Reachability Analysis of Sigmoidal Neural Networks

SUNG WOO CHOI, University of Nebraska-Lincoln, USA
MICHAEL IVASHCHENKO, University of Nebraska-Lincoln, USA
LUAN V. NGUYEN, University of Dayton, USA
HOANG-DUNG TRAN, University of Nebraska-Lincoln, USA

This paper extends the star set reachability approach to verify the robustness of feed-forward neural networks (FNNs) with sigmoidal activation functions such as Sigmoid and TanH. The main drawbacks of the star set approach in Sigmoid/TanH FNN verification are scalability, feasibility, and optimality issues in some cases due to the linear programming solver usage. We overcome this challenge by proposing a relaxed star (RStar) with symbolic intervals, which allows the usage of the back-substitution technique in DeepPoly to find bounds when overapproximating activation functions while maintaining the valuable features of a star set. RStar can overapproximate a sigmoidal activation function using four linear constraints (RStar4) or two linear constraints (RStar2), or only the output bounds (RStar0). We implement our RStar reachability algorithms in NNV and compare them to DeepPoly via robustness verification of image classification DNNs benchmarks. The experimental results show that the original star approach (i.e., no relaxation) is the least conservative of all methods yet the slowest. RStar4 is computationally much faster than the original star method and is the second least conservative approach. It certifies up to 40% more images against adversarial attacks than DeepPoly and on average 51 times faster than the star set. Last but not least, RStar0 is the most conservative method, which could only verify two cases for the CIFAR10 small Sigmoid network, $\delta = 0.014$. However, it is the fastest method that can verify neural networks up to 3528 times faster than the star set and up to 46 times faster than DeepPoly in our evaluation.

CCS Concepts: • **General and reference → Verification**; • **Software and its engineering → Formal methods**; • **Computing methodologies → Neural networks**.

Additional Key Words and Phrases: Verification, Deep Neural Networks, Formal Methods, Reachability Analysis.

1 INTRODUCTION

In recent years, deep neural network (DNN) has been a widely used technique for solving real-world tasks in various areas. DNNs consist of multiple layers of many neurons with weights and biases that are randomly initialized and transformed to high-dimensional vectors during the training throughout nonlinear activation functions. Understanding and predicting the performance of DNNs are challenging due to the enormous number of neurons, the non-linearity of activation, and the complex structure of a DNN. Thus, DNN is considered as a black box. Despite its success in classification and recognition, it is vulnerable to adversarial attacks [1, 19, 29]. A small adversarial attack on the input images that is indistinguishable from a human eye may lead DNNs to classify incorrectly with high confidence [5]. Furthermore, generating adversarial examples for DNN models does not require knowledge about the parameters and architecture of the models [22]. As neural networks are

Authors' addresses: Sung Woo Choi, schoi9@huskers.unl.edu, School of Computing, University of Nebraska-Lincoln, USA; Michael Ivashchenko, mivashchenko2@huskers.unl.edu, School of Computing, University of Nebraska-Lincoln, USA; Luan V. Nguyen, lnguyen1@udayton.edu, Department of Computer Science, University of Dayton, USA; Hoang-Dung Tran, dtran30@unl.edu, School of Computing, University of Nebraska-Lincoln, USA.

1084-4309/2023/10-ART $15.00
https://doi.org/10.1145/3627991

applied to safety-critical systems, formal methods for safety verification and robustness certification of DNNs are critical and highly necessary.

Verification and certification of DNNs can be broadly categorized into exact and overapproximation analyses, depending on activation functions and analysis methods. Nonlinear activation functions such as ReLU and leaky ReLU can either be analyzed exactly or by overapproximation due to their piece-wise linear properties [30–32, 34]. Nonlinear activation functions such as Sigmoid and TanH are generally analyzed with overapproximation methods as an exact approach is computationally intractable. Despite their wide usage in real-world neural networks, sigmoidal DNNs have not been extensively researched, and only a few verification approaches have been proposed [6, 8, 9, 11, 18, 27, 38, 47]. These activation functions are much more challenging and complex to verify due to their nonlinear properties. Most methods proposed recently for these functions are overapproximated with abstraction, such as symbolic intervals [28, 46] that are limited to overapproximate an activation function with two linear bounds. The exact verification guarantees the soundness and completeness of the reachability analysis. However, it is computationally expensive and may not be scalable to verify deep neural networks in a designated time frame. On the other hand, overapproximation analysis is faster and more scalable than the exact analysis, but it guarantees only the soundness of the results. Importantly, errors in overapproximation analysis may accumulate quickly over layers. Therefore, loose overapproximation produces a conservative reachable set which may be useless for verification. Although a trade-off between scalability and precision is necessary, the conservativeness of results may be dramatic on verification.

This research is inspired by the star reachability [32] and DeepPoly [28] approach for FNNs verification. Previously, the star reachability was used to verify ReLU FNNs. This paper extends the star set approach to verify networks with TanH and Sigmoid. The star set approach allows a nonlinear activation function to be overapproximated by many linear constraints. Therefore, it produces tighter overapproximation than the DeepPoly [28] and CROWN [46] approaches, which overapproximate activation functions such as ReLU, TanH, and Sigmoid using only upper and lower linear constraints. Using two linear constraints allows these methods to do back-substitution to find the bound at each neuron quickly to perform overapproximation without solving linear programming optimization. This key feature makes DeepPoly and CROWN more scalable than the star set approach on large networks. However, using two linear constraints in overapproximation also makes DeepPoly and CROWN more conservative than the star set approach. Based on this observation, we propose a new relaxed star (RStar) approach utilizing symbolic intervals with the back-substitution technique similar to DeepPoly to find the state bounds in reachability analysis while maintaining efficient features of the star set representation to reduce the conservativeness of DeepPoly. Users can choose to overapproximate the Sigmoid or TanH activation function by four linear constraints (RStar4), two linear constraints (RStar2), or only the lower bound and upper bound of the function's output (RStar0). Our approach is implemented in NNV [35], a tool for verifying DNNs and learning-enabled cyber-physical systems.

For the evaluation, we verify the robustness of the neural networks against the L_∞ norm attack. A neural network is considered robust to an input set if the classified label of an image holds for any perturbation attack. We prepare a set of three FNNs with different architectures that are trained with two datasets: MNIST [17] and CIFAR10 [16]. We evaluate our proposed methods in comparison with the DeepPoly approach for verifying FNNs. We discuss the evaluation in terms of scalability, time performance, conservativeness, and verification result. In short, the relaxed star approaches are more scalable than the original star set while less conservative than DeepPoly on a set of DNN benchmarks. RStar0 is capable of verifying 3528 times faster than the original star method. The RStar approaches have improved scalability due to discarding a linear programming (LP) solver and employing symbolic intervals to find the state bounds during reachability analysis. RStar4 can verify up to 40% more images than DeepPoly on bigger neural networks, while the star set approach can verify at most 17% more images than RStar4 on MNIST DNNs. The evaluation reveals that applying more linear constraints while overapproximating an activation function improves precision. Although a multi-core platform is not considered in this experiment, the star set's scalability problem can be easily improved with parallel computation of the reachable sets.

Contributions. In summary, the main contributions of the paper are as follows.

1) An extension of the star set to overapproximate reachability analysis of Sigmoid/TanH FNNs.
2) A new relaxed star (RStar) approach utilizing symbolic intervals with the back-substitution technique to overapproximate a sigmoidal activation function that includes three variants (RStar4, RStar2, RStar0) to balance between conservativeness and scalability.
3) An implementation of the star set and RStar representation and the reachability algorithms in NNV.
4) A thorough evaluation and comparison with DeepPoly on a set of benchmarks.

Organization. The remainder of the paper is organized as follows. Section 2 formally defines FNN and its reachability, in which the definition of generalized star set and symbolic intervals are stated in detail. In Section 3, we provide propositions and lemmas of how an overapproximation of non-linear functions such as Sigmoid and TanH is applied based on the star set and symbolic intervals. Moreover, we describe the back-substitution of the symbolic intervals and their composition. We then propose the relaxed star that employs the star set and symbolic intervals for the reachability analysis of an FNN. In Section 4, we explain the reachability analysis of each method based on the relaxed star reachability algorithm and show their differences. Then, we present our evaluation in Section 5. Section 6 reviews the related works, and Section 7 concludes the paper.

2 PRELIMINARIES

A k-layer feed-forward neural network $\mathcal{F}$ consists of an input layer, $k-1$ hidden layers, and an output layer. Each layer ℓ, $1 \leq \ell \leq k$, consists of n_ℓ neurons that are interconnected to $n_{\ell-1}$ neurons in a preceding layer. A hidden layer ℓ performs two operations: affine mapping and activating. The affine mapping operation is a function $f_\ell^M : \mathbb{R}^{n_{\ell-1}} \longrightarrow \mathbb{R}^{n_\ell}$ performing a linear transformation on a state vector y, which is the output of the previous layer, i.e., layer $\ell - 1$. The activating operation applies the activation function $f_\ell^\sigma : \mathbb{R}^{n_\ell} \longrightarrow \mathbb{R}^{n_\ell}$ on the output state vector from the affine mapping operation f_ℓ^M. The output vector y of the ℓ-th hidden layer is expressed as:

$$y_\ell = f_\ell(y_{\ell-1}) = f_\ell^\sigma \circ f_\ell^M(y_{\ell-1}) = f_\ell^\sigma(W_\ell y_{\ell-1} + b_\ell),$$

where $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $b_\ell \in \mathbb{R}^{n_\ell}$ are the weight matrix and the bias vector of the ℓ-th layer, respectively. In this paper, we are interested in FNNs with Sigmoid activation functions defined by $f^\sigma(x_i) = \frac{1}{1+e^{-x_i}}$ or TanH defined by $f^\sigma(x_i) = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}}$, where x_i is the i-th neuron of an input vector x. The input-output relation of the FNN can be written as:

$$y_k = \mathcal{F}(x), \ \mathcal{F} = f_k^M \circ f_{k-1}^\sigma \circ f_{k-1}^M \circ f_{k-2}^\sigma \circ f_{k-2}^M \circ \ldots \circ f_1^\sigma \circ f_1^M.$$

We note that the output layer is assumed to perform only an affine mapping operation, i.e., $f_k = f_k^M$, while other layers perform both affine mapping and activating operations, i.e., $f_\ell = f_\ell^\sigma \circ f_\ell^M, \forall \ell < k$.

Definition 2.1 (Reachability of an FNN). Given a bounded convex polytope input set $\mathcal{I} \triangleq \{x \mid Ax \leq b, x \in \mathbb{R}^{n_0}\}$, the reachability analysis of an FNN is to compute the output "reachable set" of the network defined recursively below.

$$\begin{aligned}
\mathcal{R}_1 &\triangleq \{y_1 \mid y_1 = f_1(x), x \in I\}, \\
\mathcal{R}_2 &\triangleq \{y_2 \mid y_2 = f_2(y_1), y_1 \in \mathcal{R}_1\}, \\
&\vdots \\
\mathcal{R}_k &\triangleq \{y_k \mid y_k = f_k(y_{k-1}), y_{k-1} \in \mathcal{R}_{k-1}\},
\end{aligned}$$

where the output reachable set $\mathcal{R}_k = \mathcal{F}(\mathcal{I})$ contains all the outputs of the FNN corresponding to all input vectors x in the input set $\mathcal{I}$.

Definition 2.2 (Generalized Star Set [32]). A generalized star set (or simply star) Θ is a tuple $\langle c, V, P\rangle$ created based on an input vector $x \in \Theta \subseteq \mathbb{R}^n$, where $c \in \mathbb{R}^n$ is a center vector, $V = [v_1, v_2, \cdots, v_m]$ consists of a set of m basis vectors $v \in \mathbb{R}^n$, and predicate $P : \mathbb{R}^m \rightarrow \{\top, \bot\}$. $P(\alpha) \triangleq C\alpha \leq d \wedge l_\alpha \leq \alpha \leq u_\alpha$ has a conjunction of p linear constraints, where $C \in \mathbb{R}^{p\times m}$, $d \in \mathbb{R}^p$, α is the predicate variable, and l_α and u_α are lower and upper predicate bounds. The set of states can be represented as:

$$[\![\Theta]\!] = \{x \mid x = c + \sum_{i=1}^{m} \alpha_i v_i \text{ such that } P(\alpha) = \top\}.$$

A star is an empty set, i.e., $\Theta = \emptyset$ if and only if the predicate $P(\alpha)$ is infeasible. Sometimes both the tuple Θ and the set of states $[\![\Theta]\!]$ are referred to as Θ.

