = +\infty$, which implies that the third monotone component is pushed toward $+\infty$ since it is not needed for this function.

The second example (in dashed red) is the derivative of the SiLU activation function $\Phi'(x) = (1 + e^{-x} + xe^{-x})/(1 + e^{-x})^2$, with global $\arg\min$ and $\arg\max$ defined as $\underline{z} = -2.3994$ and $\overline{z} = 2.3994$, respectively. △

The only exceptions of activation functions whose derivative do not belong to this class that the author could find in the literature are: the binary step, which is discontinuous so its derivative is undefined in 0; and the Gaussian activation function $\Phi(x) = e^{-x^2}$ which belongs to this class, but not its derivative (although $-\Phi'$ does belong to this class, so a very similar approach can still be applied).

Proposition 3: Given a scalar function φ as defined above and a bounded input domain $[\underline{x}, \overline{x}] \in \mathbb{R}$, the local bounds of φ on $[\underline{x}, \overline{x}]$ are given by:

$$\min_{x \in [\underline{x}, \overline{x}]} \varphi(x) = \begin{cases} \varphi(\underline{z}) & \text{if } \underline{z} \in [\underline{x}, \overline{x}], \\ \min(\varphi(\underline{x}), \varphi(\overline{x})) & \text{otherwise,} \end{cases}$$

$$\max_{x \in [\underline{x}, \overline{x}]} \varphi(x) = \begin{cases} \varphi(\overline{z}) & \text{if } \overline{z} \in [\underline{x}, \overline{x}], \\ \max(\varphi(\underline{x}), \varphi(\overline{x})) & \text{otherwise.} \end{cases}$$

In this paper, Proposition 3 is used to obtain local bounds of the activation function derivatives in order to compute bounds on the network's Jacobian matrix for Proposition 2. However, Proposition 3 can also be useful on activation functions themselves to compare (in Section IV) our method with the naive interval propagation through the layers.

III. Reachability algorithm

Solving Problem 1 by applying the mixed-monotonicity approach from Proposition 2 on the neural network (1) can be done in many different ways. One possibility is to iteratively compute bounds on the Jacobian matrix of the network and then apply Proposition 2 once for the whole network. However, this may result in wide bounds for the Jacobian of the whole network, and thus in a fairly conservative over-approximation of the network output due to the term α^i in Proposition 2. The dual approach is to iteratively apply Proposition 2 to each layer of the network, which would result in tighter Jacobian bounds (since one layer's Jacobian requires less interval matrix products than the Jacobian of the whole network), and thus tighter over-approximation of the layer's output. However, the loss of the dependency with respect to the network's input may induce another source of accumulated conservativeness if we have many layers. Any intermediate approach can also be considered, where we split the network into several consecutive partial networks and apply Proposition 2 iteratively to each of them.

Although all the above approaches result in over-approximations of the network output, we cannot determine in advance which method would yield the tightest bounds, since this is highly dependent on the considered network and input interval. We thus devised an algorithm that encapsulates all possible choices by running Proposition 2 on all partial networks within (1), before taking the intersection of the obtained bounds to tighten the bounds on each layer's output. The main steps of this approach are presented in Algorithm 1 and summarized below.

Given the L-layer network (1) with activation function Φ and input interval $[\underline{x^0}, \overline{x^0}]$ as in Problem 1, our goal is to apply Proposition 2 to each partial network of (1), denoted as $\texttt{NN}(k,l)$, containing only layers k to l (with $k \leq l$) and with input $x^{(k-1)}$ and output $x^{(l)}$. We initialize the Jacobian bounds of each partial network to the identity matrix. Then, we iteratively explore the network (going forward), where for each layer l we first use interval arithmetics [22] to compute the pre-activation bounds based on the knowledge of the output bounds of the previous layer, and then apply Proposition 3 to obtain local bounds on the activation function derivative ($\varphi = \Phi'$). Next, for each partial network $\texttt{NN}(k,l)$ (with $k \leq l$) ending at the current layer l, we compute its Jacobian bounds based on the Jacobian of layer l ($[\underline{\Phi'}, \overline{\Phi'}] * W^{(l)}$) and the Jacobian of the partial network $\texttt{NN}(k, l-1)$. We then apply Proposition 2 to the partial network $\texttt{NN}(k,l)$ with the Jacobian bounds we just computed and input bounds $[\underline{x^{(k-1)}}, \overline{x^{(k-1)}}]$. Finally, once we computed the l over-approximations of $x^{(l)}$ corresponding to each partial network ending at layer l, we take the intersection of all of them to obtain the final bounds for $x^{(l)}$.

Theorem 4: The interval $[\underline{x^{(L)}}, \overline{x^{(L)}}]$ returned by Algorithm 1 is a solution to Problem 1.

This is easily proved by the fact that we compute several interval over-approximations of the output of each layer using Proposition 2, which ensures that their intersection is still an over-approximation of the layer's output. Then, using these over-approximations as the input bounds of the next layers guarantees the soundness of the approach.

Input: L-layer network (1), input interval $[\underline{x^0}, \overline{x^0}]$, activation function Φ (with Φ' defined as in Section II-C)

$\forall k, l \in \{1, \dots, L\},\ \underline{J(k,l)} \leftarrow I, \overline{J(k,l)} \leftarrow I$

for $l \in \{1, \dots, L\}$ **do**

  $[\underline{\Phi'}, \overline{\Phi'}] \leftarrow \texttt{Prop3}(\Phi', W^{(l)} * [\underline{x^{(l-1)}}, \overline{x^{(l-1)}}] + b^{(l)})$

  for $k \in \{1, \dots, l\}$ **do**

    $[\underline{J(k,l)}, \overline{J(k,l)}] \leftarrow$
    $[\underline{\Phi'}, \overline{\Phi'}] * W^{(l)} * [\underline{J(k,l-1)}, \overline{J(k,l-1)}]$

    $[\underline{x(k,l)}, \overline{x(k,l)}] \leftarrow$
    $\texttt{Prop2}(\texttt{NN}(k,l), [\underline{x^{(k-1)}}, \overline{x^{(k-1)}}], [\underline{J(k,l)}, \overline{J(k,l)}])$

  $[\underline{x^{(l)}}, \overline{x^{(l)}}] \leftarrow [\underline{x(1,l)}, \overline{x(1,l)}] \cap \dots \cap [\underline{x(l,l)}, \overline{x(l,l)}]$

Output: Over-approximation $[\underline{x^{(L)}}, \overline{x^{(L)}}]$ of the network output

Algorithm 1: Mixed-monotonicity reachability analysis of a feedforward neural network.

Note also that in Algorithm 1, Proposition 2 is applied to every partial network that exists within the main network (1). Although this implies a computational complexity of $O(L^2)$, it guarantees that the resulting interval $[\underline{x^{(L)}}, \overline{x^{(L)}}]$ from Algorithm 1 is the least conservative solution to Problem 1 that could be obtained from applying the mixed-monotonicity reachability analysis of Proposition 2 to any decomposition of (1) into consecutive partial networks.

IV. Numerical comparisons

To evaluate the performances of our approach, we run Algorithm 1 on both established benchmarks and randomly generated neural networks, and compare the results with state-of-the-art tools for reachability analysis and verification of neural networks. [1] Since our method is an optimization-free reachability analysis approach using interval bounds, we focus these numerical comparisons to the most relevant toolboxes of this category, and we leave to future work the additional comparisons with less related verification tools relying on optimization-based approaches [3] or reachability analysis with more complex set representations (polyhedra, zonotopes) as in the ERAN [12] and NNV libraries [11].

We provide comparisons to the following 5 tools and methods for reachability analysis of neural networks. We first use as a baseline the naive interval bound propagation (IBP), as presented e.g. in [5], since this is the simplest approach using interval arithmetic at each layer, but it is also very conservative due to losing dependency to the network's inputs at each step. Three of the other tools (ReluVal [7], Neurify [8], VeriNet [9]) rely on methods propagating symbolic intervals (i.e. bounding functions linear in $x^{(0)}$), sometimes combined with linear relaxations of the nonlinear activation functions.

[1] All computations are done on a laptop with 1.80GHz processor and 16GB of RAM running Matlab 2021b. The code used to generate all numerical comparisons described in this section is available at: `https://gitlab.com/pj_meyer/MMRANN`

The fifth tool (CROWN [10]) instead relies on a backward propagation of these linear relaxations.

Since these tools are written in various programming languages (Matlab, C, Python), for the convenience of comparison we have re-implemented the bounding method of each tool in Matlab. The implemented approaches are those described in each of the original papers cited above, and may thus slightly differ from the latest updates of the corresponding public toolboxes. Note also that the comparisons in this section only focus on reachability analysis as formulated in Problem 1 (over-approximating the network's reachable set for a given input uncertainty), and we thus disregard the iterative refinement algorithms that some of the above tools use to verify or falsify properties on the network's behavior.