We note that any polytope can be represented as a star set. Affine mapping and intersection with a half-space of a star set is another star set that can be computed efficiently via matrix-vector multiplication [32]. The efficiency of the star set in affine mapping and intersection with half-spaces makes it useful in the reachability analysis of ReLU networks. The star-based reachability approach can compute an overapproximation of the exact reachable set of a network by overapproximating the output of a ReLU activation on all neurons using three linear constraints, i.e., the triangle overapproximation rule [3]. This overapproximation reachability analysis requires the knowledge of the lower and upper bound vectors of a state vector x in a star set, i.e., $l \leq x \leq u$. The star set approach finds these bounds by solving $2n$ LP problems, i.e. $min(max)\, x[j] = c[j] + \Sigma_{i=1}^{m} \alpha_i v_i,\ s.t.\ C\alpha \leq d, 1 \leq j \leq n$. Solving LP problem is the main bottleneck of the star set approach, making the star set approach time-consuming and less scalable. Therefore, we apply symbolic intervals $\mathcal{D} \triangleq \langle \mathcal{D}^l, \mathcal{D}^u\rangle$ that consider two affine transformers for finding the state vector bounds via back-substitution. This approach provides enhanced scalability in the trade-off of precision.

Definition 2.3 (Symbolic Interval). Given a function $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$ defining an application of either an affine mapping, ReLU, Sigmoid or TanH on a state vector $x \in X \subseteq \mathbb{R}^n$, its corresponding symbolic interval is $\mathcal{D} \triangleq \langle \mathcal{D}^l, \mathcal{D}^u\rangle$ in which $\mathcal{D}^l = \langle D^l, g^l\rangle = D^l x + g^l$ and $\mathcal{D}^u = \langle D^u, g^u\rangle = D^u x + g^u$ are the lower and upper symbolic bounds, respectively. We have: $\forall x \in X, \mathcal{D}^l \leq f(x) \leq \mathcal{D}^u$.

Since an affine mapping operation is a composition of a linear transformation and a translation on a set, we do not need to overapproximate the affine mapping function. The lower and upper symbolic bounds of an affine mapping, defined in the following proposition, are the same and equal to the affine function.

PROPOSITION 2.4 (SYMBOLIC INTERVAL OF AN AFFINE MAPPING FUNCTION). *Symbolic interval of an affine mapping operation f^M, defined by a mapping matrix W and a bias vector b, on a state vector x is $\mathcal{D} = \langle \mathcal{D}^l, \mathcal{D}^u\rangle$, $\mathcal{D}^l = \mathcal{D}^u = f^M(x) = Wx + b$.*

PROOF. The linear bounds of the symbolic interval are affine transformers, and each of them can exactly represent an affine mapping. Hence $D^l = D^u = W$, and $g^l = g^u = b$. □

Symbolic intervals propagate to a symbolic input interval through a network to compute the lower bound and upper bound of a state vector x. These bounds are essential to construct the convex set and overapproximate Sigmoid/TanH. The key computation is to derive the tightest and optimal lower and upper symbolic constraints, i.e., $\mathcal{D}^l$ and $\mathcal{D}^u$, for each operation. This requires tight overapproximation rules for the sigmoidal activation function discussed in the next section.

3 OVERAPPROXIMATION OF SIGMOIDAL ACTIVATION FUNCTIONS

3.1 Star overapproximation of Sigmoid/TanH

Unlike affine mapping, the Sigmoid or TanH activation function f^σ is nonlinear. Therefore, the reachability analysis of this function is based on overapproximation, which leads to a key difference between the DeepPoly and our approaches proposed in this paper. The DeepPoly approach uses symbolic intervals (with two linear symbolic constraints) to overapproximate the sigmoidal activation function. However, it uses the lower and upper bounds of the states for verifying FNNs. A set of interval constraints, i.e., $X = \bigtimes_{i=1}^{m}[l_i, u_i]$, is used to verify the robustness of neural networks [28]. m is the number of operations in the FNN. Our star set approach, however, has the flexibility for users to choose the number of linear constraints used for tighter/looser overapproximation. Figure 1 shows three different ways of overapproximating the sigmoidal activation function.

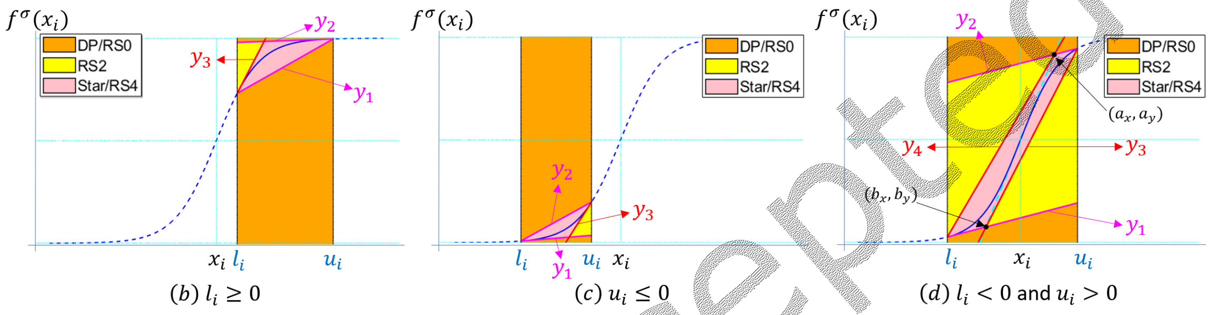


Fig. 1. **Overapproximation of TanH/Sigmoid**. DP and RS represent DeepPoly and RStar, respectively. RS2 is in yellow areas, which also includes the pink areas. Assume the computed state bounds from the LP solver and those from the back-substitution are identical. All RS methods and DP use y_1 and y_2 linear bounds to find the state bounds with back-substitution, while the star set applies all linear bounds to find the state bounds with the LP solver. Each method constrains its convex set with l_i and u_i bounds. RS2 additionally constrains its set with y_1 and y_2. Only the star set and RS4 constrain their convex set with all linear constraints and predicate bounds: $y_1, y_2, y_3, y_4, l_i, u_i$ for case (d), and y_1, y_2, y_3, l_i, u_i for cases (b) and (c). RS2 is bounded by y_1, y_2, l_i, and u_i, meaning RS2 includes Star/RS4. Thus, $RS4 \subseteq RS2 \subseteq RS0$. Since $l_i == u_i$ and $y_i = f^\sigma(l_i)$ is a single point, the case (a) is not plotted. The linear bounds are defined in Lemma 3.9 and 3.1. In terms of constraining an activation function, RS4 and Star are equivalent, but the star set uses an LP optimization, while RS4 uses symbolic intervals to compute the state bounds. RS0 and DP apply very similar overapproximation; however, they are different in verifying DNNs. Their reachable sets are presented as DP: $X = \bigtimes_{i=1}^{m}[l_i, u_i]$ and RS0: $\Omega = \{x \mid x \in [l_\ell, u_\ell]\}$.

Exploiting the convexity and concavity of the sigmoidal function on different input domains, we can overapproximate the function using two, three, or four linear constraints or simply by a rectangle. On the input regions where the sigmoidal function is either convex or concave (Figure 1-b,c), symbolic intervals overapproximate this function by two linear constraints y_1 and y_2, while the star set uses three constraints y_1, y_2 and y_3 for a tighter overapproximation. On other input regions where the sigmoidal function is neither convex nor concave (Figure 1-d), symbolic intervals still overapproximate the function by two linear constraints y_1 and y_2, while we apply four constraints y_1, y_2, y_3 and y_4 for a star set. Clearly, a star set approach is less conservative than symbolic intervals due to tighter overapproximation, which leads to the computation of tight upper and lower bounds.

Lemma 3.1 (Star Overapproximation Rule for a Sigmoidal Function). *For any input vector $x \in \mathbb{R}^n$, $l \leq x \leq u$ in which x_i is i-th individual state of x, the output $y = f^\sigma(x) \in \mathbb{R}^n$ satisfies the following rules:*
if $l_i == u_i$

$$y_i = f^\sigma(x_i) \tag{a}$$

$if\, l_i \geq 0$

$$\begin{cases} y_i \geq y_1 = \frac{f^{\sigma}(u_i) - f^{\sigma}(l_i)}{u_i - l_i} \times (x_i - l_i) + f^{\sigma}(l_i), \\ y_i \leq y_2 = f^{\sigma'}(u_i) \times (x_i - u_i) + f^{\sigma}(u_i), \\ y_i \leq y_3 = f^{\sigma'}(l_i) \times (x_i - l_i) + f^{\sigma}(l_i). \end{cases} \tag{b}$$

$if\, u_i \leq 0$

$$\begin{cases} y_i \geq y_1 = f^{\sigma'}(l_i) \times (x_i - l_i) + f^{\sigma}(l_i), \\ y_i \leq y_2 = \frac{f^{\sigma}(u_i) - f^{\sigma}(l_i)}{u_i - l_i} \times (x_i - l_i) + f^{\sigma}(l_i), \\ y_i \geq y_3 = f^{\sigma'}(u_i) \times (x_i - u_i) + f^{\sigma}(u_i). \end{cases} \tag{c}$$

$if\, l_i < 0\ and\, u_i > 0$

$$\begin{cases} y_i \geq y_1 = \lambda \times (x_i - l_i) + f^{\sigma}(l_i), \\ y_i \leq y_2 = \lambda \times (x_i - u_i) + f^{\sigma}(u_i), \\ y_i \geq y_3 = \mu_l \times x_i - \mu_l \times u_i + f^{\sigma}(u_i), \\ y_i \leq y_4 = \mu_u \times x_i - \mu_u \times l_i + f^{\sigma}(l_i). \end{cases} \tag{d}$$

$where\, \lambda = min(f^{\sigma'}(l_i), f^{\sigma'}(u_i)), \mu_u = \frac{f^{\sigma}(l_i) - a_y}{l_i - a_x}, \mu_l = \frac{f^{\sigma}(u_i) - b_y}{u_i - b_x}, a_y = f^{\sigma'}(0) \times a_x + f^{\sigma}(0), a_x = \frac{f^{\sigma}(u_i) - \lambda \times u_i - f^{\sigma}(0)}{f^{\sigma'}(0) - \lambda}, b_y$ $= f^{\sigma'}(0) \times b_x + f^{\sigma}(0), b_x = \frac{f^{\sigma}(l_i) - \lambda \times l_i - f^{\sigma}(0)}{f^{\sigma'}(0) - \lambda}.$

Proof. Since $f^{\sigma}(x_i)$ is a single point for case (a), no overapproximation is needed. For either Si gmoid or TanH, f^{σ} is a monotonically increasing function and $f^{\sigma'} > 0$. $f^{\sigma}(x)$ for $x_i \in [0, \infty)$ is convex and $f^{\sigma}(x_i)$ for $x_i \in (-\infty, 0]$ is concave. Therefore, $f^{\sigma}(x_i)$ is overapproximated by tangent lines at $f^{\sigma}(l_i)$, tangent line at $f^{\sigma}(u_i)$, and line connecting $f^{\sigma}(l_i)$ and $f^{\sigma}(u_i)$ for cases (b) and (c). Otherwise, $f^{\sigma}(x_i)$ for $x_i \in (-\infty, \infty)$ case (d) is neither convex nor concave. Since $\lambda = min(f^{\sigma'}(l_i), f^{\sigma'}(u_i))$, y_1 and y_2 overapproximate the function. Let $\boldsymbol{a}$ be the intersection point of a tangent line at $f^{\sigma'}(u_i)$ and a tangent line at $f^{\sigma'}(0)$. y_3 is a line connecting a point $f^{\sigma}(l_i)$ and a point $\boldsymbol{a}$. A similar idea applies to y_4. Hence, y_1, y_2, y_3, y_4 overapproximate the function; please refer Fig. 1. □

Example 3.2 (The star set on a TanH FNN). Given a 3-layers TanH FNN with a bounded input set $\mathcal{I} = \{x \mid -1 \leq x_0^1 \leq 1 \wedge -1 \leq x_0^2 \leq 1\}$, we demonstrate back-substitution of symbolic intervals. Here, x_j^i represents the i-th individual state of x at the j-th layer. The FNN has the following weights and biases:

$$W_1 = \begin{bmatrix} 0.2 & 0.8 \\ -0.1 & 0.4 \end{bmatrix}, W_2 = \begin{bmatrix} 0.3 & -0.4 \\ -0.1 & -0.8 \end{bmatrix}, b_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, b_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The star set. After an affine operation at the first layer, the star set Θ_1^M has $V = W$, $c = 0_{2\times1}$, and $P(\alpha) \triangleq C\alpha \leq d \wedge l_\alpha \leq \alpha \leq u_\alpha$ where $C = 0_{1\times2}$, $d = 0$, $l_\alpha = [-1, -1]^T$, $u_\alpha = [1, 1]^T$. The state vector, $x_1^M \in \Theta^M$, has the lower bound, $l_1^M = [-1, -0.5]^T$, and upper bound, $u_1^M = [1, 0.5]^T$. For the star set, these bounds are found with an LP solver. The state bounds satisfy the TanH overapproximation rule case (d), $l_2^M < 0$ and $u_2^M > 0$, in Lemma 3.9, 3.1. The star set applies four linear constraints to overapproximate TanH for case (d) in Lemma 3.1, i.e., $C^{\sigma}\alpha^{\sigma} \leq d^{\sigma}$, and the lower and upper predicate bounds, i.e., $l^\sigma_\alpha \leq \alpha^\sigma \leq u^\sigma_\alpha$.

$$C^\sigma = \begin{bmatrix} 0.0839 & 0.3359 & -1 & 0 \\ -0.0839 & -0.3359 & 1 & 0 \\ 0.1699 & 0.6799 & -1 & 0 \\ -0.1699 & -0.6799 & 1 & 0 \\ -0.0786 & 0.3145 & 0 & -1 \\ 0.0786 & -0.3145 & 0 & 1 \\ -0.0953 & 0.3815 & 0 & -1 \\ 0.0953 & -0.3815 & 0 & 1 \end{bmatrix}, d^\sigma = \begin{bmatrix} 0.3416 \\ 0.3416 \\ 0.0883 \\ 0.0883 \\ 0.0688 \\ 0.0688 \\ 0.0148 \\ 0.0148 \end{bmatrix}, l^\sigma_\alpha = \begin{bmatrix} -0.7615 \\ -0.4621 \end{bmatrix}, u^\sigma_\alpha = \begin{bmatrix} 0.7615 \\ 0.4621 \end{bmatrix}.$$

After an activating operation, Θ^σ_1 has $V = [0_{2\times2}, I_2]$, $c = 0_{2\times1}$, and $P(\alpha) \triangleq \Theta^M_1.P \wedge C^\sigma\alpha^\sigma \leq d^\sigma \wedge \leq l^\sigma_\alpha \leq \alpha^\sigma \leq u^\sigma_\alpha$, where I_n is an n-dimensional identity matrix. If the star applies two linear constraints instead (Lemma 3.9), then the constraints in blue color are discarded (this approach is used for RStar2 with symbolic intervals instead of LP solver; proposed in the section 3.3).

3.2 Symbolic interval overapproximation of Sigmoid/TanH

We have shown the overapproximation rule for the star set. Now, we derive how symbolic intervals overapproximate a sigmoidal activation function in the below.

Proposition 3.3 (Symbolic Interval of a Sigmoidal Activation Function). *For any input vectors $x \in \mathbb{R}^n$, $l \leq x \leq u$, the symbolic interval of the Sigmoid/TanH activation function f^σ on x is $\mathcal{D} = \langle D^l x + g^l, D^u x + g^u \rangle$, $D^l = diag(D^l_1, D^l_2, \ldots, D^l_n)$, $g^l = [g^l_1, g^l_2, \ldots, g^l_n]^T$, $D^u = diag(D^u_1, D^u_2, \ldots, D^u_n)$, $g^u = [g^u_1, g^u_2, \ldots, g^u_n]^T$, in which:*
if $l_i == u_i$

$$D^l_i = D^u_i = 0,\ g^l_i = g^u_i = f^\sigma(l_i). \tag{a}$$

if $l_i \geq 0$

$$\begin{cases} D^l_i = \frac{f^\sigma(u_i) - f^\sigma(l_i)}{u_i - l_i},\ g^l_i = f^\sigma(l_i) - D^l_i \times l_i, \\ D^u_i = f^{\sigma'}(u_i),\ g^u_i = f^\sigma(u_i) - D^u_i \times u_i. \end{cases} \tag{b}$$

if $u_i \leq 0$

$$\begin{cases} D^l_i = f^{\sigma'}(l_i),\ g^l_i = f^\sigma(l_i) - D^l_i \times l_i, \\ D^u_i = \frac{f^\sigma(u_i) - f^\sigma(l_i)}{u_i - l_i},\ g^u_i = f^\sigma(l_i) - D^u_i \times l_i. \end{cases} \tag{c}$$

if $l_i < 0$ and $u_i > 0$

$$\begin{cases} D^l_i = \lambda,\ g^l_i = f^\sigma(l_i) - D^l_i \times l_i, \\ D^u_i = \lambda,\ g^u_i = f^\sigma(u_i) - D^u_i \times u_i, \end{cases} \tag{d}$$

where $\lambda = min(f^{\sigma'}(l_i), f^{\sigma'}(u_i))$, $f^{\sigma'}$ is the first derivative of the function.

Proof. Based on Lemma 3.1, y_1 and y_2 are two linear functions that overapproximate a sigmoidal function. Since symbolic intervals are affine transformers, two linear functions are stored in the form of an affine transformation. □

Back-Substitution. One can see that the overapproximation is done based on the lower bound l and upper bound u of the state vector x. When x is in a star set, these bounds can be found by solving LPs (Section 2), which is time-consuming. To overcome this, we can use the back-substitution technique used in symbolic intervals to estimate these bounds without solving LPs. To estimate the lower bound and upper bound vectors (l^M_k, u^M_k) of the output at the k^{th} affine mapping operation x^M_k (i.e., $l^M_k \leq x^M_k \leq u^M_k$), we propagate the relationship from $x^M_k \rightarrow x^\sigma_{k-1} \rightarrow x^M_{k-1} \rightarrow \cdots \rightarrow x^\sigma_1 \rightarrow x^M_1 \rightarrow x_0$ using previously constructed symbolic intervals

$\{\mathcal{D}_k^M, \mathcal{D}_{k-1}^\sigma, \mathcal{D}_{k-1}^M, \ldots, \mathcal{D}_1^\sigma, \mathcal{D}_1^M\}$. Here, x_0 is the input to the network, $l_0 \leq x_0 \leq u_0$. The back-substitution procedure consists of multiple back-substitution steps since it propagates function by function. We note that the back-substitution of symbolic intervals is only applied to RStar sets defined in Definition 3.7. The original star set uses an LP solver to find the state bounds.

From Prop. 3.3, one can see that $D^{l,\sigma}$ and $D^{u,\sigma}$ are diagonal matrices. Since diagonal matrices are commutative, symbolic intervals of activation function and affine mapping can be composed into a single symbolic interval. For example, $\mathcal{D}_{k-1} = \mathcal{D}_{k-1}^\sigma \circ \mathcal{D}_{k-1}^M$ is the symbolic interval of the composed function $f_{k-1} = f_{k-1}^\sigma \circ f_{k-1}^M$ representing the polyhedral relationship within in the layer. Hence, the back-substitution can propagate layer by layer, reducing memory consumption and computation. The composition of two consecutive symbolic intervals is given in below.

Proposition 3.4. *The composition of two symbolic intervals $\mathcal{D}_1 = \langle (D_1^l, g_1^l), (D_1^u, g_1^u)\rangle$ and $\mathcal{D}_2 = \langle (D_2^l, g_2^l), (D_2^u, g_2^u)$ $\rangle$ of two consecutive operation f_1 and f_2 is given by:*

$$\begin{aligned}
&\mathcal{D}_{2\to1} = \mathcal{D}_2 \circ \mathcal{D}_1 = \langle (D_{21}^l, g_{21}^l), (D_{21}^u, g_{21}^u)\rangle,\\
&D_{21}^u = max(0, D_2^u) \cdot D_1^u + min(D_2^l, 0) \cdot D_1^l,\\
&g_{21}^u = g_2^u + max(0, D_2^u) \cdot g_1^u + min(D_2^l, 0) \cdot g_1^l,\\
&D_{21}^l = max(0, D_2^l) \cdot D_1^l + min(D_2^u, 0) \cdot D_1^u,\\
&g_{21}^l = g_2^l + max(0, D_2^l) \cdot g_1^l + min(D_2^u, 0) \cdot g_1^u.
\end{aligned}$$

Let x_0 be the input to the first operation f_1, we have:

$$D_{21}^l x_0 + g_{21} \leq f_2(f_1(x_0)) \leq D_{21}^u x_0 + g_{21}^u.$$

Proof. Considering back-substitution equations and proof in Prop. 3.5,

$$\mathcal{D}_2 \circ \mathcal{D}_1(x_0) = \mathcal{D}_{2\to1}(x_0) = \langle \mathcal{D}_{2\to1}^u, \mathcal{D}_{2\to1}^l\rangle(x_0),$$

$$\begin{aligned}
\mathcal{D}_{2\to1}^u(x_0) &= max(0, D_2^u) \cdot \mathcal{D}_1^u(x_0) + min(D_2^l, 0) \cdot \mathcal{D}_1^l(x_0) + g_2^u\\
&= max(0, D_2^u) \cdot D_1^u \cdot x_0 + min(D_2^l, 0) \cdot D_1^l \cdot x_0 +\\
&\quad g_2^u + max(0, D_2^u) \cdot g_1^u + min(D_2^l, 0) \cdot g_1^l\\
&= D_{21}^u \cdot x_0 + g_{21}^u,
\end{aligned}$$

$$\begin{aligned}
\mathcal{D}_{2\to1}^l(x_0) &= min(D_2^u, 0) \cdot \mathcal{D}_1^u(x_0) + max(0, D_2^l) \cdot \mathcal{D}_1^l(x_0) + g_2^l\\
&= min(D_2^u, 0) \cdot D_1^u \cdot x_0 + max(0, D_2^l) \cdot D_1^l \cdot x_0 +\\
&\quad g_2^l + min(D_2^u, 0) \cdot g_1^u + max(0, D_2^l) \cdot g_1^l\\
&= D_{21}^l \cdot x_0 + g_{21}^l.
\end{aligned}$$

□

We note that the composition of two symbolic intervals happens only between activating operation and affine mapping, i.e., $\mathcal{D}_{\ell-1} = \mathcal{D}_{\ell-1}^\sigma \circ \mathcal{D}_{\ell-1}^M$ since $D^{l,\sigma}$ and $D^{u,\sigma}$ are diagonal matrices having commutative properties.

Once two symbolic intervals are composed into a single symbolic interval, a set of symbolic intervals consists of $\{\mathcal{D}_k, \mathcal{D}_{k-1}, \ldots, \mathcal{D}_1\}$, where $\mathcal{D}_k = \mathcal{D}_k^M$. Using the *back-substitution* of these symbolic intervals, we can achieve lower bound and upper bound vectors (l_k, u_k) of a state vector x_k. The back-substitution propagation becomes $x_k \to x_{k-1} \to \cdots \to x_1 \to x_0$ and is described in the following.