VNN benchmarks: We first compare our mixed-monotonicity approach with the above 5 reachability tools on the *MNISTFC* benchmark as presented in the 2021 VNN competition [21]. This benchmark is composed of 3 feedfoward ReLU neural networks trained on the MNIST dataset for the recognition of handwritten digits. All 3 networks have 784 inputs (28×28 pixels of a gray-scale picture), an output layer with 10 nodes, and an additional 2, 4 or 6 hidden layers of 256 nodes each. We run all 6 reachability methods on these 3 networks for 250 different input pictures, each with an $\epsilon = 0.03$ uncertainty around their nominal values. In terms of tightness of the output bounds, our approach is 10 to 1000 tighter than IBP and has a comparable order of magnitude with the other four tools. In the top half of Table I, the first three columns summarize the percentage of the 250 evaluated input pictures for which our mixed-monotonicity approach results in tighter (or at least as tight) output bounds than the other five methods. We can see in particular that our method outperforms all others in at least 70% cases on the benchmark with two hidden layers. In terms of computation times (bottom of Table I), our approach is significantly slower than IBP and VERINET and a bit slower than CROWN, but it is also up to 3 times faster than RELUVAL and NEURIFY.

We ran the same comparison on the only non-ReLU feedforward benchmark of the 2021 VNN competition [21]: the *ERAN-sigmoid* benchmark is also trained on the MNIST dataset and has the same structure as the above MNISTFC networks, but with 6 hidden layers of 200 nodes each, sigmoid activation functions and input uncertainty $\epsilon = 0.012$. On the same 250 input pictures, we observe that: IBP is always more conservative than our approach (as expected); RELUVAL and NEURIFY cannot be run on non-ReLU networks; VERINET and CROWN both fail for every 250 inputs because their results are so conservative on this benchmark that Matlab evaluates $e^{-x} = \infty$ when the pre-activation value x goes too far in the negative, and the (theoretically horizontal) tangent of the sigmoid $e^{-x}/(1+e^{-x})^2$ cannot be computed. VERINET and CROWN's failure due to excessive conservativeness on this sigmoid benchmark is also observed when using an input uncertainty $\epsilon = 1.2 * 10^{-5}$ which is 1000 smaller than the one suggested in the VNN benchmark [21].

Random networks: To get more representative comparisons on a wider variety of network dimensions (since the above benchmarks are only 4 specific networks), and more importantly to highlight the main strength of our approach handling general activation functions (since we could not find well-established benchmarks using any activation other than ReLU and Sigmoid), we run these comparisons over two large sets of randomly generated neural networks.

Method	**mnistfc-2**	**mnistfc-4**	**mnistfc-6**	**ERAN-sig**
IBP [5]	100%	100%	100%	100%
RELUVAL [7]	80%	74.8%	65.6%	-
NEURIFY [8]	72.4%	72%	58.8%	-
VERINET [9]	70.4%	54%	15.6%	100%
CROWN [10]	70.4%	54%	15.6%	100%
IBP [5]	0.012	0.019	0.025	0.016
RELUVAL [7]	42	62	85	-
NEURIFY [8]	43	62	84	-
VERINET [9]	0.012	0.019	0.026	0.017
CROWN [10]	0.78	4.1	10	5.7
Mixed-Mono	15	36	69	39

TABLE I

TOP: PROPORTION OF THE **250** MNIST INPUT PICTURES FOR WHICH OUR APPROACH GIVES OUTPUT BOUNDS TIGHTER THAN (OR EQUAL TO) OTHER METHODS. BOTTOM: AVERAGE COMPUTATION TIME IN s.