Proposition 3.5 (Back-Substitution). *For*

$$l_\ell \leq \mathcal{D}^l_\ell(x_{\ell-1}) \leq x_\ell = f_\ell(x_{\ell-1}) \leq \mathcal{D}^u_\ell(x_{\ell-1}) \leq u_\ell,$$

backward propagation of a single layer can be achieved with the following equations:

$$u_\ell = max(0, D^u_\ell) \cdot u_{\ell-1} + min(D^l_\ell, 0) \cdot l_{\ell-1} + g^u_\ell, \tag{1}$$

$$l_\ell = min(D^u_\ell, 0) \cdot l_{\ell-1} + max(0, D^l_\ell) \cdot u_{\ell-1} + g^l_\ell. \tag{2}$$

In order to propagate backward all the way to the input layer, $u_{\ell-1} \longrightarrow u_{\ell-2} \longrightarrow \cdots \longrightarrow u_0$ and $l_{\ell-1} \longrightarrow l_{\ell-2} \longrightarrow \cdots \longrightarrow l_0$.

Proof.

$$\begin{aligned} u_\ell &\geq x_\ell = f_\ell(x_{\ell-1}) \\ &= max(0, D^u_\ell) \cdot u_{\ell-1} + min(D^l_\ell, 0) \cdot l_{\ell-1} + g^u_\ell \\ &\geq max(0, D^u_\ell) \cdot \mathcal{D}^u_{\ell-1}(x_{\ell-2}) + min(D^l_\ell, 0) \cdot \mathcal{D}^l_{\ell-1}(x_{\ell-2}) + g^u_\ell \\ &\geq max(0, D^u_\ell) \cdot f_{\ell-1}(x_{\ell-2}) + min(D^l_\ell, 0) \cdot f_{\ell-1}(x_{\ell-2}) + g^u_\ell \\ &= max(0, D^u_\ell) \cdot x_{\ell-1} + min(D^l_\ell, 0) \cdot x_{\ell-1} + g^u_\ell. \end{aligned}$$

$$\begin{aligned} l_\ell &\leq x_\ell = f_\ell(x_{\ell-1}) \\ &= min(D^u_\ell, 0) \cdot l_{\ell-1} + max(0, D^l_\ell) \cdot u_{\ell-1} + g^l_\ell \\ &\leq min(D^u_\ell, 0) \cdot \mathcal{D}^u_{\ell-1}(x_{\ell-2}) + max(0, D^l_\ell) \cdot \mathcal{D}^l_{\ell-1}(x_{\ell-2}) + g^l_\ell \\ &\leq min(D^u_\ell, 0) \cdot f_{\ell-1}(x_{\ell-2}) + max(0, D^l_\ell) \cdot f_{\ell-1}(x_{\ell-2}) + g^l_\ell \\ &= min(D^u_\ell, 0) \cdot x_{\ell-1} + max(0, D^l_\ell) \cdot x_{\ell-1} + g^l_\ell. \end{aligned}$$

□

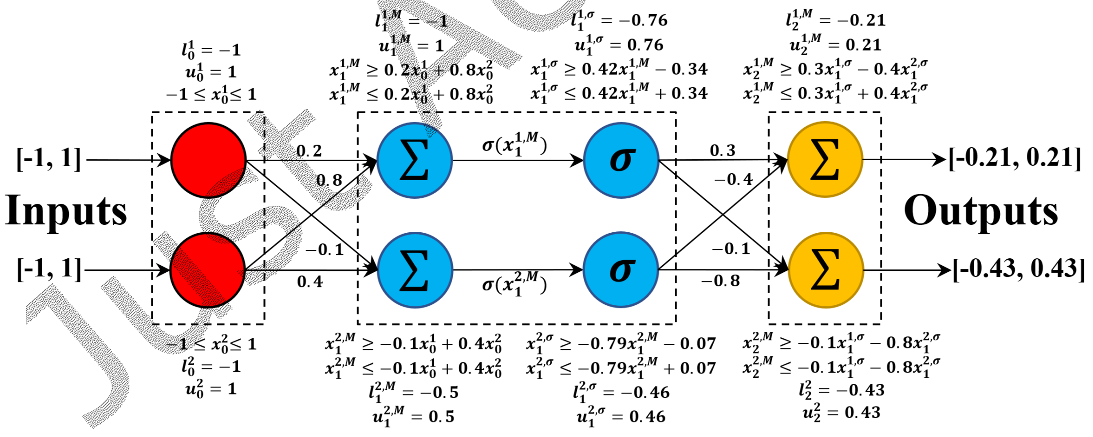
The lower and upper bounds of x^M_k are achieved using the back-substitution of the symbolic intervals at the output of the k^{th} affine mapping. With these bounds, we can construct the symbolic interval $\mathcal{D}^\sigma_k$ of the k^{th} activating operation using Prop. 3.3. This symbolic interval is composed of a symbolic interval of the k^{th} affine mapping, $\mathcal{D}^M_k$. The composed symbolic interval is then used (with previously constructed symbolic intervals) to calculate the lower bound and upper bound vectors (l_{k+1}, u_{k+1}) of x_{k+1}, the output state vector of the $(k+1)^{th}$ affine mapping. The back-substitution algorithm is shown in Algorithm 1. Based on eq. (1) the ubBackSub function propagates backward in lines 4-7 and finds the upper state bound in line 9. Similarly, the lbBacksub function applies back-substitution propagation in lines 14-17 and computes lower state bound using eq. (2) in line 19.

Example 3.6 (Symbolic Intervals on a TanH FNN). Consider the same FNN and input set as in Example 3.2. The predefined FNN is shown in Figure 2. We demonstrate the back-substitution of symbolic intervals. Here, x^i_j represents the i-th individual state of x at the j-th layer.

Symbolic Intervals. The symbolic intervals at affine mapping, $\mathcal{D}^M_1$ has $D^{l,M}_1 = D^{u,M}_1 = W$, and $g^{l,M}_1 = g^{u,M}_1 = b_1$ (Proposition 2.4). Following Proposition 3.3, the symbolic interval of the 1st layer TanH, $\mathcal{D}^\sigma_1 = \langle (D^{l,\sigma}_1, g^{l,\sigma}_1), (D^{u,\sigma}_1, g^{u,\sigma}_1) \rangle = \left\langle \left(\begin{bmatrix} 0.42 & 0 \\ 0 & 0.79 \end{bmatrix}, \begin{bmatrix} -0.34 \\ -0.07 \end{bmatrix} \right), \left(\begin{bmatrix} 0.42 & 0 \\ 0 & 0.79 \end{bmatrix}, \begin{bmatrix} 0.34 \\ 0.07 \end{bmatrix} \right) \right\rangle$, is constructed based on the lower and upper bounds of x^M_1, the output state vector of the 1st affine mapping, found by back-substitution. Then,

Algorithm 1 Back-substitution algorithm.

```
Input: $\mathcal{D}, l_0, u_0$        ▷ lower and upper symbolic constraints and input vector bounds
Output: $l, u$        ▷ upper and lower bounds
1: procedure $u$ = UBBACKSUB($\mathcal{D}^l, \mathcal{D}^u, l_0, u_0$)
2:     $A = D^u_k$        ▷ $k$ is number of functions, $length(\mathcal{D}^u)$
3:     $bl = 0_{n\times 1}, bu = g^u_k$        ▷ n is number of neurons of the layer
4:     for $i \leftarrow k$ to 0 do
5:         $A = min(A, 0) \cdot D^l_i + max(0, A) \cdot D^u_i$
6:         $bl = min(A, 0) \cdot g^l_i + bl$
7:         $bu = max(0, A) \cdot g^u_i + bu$
8:     end for
9:     $u = min(A, 0) \cdot l_0 + bl + max(0, A) \cdot u_0 + bu$
10: end procedure
11: procedure $l$ = LBBACKSUB($\mathcal{D}, l_0, u_0$)
12:     $A = D^l_k$        ▷ $k$ is number of functions, $length(D^l)$
13:     $bl = g^l_k, bu = 0_{n\times 1}$        ▷ n is number of neurons of the layer
14:     for $i \leftarrow k$ to 0 do
15:         $A = max(0, A) \cdot D^l_i + min(A, 0) \cdot D^u_i$
16:         $bl = max(0, A) \cdot g^l_i + bl$
17:         $bu = min(A, 0) \cdot g^u_i + bu$
18:     end for
19:     $l = max(0, A) \cdot l_0 + bl + min(A, 0) \cdot u_0 + bu$
20: end procedure
```



$l_0^1 = -1$
$u_0^1 = 1$
$-1 \le x_0^1 \le 1$
$l_1^{1,M} = -1$
$u_1^{1,M} = 1$
$x_1^{1,M} \ge 0.2x_0^1 + 0.8x_0^2$
$x_1^{1,M} \le 0.2x_0^1 + 0.8x_0^2$
$l_1^{1,\sigma} = -0.76$
$u_1^{1,\sigma} = 0.76$
$x_1^{1,\sigma} \ge 0.42x_1^{1,M} - 0.34$
$x_1^{1,\sigma} \le 0.42x_1^{1,M} + 0.34$
$l_2^{1,M} = -0.21$
$u_2^{1,M} = 0.21$
$x_2^{1,M} \ge 0.3x_1^{1,\sigma} - 0.4x_1^{2,\sigma}$
$x_2^{1,M} \le 0.3x_1^{1,\sigma} + 0.4x_1^{2,\sigma}$
[-1, 1]
Inputs
[-1, 1]
0.2
0.8
−0.1
0.4
$\sigma(x_1^{1,M})$
$\sigma(x_1^{2,M})$
0.3
−0.4
−0.1
−0.8
[-0.21, 0.21]
Outputs
[-0.43, 0.43]
$-1 \le x_0^2 \le 1$
$l_0^2 = -1$
$u_0^2 = 1$
$x_1^{2,M} \ge -0.1x_0^1 + 0.4x_0^2$
$x_1^{2,M} \le -0.1x_0^1 + 0.4x_0^2$
$l_1^{2,M} = -0.5$
$u_1^{2,M} = 0.5$
$x_1^{2,\sigma} \ge -0.79x_1^{2,M} - 0.07$
$x_1^{2,\sigma} \le -0.79x_1^{2,M} + 0.07$
$l_1^{2,\sigma} = -0.46$
$u_1^{2,\sigma} = 0.46$
$x_2^{2,M} \ge -0.1x_1^{1,\sigma} - 0.8x_1^{2,\sigma}$
$x_2^{2,M} \le -0.1x_1^{1,\sigma} - 0.8x_1^{2,\sigma}$
$l_2^2 = -0.43$
$u_2^2 = 0.43$

Fig. 2. An example of TanH FNN

$\mathcal{D}_1^\sigma$ is composed with $\mathcal{D}_1^M$ such that $\mathcal{D}_1 = \mathcal{D}_1^\sigma \circ \mathcal{D}_1^M = \left\langle \left(\begin{bmatrix} 0.08 & 0.34 \\ -0.08 & 0.31 \end{bmatrix}, \begin{bmatrix} -0.34 \\ -0.07 \end{bmatrix} \right), \left(\begin{bmatrix} 0.08 & 0.34 \\ -0.08 & 0.31 \end{bmatrix}, \begin{bmatrix} 0.34 \\ 0.07 \end{bmatrix} \right) \right\rangle$ (Proposition 3.4). Once the symbolic interval of the 2nd layer affine mapping is constructed, we can achieve the output bounds of the FNN by applying the back-substitution. Based on Proposition 3.5,

$$\begin{aligned} l_2^M &\leq min\left(\begin{bmatrix} 0.3 & -0.4 \\ -0.1 & -0.8 \end{bmatrix}, 0\right)\left(\begin{bmatrix} 0.08 & 0.33 \\ -0.08 & 0.31 \end{bmatrix} x_1 + \begin{bmatrix} 0.34 \\ 0.07 \end{bmatrix}\right) \\ &\quad + max\left(0, \begin{bmatrix} 0.3 & -0.4 \\ -0.1 & -0.8 \end{bmatrix}\right)\left(\begin{bmatrix} 0.08 & 0.33 \\ -0.08 & 0.31 \end{bmatrix} x_1 + \begin{bmatrix} -0.34 \\ -0.07 \end{bmatrix}\right) \\ &= \begin{bmatrix} 0.056 & -0.025 \\ 0.056 & -0.281 \end{bmatrix} x_1 + \begin{bmatrix} -0.13 \\ -0.09 \end{bmatrix} \\ &\leq min\left(\begin{bmatrix} 0.056 & -0.025 \\ 0.056 & -0.281 \end{bmatrix}, 0\right)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + max\left(0, \begin{bmatrix} 0.056 & -0.025 \\ 0.056 & -0.281 \end{bmatrix}\right)\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.21 \\ -0.43 \end{bmatrix}. \end{aligned}$$

The output bounds of the FNN are $l_2^M = \begin{bmatrix} -0.21, & -0.43 \end{bmatrix}^\top$ and $u_2^M = \begin{bmatrix} 0.21, & 0.43 \end{bmatrix}^\top$ with back-substitution of symbolic intervals. If the bounds are solved by an LP solver based on the star set that applies four linear constraints, $l_2^M = \begin{bmatrix} -0.15, & -0.43 \end{bmatrix}^\top$ and $u_2^M = \begin{bmatrix} 0.15, & 0.43 \end{bmatrix}^\top$. The star set achieved tighter bounds than the symbolic interval as it applied four linear constraints to overapproximate the TanH.