We first consider a set of 10000 small feedforward networks as defined in (1) and whose parameters are randomly chosen as follows: depth $L \in \{1, \dots, 5\}$; input and output dimensions $n_0, n_L \in \{1, \dots, 10\}$; width of hidden layers (each layer may have a different width) $n_1, \dots, n_{L-1} \in \{1, \dots, 30\}$. The second set contains 1000 deeper and wider networks with: depth $L \in \{5, \dots, 10\}$; input dimension $n_0 \in \{500, \dots, 1000\}$; output dimension $n_L \in \{10, \dots, 50\}$; width of hidden layers $n_1, \dots, n_{L-1} \in \{100, \dots, 200\}$. The input bounds as in Problem 1 are defined as the hypercube $[\underline{x^{(0)}}, \overline{x^{(0)}}] = \{x^*\} + [-0.1, 0.1]^{n_0}$ around a randomly chosen center input $x^* \in [-1, 1]^{n_0}$. The comparison between our mixed-monotonicity method in Algorithm 1 and the five approaches from the literature cited above (when applicable) is done for each of these 4 activation functions:

- Rectified Linear Unit (ReLU) is the piecewise-affine function $\Phi(x) = \max(0, x)$, handled by all 6 tools;
- Hyperbolic tangent (TanH) is the S-shaped function $\Phi(x) = (e^x - e^{-x})/(e^x + e^{-x})$, handled by all tools apart from RELUVAL and NEURIFY;
- Exponential Linear Unit (ELU) is the monotone function $\Phi(x) = e^x - 1$ if $x \leq 0$ and $\Phi(x) = x$ if $x \geq 0$, natively handled only by IBP and our method. ELU is compatible with the framework of VERINET and CROWN (although not implemented) at the condition of providing a method to compute linear equations bounding Φ on any bounded input set. We added this to our own Matlab implementation of these tools;
- Sigmoid Linear Unit (SiLU) is the non-monotone function $\Phi(x) = x/(1 + e^{-x})$, only natively handled by our method. Although the implementation of IBP in [5] is restricted to monotone activation functions, our own Matlab implementation of IBP is extended to handle non-monotone activation by using Proposition 3.

All these activation functions are Lipschitz continuous and their derivatives satisfy the desired shape described in Sec-

tion II-C. Algorithm 1 can thus be applied to all of them.

For the set of 10000 small networks, Table II gives the average computation times of each methods for each activation function. For these time averaging to be meaningful despite the varying sizes of the networks, we actually divide the computation times by the total number of neurons in the considered network, and only then we take the average. Table III gives the proportion of the 10000 small networks for which our mixed-monotonicity approach results in at least as tight output bounds as the other methods. Table IV gives the same result but only when our approach is strictly tighter. For the 1000 large networks, the same results are displayed in Tables V-VII. Note that the time tables are in microseconds for small networks (Table II), and milliseconds for large ones (Table V). In the next three paragraphs, we analyse these results and highlight the main take-away conclusions from these numerical comparisons in terms of generality of the methods, complexity, and tightness of the output bounds.

Among all 6 compared methods, our mixed-monotonicity approach is the most general one and the only one that could natively handle all considered activation functions. Indeed, RELUVAL and NEURIFY are limited to piecewise-affine functions (such as ReLU) and VERINET and CROWN cannot handle non-monotone functions (such as SiLU). In addition, although we added an implementation for IBP to handle SiLU, and for VERINET and CROWN to handle ELU, the original tools do not natively handle these activation functions. (The corresponding results are given in parentheses in Tables II-VII.) This last comment brings up another important aspect of the generality of our mixed-monotonicity approach: the ease of implementation to add new activation functions. Indeed, NEURIFY, VERINET and CROWN rely on linear relaxations of the nonlinear activation functions, which may require long and complex implementations to be provided by the user for each new function (e.g. finding the optimal relaxations of sigmoid-shaped functions takes several hundreds of lines of code in the implementation of VERINET). In contrast, for a user to add a new activation type to be used within our mixed-monotonicity approach, all they need to provide is the definition of the activation function and its derivative, and the global $\arg\min$ and $\arg\max$ of the derivative as defined in Section II-C. Everything else is automatically handled internally by Algorithm 1.

This generality and ease of use however comes at the cost of a higher computational complexity of $O(L^2)$ since Algorithm 1 calls the mixed-monotonicity reachability method from Proportion 2 on all $L(L+1)/2$ partial networks within the main L-layer network. On the set of smaller networks, this implies that our approach is the slowest, with a similar order of magnitude as CROWN (which also has a complexity of $O(L^2)$), while the other four methods are 10 to 20 times faster (Table II). When the size of the network increases, IBP and VERINET are always the fastest methods and keep a consistent average computation time (per number of neurons in the network), while all other methods see their computation time per neuron increase, the worst being RELUVAL and NEURIFY which become slower than our mixed-monotonicity approach (Table V).