3.3 Relaxed-star overapproximation of Sigmoid/TanH

A star set can achieve the tightest bounds of the state vector by overapproximating the sigmoidal activation function with multiple linear constraints, albeit its scalability is limited due to linear programming optimization. In some cases, an LP solver computes the state bounds, which leads to a star set being an infeasible reachable set due to a floating point error in an LP solver. Therefore, taking advantage of the star set [32] and symbolic intervals that have the back-substitution, we introduce a relaxed star set representation. A relaxed star, defined in the following, is a star set [32] with symbolic intervals. Since the bounds of state variables computed by symbolic intervals are looser than those computed by an LP solver that considers multiple linear constraints, relaxation in terminology means "a star set with less tight state bounds."

Definition 3.7 (Relaxed Star Set). A relaxed star set (or simply RStar) Ω is a tuple $\langle \Theta, \mathcal{D}, l_0, u_0 \rangle$, where Θ is a star set, $\mathcal{D}$ is a set of symbolic intervals, l_0 and u_0 are input bounds for the back-substitution of symbolic intervals. The lower bound l and upper bound u of the state vector $x \in \Theta$ can be achieved by the back-substitution of symbolic intervals $\mathcal{D}$. The set of states can be represented as:

$$\llbracket \Omega \rrbracket = \{x \mid x = \Theta.c + \sum_{i=1}^{m} \alpha_i \Theta.v_i \text{ such that } \Theta.P(\alpha) = \top, l \leq x \leq u\}.$$

We note that an RStar is an empty set, *i.e.* $\Omega = \emptyset$ if the predicate $\Theta.P(\alpha)$ is infeasible. Sometimes both Ω and $\llbracket \Omega \rrbracket$ are referred as Ω.

Proposition 3.8 (Affine Mapping of a Relaxed Star). *Given a weight matrix $W \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and a bias vector $b \in \mathbb{R}^{n_\ell}$, the affine mapping of a RStar Ω is another RStar and can be defined as:*

$$\bar{\Omega} = \langle \langle \bar{c}, \bar{V}, \bar{P} \rangle, \bar{\mathcal{D}}, l_0, u_0 \rangle,$$

where $\bar{c} = Wc + b, \bar{V} = WV, \bar{P} \equiv P$, $\bar{\mathcal{D}} = \{\mathcal{D}_{1\ldots k}, \mathcal{D}_{k+1}\}, \mathcal{D}_{k+1} = \langle (W, b), (W, b) \rangle$.

Proof. Based on the definition of star set [32], after affine mapping, $\bar{\Theta} = \{y \mid y = Wc + b + \sum_{i=1}^{m}(\alpha_i W v_i)$ such that $P(\alpha) = \top\}$ implies another star with the center $\bar{c} = Wc + b$, basic vectors $\bar{V} = [Wv_1, Wv_2, \ldots, Wv_m]$, and has the same predicate as the star set, $\Omega.\Theta$. $\mathcal{D}_{k+1} = \langle (W, b), (W, b) \rangle$ implies another symbolic interval with upper symbolic constraint $D_{k+1}^u = (W, b)$ and lower symbolic constraint $D_{k+1}^l = (W, b)$. Hence, $\bar{\Omega} = \langle \bar{\Theta}, \bar{\mathcal{D}} \rangle = \{x \mid x =$

$\bar{\Theta}.c + \sum_{i=1}^{m} \alpha_i \bar{\Theta}.v_i$ such that $\bar{\Theta}.P(\alpha) = \top, \bar{l} \leq x \leq \bar{u}\}$ is another relaxed star, where a set of symbolic intervals $\bar{\mathcal{D}} = \{\mathcal{D}_{1...k}, \mathcal{D}_{k+1}\}$, $\bar{l}$ and $\bar{u}$ are new bounds of the state vector from the back-substitution of symbolic intervals. □

Lemma 3.9 (RStar2 Overapproximation Rule of a Sigmoidal Activation Function). *For any input vectors $x \in \mathbb{R}^n$, $l \leq x \leq u$ in which x_i is the i-th individual state of x, the output $y = f^{\sigma}(x) \in \mathbb{R}^n$ satisfies the following rules:*
if $l_i == u_i$

$$y_i = f^{\sigma}(l_i). \tag{a}$$

if $l_i \geq 0$

$$\begin{cases} y_i \geq y_1 = \frac{f^{\sigma}(u_i) - f^{\sigma}(l_i)}{u_i - l_i} \times (x_i - l_i) + f^{\sigma}(l_i), \\ y_i \leq y_2 = f^{\sigma'}(u_i) \times (x_i - u_i) + f^{\sigma}(u_i). \end{cases} \tag{b}$$

if $u_i \leq 0$

$$\begin{cases} y_i \geq y_1 = f^{\sigma'}(l_i) \times (x_i - l_i) + f^{\sigma}(l_i), \\ y_i \leq y_2 = \frac{f^{\sigma}(u_i) - f^{\sigma}(l_i)}{u_i - l_i} \times (x_i - l_i) + f^{\sigma}(l_i). \end{cases} \tag{c}$$

if $l_i < 0$ and $u_i > 0$

$$\begin{cases} y_i \geq y_1 = \lambda \times (x_i - l_i) + f^{\sigma}(l_i), \\ y_i \leq y_2 = \lambda \times (x_i - u_i) + f^{\sigma}(u_i). \end{cases} \tag{d}$$

where $\lambda = min(f^{\sigma'}(l_i), f^{\sigma'}(u_i))$, and $f^{\sigma'}$ is the first derivative of the function.

Proof. Based on Lemma 3.1, we know y_1 and y_2 overapproximate $f^{\sigma}(x_i)$ for all cases. Unlike the original star set, RStar2 considers only y_1 and y_2. □

Unlike the original star set, which requires overapproximation of an activation function with three constraints for cases (b) and (c) to compute the tightest state bounds, the star set inside the relaxed star, i.e., $\Theta \in \Omega$, doesn't necessarily need to apply the same overapproximation rule as the original star set. If the original star set applies the same two constraints as symbolic intervals for all cases, then the state bounds computed from an LP solver will be the same as those achieved from the symbolic intervals. However, the scalability of the star set does not increase. Therefore, the relaxed star uses symbolic intervals for finding bounds to improve scalability. In this paper, we propose three different variants of the relaxed star: RStar4, RStar2, and RStar0. The key difference between these methods lies in the overapproximation rule of a Sigmoid/TanH activation function using the star set. The relaxed star requires the star set to construct a convex polytope to perform neural network verification. We remind that the overapproximation of a star set involves the intersection between a star set and half-spaces (linear constraints). Since the relax star uses symbolic intervals to find the state bounds, it has the flexibility to constrain the activation function with two linear constraints (Lemma 3.9) for RStar2 or up to four linear constraints (Lemma 3.1) for RStar4. RStar0 is a special case in which only state vector bounds are used to generate the star set, i.e., $\Omega = \{x \,|\, x \in [l_\ell, u_\ell]\}$. The star set inside RStar0 does not affine map nor overapproximate an activating operation; it gets recreated after each function by the state bounds found from the back-substitution of symbolic intervals. In other words, RStar0 is an outer box of the RStar4 and RStar2 reachable set.

Example 3.10 (Relaxed star on a TanH FNN). We demonstrate the overapproximation of the relaxed star based on Example 3.2 and 3.6. Let Ω_4 and Ω_2 represent RStar4 and RStar2, respectively.

After the first affine mapping, $\Omega_2 = \Omega_4$ as their differences come during overapproximation. Since symbolic intervals are exact on affine mappings, no overapproximation error is present. $\Omega_2 = \langle \Theta_2, \mathcal{D}, l_0, u_0 \rangle$, where $\Theta_2 = \Theta_1^M$ (from Example 3.2), $\mathcal{D} = \{\mathcal{D}_1^M\}$, where $\mathcal{D}_1^M$ is defined in Example 3.6. For the overapproximation of TanH, both relaxed stars find the state bounds of x_1^M with symbolic intervals by back-substitution. $l_1^M = [-1, -0.5]^T$ and

$u_1^M = [1, 0.5]^T$. After the activation function, $\mathcal{D}_1 = \mathcal{D}_1^\sigma \circ \mathcal{D}_1^M$ (Example 3.6). We note that the bounds, l_1^σ and u_1^σ, are not found by the back-substitution since $l_1^\sigma = f_1^\sigma(l_1^M)$ and $u_1^\sigma = f_1^\sigma(u_1^M)$. If these bounds are computed by the back-substitution, then the bounds will have an overapproximation error. Symbolic intervals are incomplete on activation functions due to overapproximation. Since no overapproximation error is presented in the first layer, $\Omega_4.\Theta = \Theta_1^\sigma$ (Example 3.2) and $\Omega_4.\mathcal{D} = \{\mathcal{D}_1\}$ at the output of the activation function. For RStar2, $\Omega_2.\mathcal{D} = \{\mathcal{D}_1\}$ but $\Omega_2.\Theta$ has different predicate P. $P(\alpha) \triangleq \Theta_1^M.P \wedge C^\sigma \alpha^\sigma \leq d^\sigma \wedge \leq l_\alpha^\sigma \leq \alpha^\sigma \leq u_\alpha^\sigma$, where

$$C^\sigma = \begin{bmatrix} 0.0839 & 0.3359 & -1 & 0 \\ -0.0839 & -0.3359 & 1 & 0 \\ -0.0786 & 0.3145 & 0 & -1 \\ 0.0786 & -0.3145 & 0 & 1 \end{bmatrix}, d^\sigma = \begin{bmatrix} 0.3416 \\ 0.3416 \\ 0.0688 \\ 0.0688 \end{bmatrix}, l_\alpha^\sigma = \begin{bmatrix} -0.7615 \\ -0.4621 \end{bmatrix}, u_\alpha^\sigma = \begin{bmatrix} 0.7615 \\ 0.4621 \end{bmatrix}.$$

4 REACHABILITY ANALYSIS USING RELAXED STAR SETS

In this section, we present the algorithm for the reachability analysis of Sigmoid/TanH FNNs with the relaxed star and how it can be modified for other proposed approaches.

Overapproximate Reachability Algorithm. In Algorithm 2, the reachability analysis of sigmoidal FNNs using a relaxed star is achieved layer-by-layer. As mentioned in Section 2, each hidden layer contains an affine mapping (line 4) and a sigmoidal activation function (line 6). Since the relaxed star consists of a star set and a set of symbolic intervals, affine mapping of RStar is simply an affine mapping of a star set and symbolic intervals contained in RStar (Proposition 3.8). The APPROX_σ function overapproximates Sigmoid or TanH for both star set (line 13) and symbolic intervals (line 16, Proposition 3.3) using lower and upper bounds of the state vector x.

Algorithm 2 Relaxed Star Reachability of a FNN.

```
    Input: $\Omega = \langle \Theta, \mathcal{D}, l_0, u_0 \rangle, \mathcal{F} = \{(W_1, b_1), ..., (W_k, b_k)\}$        ▷ input set and FNN
    Output: $R$                                                                              ▷ the reachable output set
 1: procedure $R$ = REACH($\Omega, \mathcal{F}$)
 2:     $\mathcal{I}_\ell = \Omega$
 3:     for $i = 1 : k$ do
 4:         $\mathcal{I}_\ell = \mathcal{I}_\ell$.AFFINEMAP($W_i, b_i$)                                          ▷ [Prop. 3.8]
 5:         if i < k then
 6:             $\mathcal{I}_\ell$ =APPROX_$\sigma(\mathcal{I}_\ell)$
 7:         end if
 8:     end for
 9:     $R = \mathcal{I}_\ell$
10: end procedure
11: procedure $\Omega'$ = APPROX_$\sigma(\tilde{\Omega})$
12:     $[l, u] = \tilde{\Omega}.getRanges$                      ▷ get ranges of $x \in \tilde{\Omega}$, i.e. $l \leq x \leq u$ [Prop. 3.5]
13:     $\Theta' = \tilde{\Omega}.\Theta$.STAR_APPROX_$\sigma(l, u)$                                    ▷ [Lemma 3.1 or 3.9]
14:     $\mathcal{D}' = \{\mathcal{D}'_1, \ldots, \mathcal{D}'_m\} = \tilde{\Omega}.\mathcal{D}$
15:     $\mathcal{D}^M = \langle \mathcal{D}^{l,M}, \mathcal{D}^{u,M} \rangle = \mathcal{D}'_m$                                       ▷ $m = length(\mathcal{D})$
16:     $\mathcal{D}^\sigma = \langle \mathcal{D}^{l,\sigma}, \mathcal{D}^{u,\sigma} \rangle$ = SYMBOLIC_INTERVAL_APPROX_$\sigma(l, u)$                ▷ [Prop. 3.3]
17:     $\mathcal{D}'_m$ = COMPOSITION($\mathcal{D}^\sigma, \mathcal{D}^M$)                        ▷ $\mathcal{D}'_m = \mathcal{D}^\sigma \circ \mathcal{D}^M$ [Prop. 3.4]
18:     $\Omega' = \langle \Theta', \mathcal{D}' \rangle$
19: end procedure
```

Algorithm 2 Relaxed Star Reachability of a FNN. (continue)

```
20: procedure $\Theta' = \text{APPROX\_}\sigma(\tilde{\Theta}, l, u)$
21:     $n = \tilde{\Theta}.dim$    ▷ $n$ is the dimension of the star set
22:     for $i = 1 : n$ do
23:         $\mathcal{I}_\ell = \text{APPROX\_}\sigma_i(\mathcal{I}_\ell, i, l_i, u_i)$    ▷ [Lemma 3.1] (for Lemma 3.9 remove additional constraints)
24:     end for
25: end procedure
26: procedure $\tilde{R} = \text{APPROX\_}\sigma_i(\tilde{\Theta} = \langle \tilde{c}, \tilde{V}, \tilde{P} \rangle, i, l, u)$
27:     $M = [e_1\ e_2\ \cdots\ e_{i-1}\ 0\ e_{i+1}\ \cdots\ e_n]$
28:     $y_l = f^\sigma(l), y_u = f^\sigma(u), y'_l = f^{\sigma'}(l), y'_u = f^{\sigma'}(u)$
29:     if $l == u$ then
30:         $\tilde{c}[i] = y_l$
31:         $\tilde{R} = \langle \tilde{c}, M\tilde{V}, \tilde{P} \rangle$
32:     else
33:         $\tilde{P}(\alpha) \triangleq \tilde{C}\alpha \leq \tilde{d} \wedge \tilde{l}_\alpha \leq \alpha \leq \tilde{u}_\alpha, \alpha = [\alpha_1\ \alpha_2\ \cdots\ \alpha_m]^\intercal$    ▷ input set's predicate
34:         $\alpha' = [\alpha_1\ \alpha_2\ \cdots\ \alpha_m\ \alpha_{m+1}]^\intercal$    ▷ new variable $\alpha_{m+1}$
35:         $C_0 = [\tilde{C}\ \ 0_{m\times 1}], d_0 = \tilde{d}$
36:         $\lambda = (y_u - y_l)/(u - l)$
37:         if $l \geq 0$ then
38:             $C_1 = [\lambda\tilde{V}[I,:]\ \ \text{-}1], d_1 = \text{-}\lambda(\tilde{c}[i] - l) - y_l$    ▷ $\alpha_{m+1} \geq \lambda(x_i - l) + y_l$
39:             $C_2 = [\text{-}y'_u\tilde{V}[I,:]\ \ 1], d_2 = y'_u(\tilde{c}[i] - u) + y_u$    ▷ $\alpha_{m+1} \leq y'_u(x_i - u) + y_u$
40:             $C_3 = [\text{-}y'_l\tilde{V}[I,:]\ \ 1], d_3 = y'_l(\tilde{c}[i] - l) + yl$    ▷ $\alpha_{m+1} \leq y'_l(x_i - l) + y_l$
41:             $C' = [C_0; C_1; C_2; C_3], d' = [d_0; d_1; d_2; d_3]$
42:         else if $u \leq 0$ then
43:             $C_1 = [y'_l\tilde{V}[i,:]\ \ \text{-}1], d_1 = \text{-}y'_l(\tilde{c}[i] - l) - y_l$    ▷ $\alpha_{m+1} \geq y'_l(x_i - l) + y_l$
44:             $C_2 = [\text{-}\lambda\tilde{V}[i,:]\ \ 1], d_2 = \lambda(\tilde{c}[i] - l) + y_l$    ▷ $\alpha_{m+1} \leq \lambda(x_i - l) + y_l$
45:             $C_3 = [y'_u\tilde{V}[i,:]\ \ \text{-}1], d_3 = \text{-}y'_u(\tilde{c}[i] - u) - y_u$    ▷ $\alpha_{m+1} \geq y'_u(x_i - u) + y_u$
46:             $C' = [C_0; C_1; C_2; C_3], d' = [d_0; d_1; d_2; d_3]$
47:         else
48:             $\gamma = min(y'_l[i], y'_u[i])$
49:             $a_x = \frac{f^\sigma(u) - \gamma u - f^\sigma(0)}{f^{\sigma'}(0) - \gamma}, a_y = f^{\sigma'}(0)a_x + f^\sigma(0)$
50:             $b_x = \frac{f^\sigma(l) - \gamma l - f^\sigma(0)}{f^{\sigma'}(0) - \gamma}, b_y = f^{\sigma'}(0)b_x + f^\sigma(0)$
51:             $\mu_u = \frac{f^\sigma(l) - a_y}{l - a_x}, \mu_l = \frac{f^\sigma(u) - b_y}{u - b_x}$
52:             $C_1 = [\gamma\tilde{V}[i,:]\ \ \text{-}1], d_1 = \text{-}\gamma(\tilde{c}[i] - l) - y_l$    ▷ $\alpha_{m+1} \geq \gamma(x_i - l) + y_l$
53:             $C_2 = [\text{-}\gamma\tilde{V}[i,:]\ \ 1], d_2 = \gamma(\tilde{c}[i] - u) + y_u$    ▷ $\alpha_{m+1} \leq \gamma(x_i - u) + y_l$
54:             $C_3 = [\mu_l\tilde{V}[i,:]\ \ \text{-}1], d_3 = \text{-}\mu_l(\tilde{c}[i] - u) - f^\sigma(u)$    ▷ $\alpha_{m+1} \geq \mu_l(x_i - u) + f^\sigma(u)$
55:             $C_4 = [\text{-}\mu_u V[\tilde{i},:]\ \ 1], d_4 = \mu_u(\tilde{c}[i] - l) + f^\sigma(l)$    ▷ $\alpha_{m+1} \leq \mu_u(x_i - l) + f^\sigma(l)$
56:             $C' = [C_0; C_1; C_2; C_3; C_4], d' = [d_0; d_1; d_2; d_3; d_4]$
57:         end if
58:         $P'(\alpha') \triangleq C'\alpha' \leq d' \wedge l'_\alpha \leq \alpha' \leq u'_\alpha$    ▷ output set's predicate
59:         $c' = M\tilde{c}, V' = M\tilde{V}, V' = [V'\ e_i]$    ▷ $y[i] = f^\sigma(x_i) = a_{m+1}$
60:         $l'_\alpha = [\tilde{l}_\alpha; y_l], u'_\alpha = [\tilde{u}_\alpha; y_u]$
61:         $\tilde{R} = \langle c', V', P' \rangle$
62:     end if
63: end procedure
```

RStar uses the back-substitution of symbolic intervals to find the bounds of x with improved scalability over LPs. In the STAR_APPROX_σ function (line 13), as mentioned in the previous section, RStar4 applies up to four linear constraints to overapproximate activation functions following Lemma 3.1, while RStar2 applies two linear constraints, as shown in Lemma 3.9. Lines 40, 45, 54, and 55 are discarded for RStar2. Since RStar0 is an outer box of RStar4/RStar2, $\Theta' = Star(l, u)$ in line 13 for RStar0. In order to further reduce the number of back-substitution propagation, symbolic intervals of affine mapping, i.e., $\mathcal{D}^M$ and activation, i.e., $\mathcal{D}^\sigma$ are composed into a single symbolic interval (line 17) based on Proposition 2.4. A new RStar is created with an overapproximated star set and a set of symbolic intervals in line 18. In the case of the original star set reachability analysis, the state bounds in line 12 are solved by an LP solver, and an overapproximation rule applies based on Lemma 3.1 in line 13. All functionality of symbolic intervals, lines 14-17, are discarded for the original star set.

Example 4.1 (Reachability analysis of a TanH FNN). Given the same 3-layer TanH FNN in Example 3.2. Figure 3 visualizes the reachable intermediate set of each operation and the reachable output set of FNN for all methods.

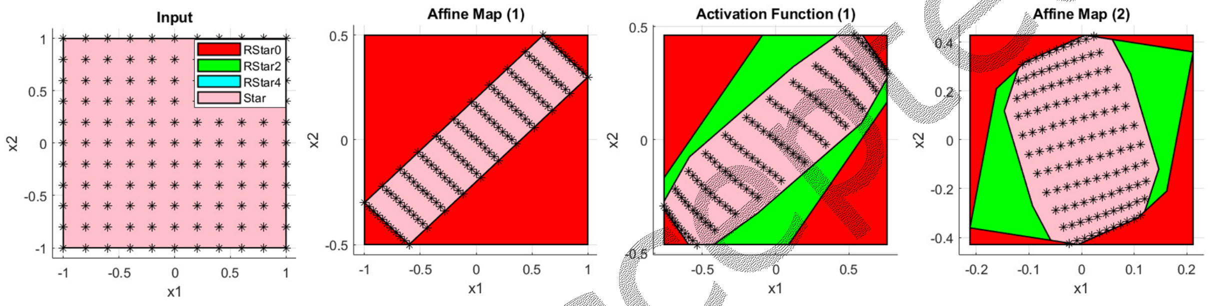


Fig. 3. Reachability analysis of TanH FNN

Initially, 121 star-shaped points are uniformly distributed within the convex input set. We consider these points to check the soundness and conservativeness of each method. For this example, although the bounds of the state variable at the output reachable set of RStar4 are more conservative than those of a star set, RStar4 has the same polyhedral configuration as a star set. If an LP solver is used instead of symbolic intervals, then the bounds of RStar4 at the reachable output set will be the same as those of a star set. In Figure 3, RStar4 (cyan color) matches exactly to a star set; therefore, it is not visible as a star set is displayed on top of all figures. We can clearly see that a star set and RStar4 are the least conservative method. RStar0 is the most conservative method.