Method	ReLU	TanH	ELU	SiLU
IBP [5]	12	18	11	(13)
RELUVAL [7]	29	-	-	-
NEURIFY [8]	27	-	-	-
VERINET [9]	14	33	(25)	-
CROWN [10]	199	213	(177)	-
Mixed-Monotonicity	591	462	550	543

TABLE II

AVERAGE COMPUTATION TIME (PER NEURON IN THE NETWORK) IN μs FOR THE SET OF **10000** SMALL NETWORKS.

Method	ReLU	TanH	ELU	SiLU
IBP [5]	100%	100%	100%	(100%)
RELUVAL [7]	68%	-	-	-
NEURIFY [8]	46%	-	-	-
VERINET [9]	43%	32%	(40%)	-
CROWN [10]	43%	31%	(38%)	-

TABLE III

PROPORTION OF THE **10000** SMALL RANDOM NETWORKS FOR WHICH OUR MIXED-MONOTONICITY APPROACH RESULTS IN TIGHTER (OR EQUAL) OUTPUT BOUNDS THAN OTHER METHODS.

Finally, in terms of tightness (measured as the 2-norm width of the output bounds), we first observe that, as expected, our approach is always at least as tight as the naive IBP method (Tables III and VI), and even strictly tighter in 70-80% cases in the set of small networks (Table IV). Apart from the very conservative IBP, it should be noted that none of the other 5 methods always outperforms the others. For example on the set of 10000 small ReLU networks, our approach is strictly tighter than the other four in 12-36% cases, equal in 31% cases, and strictly looser in the remaining 32-57% cases (Tables III and IV). The respective strengths of each method thus make them complementary. On average, our approach returns at least as tight output bounds as the state-of-the-art tools in less than half cases on smaller networks, but this percentage significantly increases on larger networks except for ReLU (Tables III and VI). When the size of the network increases, we also observe a larger proportion of cases where the compared methods return approximately equal output bounds, particularly on activation functions with approximate saturations, such as TanH (Tables VI and VII). However, one interesting case if with the ELU activation (for which the state-of-the-art tools where not designed and optimized), where our approach not only outperforms both VERINET and CROWN on almost all 1000 tested large networks, but it also obtains strictly tighter output bounds than VERINET in 79% cases, and CROWN is seen to fail to return a numerical result in all cases due to its excessive conservativeness and its formulation in [10] being incompatible with horizontal linear relaxations (similarly to the ERAN-sigmoid benchmark).

V. CONCLUSIONS

This paper presents a new method for the sound reachability analysis of feedforward neural networks using mixed-monotonicity. The main strength of our approach is its very broad applicability to any network with Lipschitz-continuous activation functions, which is satisfied by the vast majority of activation functions currently in use in the literature. Another

Method	ReLU	TanH	ELU	SiLU
IBP [5]	73%	71%	79%	(79%)
RELUVAL [7]	36%	-	-	-
NEURIFY [8]	16%	-	-	-
VERINET [9]	12%	11%	(19%)	-
CROWN [10]	12%	10%	(17%)	-

TABLE IV

PROPORTION OF THE **10000** SMALL RANDOM NETWORKS FOR WHICH OUR MIXED-MONOTONICITY APPROACH RESULTS IN STRICTLY TIGHTER OUTPUT BOUNDS THAN OTHER METHODS.

Method	ReLU	TanH	ELU	SiLU
IBP [5]	0.016	0.018	0.016	(0.018)
RELUVAL [7]	44	-	-	-
NEURIFY [8]	44	-	-	-
VERINET [9]	0.034	0.05	(0.037)	-
CROWN [10]	4.8	4.8	(4.8)	-
Mixed-Monotonicity	33	29	34	33

TABLE V

AVERAGE COMPUTATION TIME (PER NEURON IN THE NETWORK) IN ms FOR THE SET OF **1000** LARGE NETWORKS.

significant strength of our framework is the greater simplicity to implement new activation functions compared to existing tools relying on linear relaxations. We compared our approach with the naive Interval Bound Propagation and 4 state-of-the-art tools using interval reachability analysis on neural network (RELUVAL, NEURIFY, VERINET, CROWN) on both benchmarks from the VNN competition [21] and sets of randomly generated neural networks of a wide variety of depth and width. From these comparisons, we observed a good complementarity between our approach and the state-of-the-art tools (when applicable on the considered activation functions), where each outperforms the others on a significant percentage of the tested inputs and networks.