Now, we straightforwardly show the reachable output sets of the star set and the relaxed star sets used for neural network verification. Let Θ, Ω_0, Ω_2, Ω_4 represent input sets of the original star set, RStar0, RStar2, and RStar4, respectively. $\bar{\Theta}$, $\bar{\Omega}_0$, $\bar{\Omega}_2$, $\bar{\Omega}_4$ represent the reachable output sets. In other words, $\bar{\Omega}_4 = \mathcal{F}(\Omega_4)$. Given a 3-layer TanH FNN addressed in Example 3.2, a bounded input set can be represented in the star set and relaxed star sets as follows:

$\Theta = \langle c, V, P(\alpha) \triangleq C\alpha \leq d \wedge l_\alpha \leq \alpha \leq u_\alpha \rangle$, where

$$c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 \end{bmatrix}, d = 0, l_\alpha = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, u_\alpha = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

For symbolic intervals, $\mathcal{D} = \emptyset$, $l_0 = [-1, -1]^T$, and $u_0 = [1, 1]^T$. Initially, $\Omega_0 = \Omega_2 = \Omega_4 = \langle \Theta, \mathcal{D}, l_0, u_0 \rangle$.

Once overapproximate reachability analysis proceeded through the network, the reachable output sets of each method at the output layer are shown below.

$\bar{\Theta} = \langle \bar{c}, \bar{V}, \bar{P}(\alpha) \triangleq \bar{C}\alpha \leq \bar{d} \wedge \bar{l}_\alpha \leq \alpha \leq \bar{u}_\alpha \rangle$:

$$\bar{c} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \bar{V} = \begin{bmatrix} 0 & 0 & 0.3 & -0.4 \\ 0 & 0 & -0.1 & -0.8 \end{bmatrix}, \bar{C} = \begin{bmatrix} 0.0839 & 0.3359 & -1 & 0 \\ -0.0786 & 0.3145 & 0 & -1 \\ -0.0839 & -0.3359 & 1 & 0 \\ 0.0786 & -0.3145 & 0 & 1 \\ 0.1699 & 0.6799 & -1 & 0 \\ -0.0953 & 0.3815 & 0 & -1 \\ -0.1699 & -0.6799 & 1 & 0 \\ 0.0953 & -0.3815 & 0 & 1 \end{bmatrix}, \bar{d} = \begin{bmatrix} 0.3416 \\ 0.0688 \\ 0.3416 \\ 0.0688 \\ 0.0883 \\ 0.0148 \\ 0.0883 \\ 0.0148 \end{bmatrix},$$

$$\bar{l}_\alpha = \begin{bmatrix} -1 \\ -1 \\ -0.7615 \\ -0.4621 \end{bmatrix}, \bar{u}_\alpha = \begin{bmatrix} 1 \\ 1 \\ 0.7615 \\ 0.4621 \end{bmatrix}.$$

Since the relaxed star sets of each method use the same symbolic intervals algorithms on an affine mapping and an activation function, each relaxed star set contains equivalent symbolic intervals, i.e., $\bar{\mathcal{D}}$, at the output layer.

$\bar{\mathcal{D}} = \{\mathcal{D}_2, \mathcal{D}_1\}$, where $\mathcal{D}_1 = \langle (D_1^l, g_1^l), (D_1^u, g_1^u) \rangle$, $\mathcal{D}_2 = \langle (D_2^l, g_2^l), (D_2^u, g_2^u) \rangle$,

$$D_1^l = \begin{bmatrix} 0.0839 & 0.3359 \\ -0.0786 & 0.3145 \end{bmatrix}, g_1^l = \begin{bmatrix} -0.3416 \\ -0.0688 \end{bmatrix}, D_1^u = \begin{bmatrix} 0.0839 & 0.3359 \\ -0.0786 & 0.3145 \end{bmatrix}, g_1^u = \begin{bmatrix} 0.3416 \\ 0.0688 \end{bmatrix},$$

$$D_2^l = \begin{bmatrix} 0.3 & -0.4 \\ -0.1 & -0.8 \end{bmatrix}, g_2^l = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, D_2^u = \begin{bmatrix} 0.3 & -0.4 \\ -0.1 & -0.8 \end{bmatrix}, g_2^u = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

$\bar{\Omega}_0 = \langle \bar{\Theta}_0, \bar{\mathcal{D}}, l_0, u_0 \rangle$, where $\bar{\Theta}_0 = \langle \bar{c}_0, \bar{C}_0, \bar{P}_0(\alpha) \triangleq \bar{C}_0\alpha \leq \bar{d}_0 \wedge \bar{l}_{\alpha 0} \leq \alpha \leq \bar{u}_{\alpha 0} \rangle$,

$$\bar{c}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \bar{V}_0 = \begin{bmatrix} 0.2117 & 0 \\ 0 & 0.4290 \end{bmatrix}, \bar{C}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}, \bar{d}_0 = 0, \bar{l}_{\alpha 0} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \bar{u}_{\alpha 0} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

$\bar{\Omega}_2 = \langle \bar{\Theta}_2, \bar{\mathcal{D}}, l_0, u_0 \rangle$, where $\bar{\Theta}_2 = \langle \bar{c}_2, \bar{C}_2, \bar{P}_2(\alpha) \triangleq \bar{C}_2\alpha \leq \bar{d}_2 \wedge \bar{l}_{\alpha 2} \leq \alpha \leq \bar{u}_{\alpha 2} \rangle$,

$$\bar{c}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \bar{V}_2 = \begin{bmatrix} 0 & 0 & 0.3 & -0.4 \\ 0 & 0 & -0.1 & -0.8 \end{bmatrix}, \bar{C}_2 = \begin{bmatrix} 0.0839 & 0.3359 & -1 & 0 \\ -0.0839 & -0.3359 & 1 & 0 \\ -0.0786 & 0.3145 & 0 & -1 \\ 0.0786 & -0.3145 & 0 & 1 \end{bmatrix}, \bar{d}_2 = \begin{bmatrix} 0.3416 \\ 0.3416 \\ 0.0688 \\ 0.0688 \end{bmatrix},$$

$$\bar{l}_{\alpha 2} = \begin{bmatrix} -1 \\ -1 \\ -0.7615 \\ -0.4621 \end{bmatrix}, \bar{u}_{\alpha 2} = \begin{bmatrix} 1 \\ 1 \\ 0.7615 \\ 0.4621 \end{bmatrix}.$$

$\bar{\Omega}_4 = \langle \bar{\Theta}_4, \bar{\mathcal{D}}, l_0, u_0 \rangle$, where $\bar{\Theta}_4 = \langle \bar{c}_4, \bar{C}_4, \bar{P}_4(\alpha) \triangleq \bar{C}_4\alpha \leq \bar{d}_4 \wedge \bar{l}_{\alpha 4} \leq \alpha \leq \bar{u}_{\alpha 4} \rangle$,

$$\bar{c}_4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \bar{V}_4 = \begin{bmatrix} 0 & 0 & 0.3 & -0.4 \\ 0 & 0 & -0.1 & -0.8 \end{bmatrix}, \bar{C}_4 = \begin{bmatrix} 0.0839 & 0.3359 & -1 & 0 \\ -0.0839 & -0.3359 & 1 & 0 \\ 0.1699 & 0.6799 & -1 & 0 \\ -0.1699 & -0.6799 & 1 & 0 \\ -0.0786 & 0.3145 & 0 & -1 \\ 0.0786 & -0.3145 & 0 & 1 \\ -0.0953 & 0.3815 & 0 & -1 \\ 0.0953 & -0.3815 & 0 & 1 \end{bmatrix}, \bar{d}_4 = \begin{bmatrix} 0.3416 \\ 0.3416 \\ 0.0883 \\ 0.0883 \\ 0.0688 \\ 0.0688 \\ 0.0148 \\ 0.0148 \end{bmatrix},$$

$$\bar{l}_{\alpha 4} = \begin{bmatrix} -1 \\ -1 \\ -0.7615 \\ -0.4621 \end{bmatrix}, \bar{u}_{\alpha 4} = \begin{bmatrix} 1 \\ 1 \\ 0.7615 \\ 0.4621 \end{bmatrix}.$$

5 EVALUATION

Experimental Setup. In this section, we evaluate the proposed reachability algorithms based on the original star set and the relaxed star approaches compared to an existing overapproximate method DeepPoly [28]. The proposed reachability algorithms are implemented in NNV [35], a tool for verifying deep neural networks and learning-enabled CPS. To ensure fair and exact comparisons, we verify the robustness of our FNNs with the star set and RStar approaches in the NNV and DeepPoly method via the ERAN [21] toolbox. In this experiment, AbsDom LP represents the star-based approach of RStar2 (two linear constraints) using an LP solver instead of the back-substitution. r_{RS} represents the verification time improvement of the RStar0 over the star set, r_{RD} represents the verification time improvement of the RStar0 over DeepPoly, and '——' denotes time-out in Tables 2 and 3. An experiment is considered time-out if it takes more than 5 hours to verify 100 random images per δ. In Figures 4 and 5, (N) represents NNV, and (E) represents ERAN. We use two image datasets for our evaluation: MNIST [17], and CIFAR10 [16]. We note that MATLAB linprog optimizer is used for the star-set approach for the MNIST dataset only as part of reachability analysis due to the in-feasibility of finding an optimal solution via the gurobi optimizer [7]. Otherwise, the gurobi optimizer is applied for the reachability analysis. This experiment is done on a computer with the following configurations: Intel Core i7-10700 CPU @ 2.90GHz × 8 Processor, 63.7 GiB Memory, 64-bit Ubuntu 18.04.6 LTS OS.

5.1 Robustness Certification of Image Classification DNNs

MNIST dataset consists of 28×28 handwritten images from 0 to 9. The CIFAR10 dataset contains 32×32 RGB images of 10 categories, such as ship, truck, bird, etc. A network is considered robust to an input set if the classified label of an image holds for any perturbation attack. To verify the robustness of the neural networks, we perform the L_∞ norm attack on images correctly classified by the respective neural network and randomly select 100 images that can be verified at least by one approach at $\delta = 0.01$ for each network. L_∞ norm attack [1] applies some bounded disturbance δ to the normalized input vector x, i.e., $||x - x'||_\infty \leq \delta$, where x' is the perturbed image. Then, we compute the reachable output sets for each perturbed image to prove the robustness of the neural network. The neural network is classified as robust against L_∞ norm attack if $y_c - y_i > 0$, where c is the correctly classified index and $y_i \in \mathcal{R}_k - \{y_c\}$. Since we only need to determine that the correctly classified output always has the maximum value in the set compared to other outputs, the *softmax* function is neglected in our experiments. For each dataset, we train a set of three image classification FNNs with different architectures: small, medium, and big. The architecture of these networks is shown in Table 1. For instance, a small MNIST model has 2 hidden layers and 200 hidden units, which means each hidden layer has 100 units. All neural networks trained based on the MNIST dataset set have a test accuracy of at least 94.9%, and those trained with CIFAR10 have a test accuracy of at most 43.67% with unnormalized input data.

Table 1. The architecture of neural networks.