Although this tool can already be used on its own for the reachability analysis of neural networks, our next objective is to address its main limitation of a $O(L^2)$ complexity to make it more compatible with iterative refinement approaches similar to existing neural network verification algorithms.

Method	ReLU	TanH	ELU	SiLU
IBP [5]	100%	100%	100%	(100%)
RELUVAL [7]	100%	-	-	-
NEURIFY [8]	2%	-	-	-
VERINET [9]	0.1%	100%	(98%)	-
CROWN [10]	0.1%	100%	(100%)	-

TABLE VI

PROPORTION OF THE **1000** LARGE RANDOM NETWORKS FOR WHICH OUR MIXED-MONOTONICITY APPROACH RESULTS IN TIGHTER (OR EQUAL) OUTPUT BOUNDS THAN OTHER METHODS.

Method	ReLU	TanH	ELU	SiLU
IBP [5]	0%	0%	0%	(0%)
RELUVAL [7]	0%	-	-	-
NEURIFY [8]	0%	-	-	-
VERINET [9]	0%	0%	(79%)	-
CROWN [10]	0%	2%	(100%)	-

TABLE VII

PROPORTION OF THE **1000** LARGE RANDOM NETWORKS FOR WHICH OUR MIXED-MONOTONICITY APPROACH RESULTS IN STRICTLY TIGHTER OUTPUT BOUNDS THAN OTHER METHODS.

REFERENCES

[1] W. Xiang, P. Musau, A. A. Wild, D. M. Lopez, N. Hamilton, X. Yang, J. Rosenfeld, and T. T. Johnson, "Verification for machine learning, autonomy, and neural networks survey," *arXiv preprint arXiv:1810.01989*, 2018.

[2] E. Kim, D. Gopinath, C. Pasareanu, and S. A. Seshia, "A programmatic and semantic approach to explaining and debugging neural network based object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 128–11 137.

[3] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," *Foundation and Trend in Optimization*, vol. 4, no. 3-4, pp. 244–404, 2021.

[4] M. Akintunde, A. Lomuscio, L. Maganti, and E. Pirovano, "Reachability analysis for neural agent-environment systems," in *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2018.

[5] W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson, "Reachable set estimation for neural network control systems: A simulation-guided approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 1821–1830, 2020.

[6] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robustness verification of neural networks," *arXiv preprint arXiv:1902.08722*, 2019.

[7] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Formal security analysis of neural networks using symbolic intervals," in *27th USENIX Security Symposium*, 2018.

[8] ——, "Efficient formal safety analysis of neural networks," in *32nd Conference on Neural Information Processing Systems*, 2018.

[9] P. Henriksen and A. Lomuscio, "Efficient neural network verification via adaptive refinement and adversarial search," in *ECAI 2020*. IOS Press, 2020, pp. 2513–2520.

[10] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in Neural Information Processing Systems*, vol. 31, pp. 4939–4948, 2018.

[11] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *International Conference on Computer Aided Verification*. Springer, 2020, pp. 3–17.

[12] "ERAN: ETH robustness analyzer for neural networks," 2022, available online at: https://github.com/eth-sri/eran.

[13] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, *et al.*, "The marabou framework for verification and analysis of deep neural networks," in *International Conference on Computer Aided Verification*, 2019, pp. 443–452.

[14] E. Botoeva, P. Kouvaros, J. Kronqvist, A. Lomuscio, and R. Misener, "Efficient verification of relu-based neural networks via dependency analysis," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3291–3299.

[15] K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, and P. Kohli, "A dual approach to scalable verification of deep networks." in *UAI*, vol. 1, no. 2, 2018, p. 3.

[16] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations*, 2018.

[17] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[18] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[19] M. Zhu, W. Min, Q. Wang, S. Zou, and X. Chen, "PFLU and FPFLU: Two novel non-monotonic activation functions in convolutional neural networks," *Neurocomputing*, vol. 429, pp. 110–117, 2021.

[20] P.-J. Meyer, A. Devonport, and M. Arcak, *Interval Reachability Analysis: Bounding Trajectories of Uncertain Systems with Boxes for Control and Verification*. Springer, 2021.

[21] S. Bak, C. Liu, and T. Johnson, "The second international verification of neural networks competition (VNN-COMP 2021): Summary and results," *arXiv preprint arXiv:2109.00498*, 2021.

[22] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied interval analysis: with examples in parameter and state estimation, robust control and robotics*. Springer Science & Business Media, 2001.