Dataset	Model	#Hidden Layers	#Hidden units
MNIST	Small	2	200
	Medium	6	900
	Large	4	1600
CIFAR10	Small	2	200
	Medium	6	900
	Large	4	1600

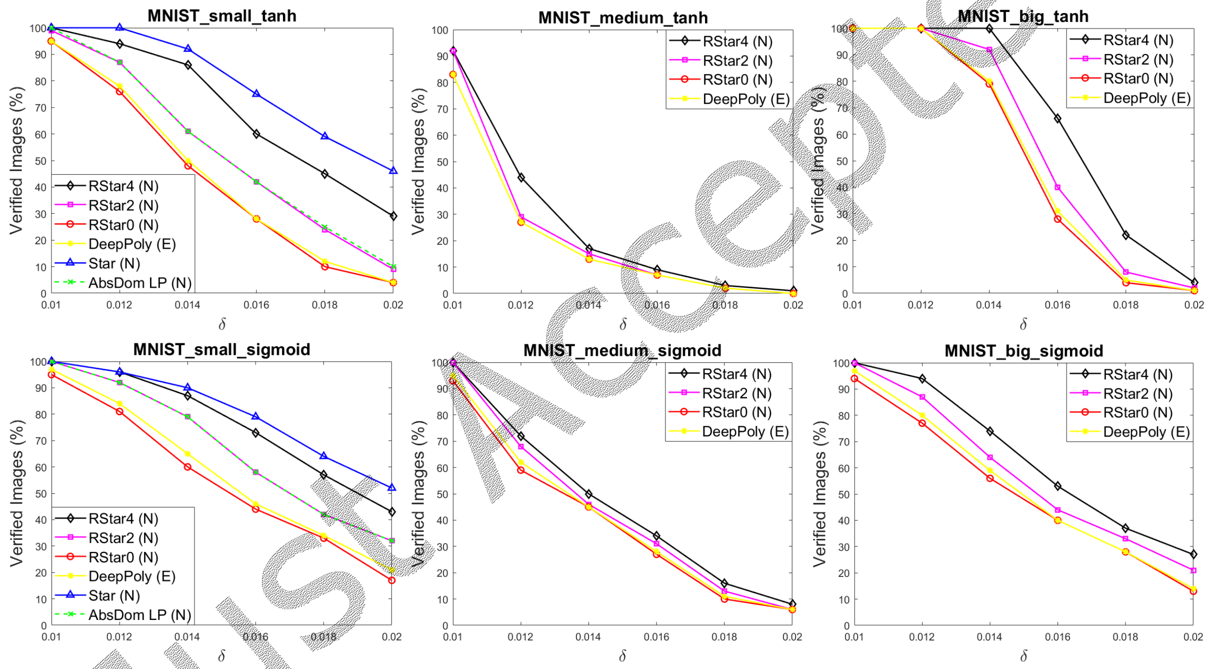


Fig. 4. Robustness verification of MNIST neural networks.

5.2 Verification Results and Time performance.

We are interested in the verification time and the number of images that can be verified by each method to guarantee the robustness of neural networks under different disturbances.

Scalability. Non-linear activation functions such as TanH and Sigmoid are bounded and have saturation regions. For example, TanH saturates inputs to range between [-1, 1]. Upper bounds tend to result closer to each other, approaching the value of 1 as reachability analysis passes through the layers. In contrast, lower bounds result closer to each other, approaching the value of -1. Due to these behaviors, the computation time of the LP

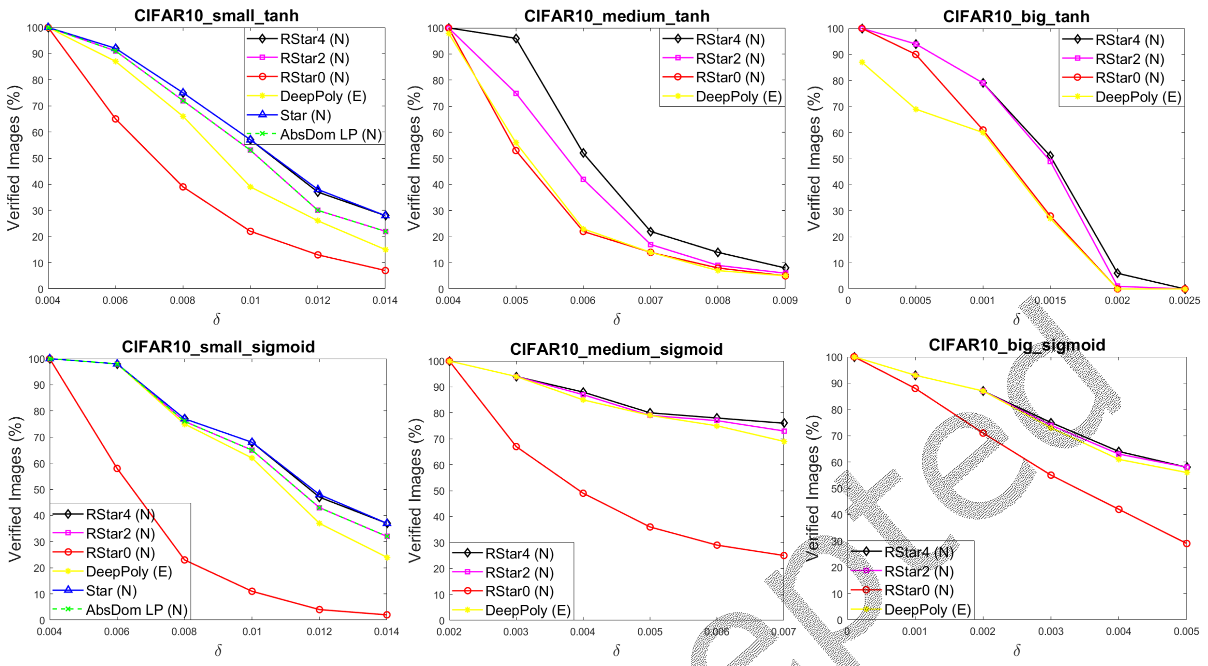


Fig. 5. Robustness verification of CIFAR10 neural networks.

solver to compute the state vector's bounds increases for deeper layers. In the worst case, LP solvers could not find feasible bounds. As a result, the star set approach does not persist in scalability for deeper neural networks (medium and big) with these activation functions. However, the RStar approaches maintain improved scalability by acquiring the back-substitution throughout the experiment.

Time performance. The computation time of the reachability analysis of RStar is significantly reduced by using symbolic intervals with back-substitution. In our experiment, RStar0 verifies neural networks up to 3528 times faster than the original star set approach. Our most computationally expensive relaxed star approach, RStar4, verified on average 51 times faster than the star set. For MNIST FNNs, RStar0 verifies a number of images close to that of DeepPoly, but it verifies up to 20 times faster than DeepPoly. For CIFIAR10 FNNs, RStar0 verifies neural networks up to 46 times faster than DeepPoly. In general, both RStar2 and RStar4 verify faster than the star set but slower than DeepPoly. RStar0 is the fastest approach among all methods to verify DNNs. Unlike all other methods, the relaxed star approaches can choose a number of linear constraints for overapproximation to trade off computation time and scalability over precision and conservativeness.

Conservativness. Figures 4 and 5 show the number of images each method verifies on CIFAR10 and MNIST DNNs. Both star set and RStar4 apply up to four linear constraints to overapproximate the sigmoidal activation function (Lemma 3.1). However, RStar4 is more conservative than the star set as it finds the state bounds using symbolic intervals with two linear constraints on each activating operation. The star set uses all bounds proposed in Fig. 1, i.e., $l_i, u_i, y_1, y_2, y_3, y_4$, on each activating operation with an LP solver to the state bounds. As expected, the star set verifies the most images due to solving multiple linear constraints based on the LP solver. Hence, the star set is the least conservative method, and $Star \subseteq RStar4 \subseteq RStar2 \subseteq RStar0$. DeepPoly applies two linear constraints to find a state vector's lower and upper bounds. However, it uses only the lower and upper bounds that overapproximate the two symbolic intervals when it comes to neural network verification. A set of interval

Table 2. Verification time of MNIST neural networks.

Model	δ	Verification Time of 100 images (sec)							
		Star	AbsDom LP	RStar4	RStar2	RStar0	DeepPoly	r_{RS}	r_{RD}
Small TanH	0.010	1960.31	2312.60	26.39	13.36	1.43	7.17	**1370.84** ×	**5.01** ×
	0.012	1999.55	2323.64	31.42	15.42	1.43	7.87	**1398.29** ×	**5.50** ×
	0.014	2015.20	2333.79	41.76	17.25	1.41	9.09	**1429.22** ×	**6.45** ×
	0.016	2059.38	2253.97	49.45	19.84	1.40	9.92	**1470.98** ×	**7.09** ×
	0.018	2131.21	2289.73	59.76	21.25	1.43	10.37	**1490.36** ×	**7.25** ×
	0.020	2222.42	2284.14	58.34	21.06	1.49	10.69	**1491.56** ×	**7.17** ×
Med TanH	0.010	−−	−−	368.85	215.37	6.95	75.31		**10.84** ×
	0.012	−−	−−	461.19	234.28	6.69	107.77		**16.11** ×
	0.014	−−	−−	474.15	229.68	6.65	117.19		**17.62** ×
	0.016	−−	−−	482.83	227.95	6.68	120.59		**18.05** ×
	0.018	−−	−−	492.67	223.52	6.68	125.99		**18.86** ×
	0.020	−−	−−	497.44	218.41	6.59	127.63		**19.37** ×
Big TanH	0.010	−−	−−	2489.53	1452.67	23.42	243.37		**10.39** ×
	0.012	−−	−−	2598.16	1255.24	22.82	250.23		**10.97** ×
	0.014	−−	−−	2608.51	1512.07	22.58	291.11		**12.89** ×
	0.016	−−	−−	2947.90	1054.31	22.84	391.98		**17.16** ×
	0.018	−−	−−	2803.49	1061.04	23.78	463.65		**19.49** ×
	0.020	−−	−−	2931.64	1047.88	25.04	464.72		**18.56** ×
Small Sigmoid	0.010	2083.11	2257.16	27.47	9.67	1.56	6.824	**1335.32** ×	**4.37** ×
	0.012	2102.27	2200.89	29.72	15.31	1.47	7.359	**1430.11** ×	**5.01** ×
	0.014	2122.31	2212.71	36.18	26.18	1.44	8.181	**1473.83** ×	**5.68** ×
	0.016	2128.31	2253.01	43.68	32.30	1.42	8.919	**1498.81** ×	**6.28** ×
	0.018	2168.24	2230.74	50.07	38.75	1.44	9.286	**1505.72** ×	**6.45** ×
	0.020	2191.55	2203.99	56.38	44.95	1.46	9.694	**1501.06** ×	**6.64** ×
Med Sigmoid	0.010	−−	−−	369.28	223.44	6.61	71.88		**10.87** ×
	0.012	−−	−−	426.42	233.75	6.90	93.56		**13.56** ×
	0.014	−−	−−	447.59	236.06	6.96	103.44		**14.86** ×
	0.016	−−	−−	478.85	233.16	7.00	114.65		**16.38** ×
	0.018	−−	−−	505.24	237.95	6.77	123.36		**18.22** ×
	0.020	−−	−−	503.56	238.86	6.58	125.54		**19.07** ×
Big Sigmoid	0.010	−−	−−	2694.19	1049.52	24.93	273.225		**10.96** ×
	0.012	−−	−−	3016.54	1069.88	24.23	311.305		**12.85** ×
	0.014	−−	−−	2997.89	1106.53	24.90	357.426		**14.35** ×
	0.016	−−	−−	3299.61	1128.53	23.30	399.484		**17.15** ×
	0.018	−−	−−	3186.45	1141.06	23.68	425.632		**17.97** ×
	0.020	−−	−−	3415.57	1142.41	23.01	458.542		**19.93** ×

constraints, i.e., $X = \bigtimes_{i=1}^{m} [l_i, u_i]$, is used to verify the robustness of the neural network [28]. m is the number of operations in the DNN. Therefore, DeepPoly is more conservative than RStar2. Given k-layers FNN, the reachable output set with RStar0 is $\Omega = \{x \mid x \in [l_k, u_k]\}$, meaning it only considers the state bounds at the output layer. Since the symbolic interval approach is one of the abstract interpolation methods, and abstract transformers in DeepPoly apply the same linear bounds y_1 and y_2, we expect DeepPoly to be less conservative than RStar0. According to Fig 4 and 5, DeepPoly verifies more images than RStar0 generally. However, we notice that DeepPoly proves fewer images than RStar0 on the CIFAR10 medium TanH network for $\delta = 0.008$ and CIFAR10 big TanH network for $\delta = \{0.0001, 0.001, 0.0015\}$. We believe there is some bug in DeepPoly.

Table 3. Verification time of CIFAR10 neural networks.

Model	δ	Verification Time of 100 images (sec)							
		Star	AbsDom LP	RStar4	RStar2	RStar0	DeepPoly	r_{RS}	r_{RD}
Small TanH	0.004	3112.38	2018.85	64.34	43.62	2.06	20.30	**1510.86** ×	**9.85** ×
	0.006	3201.92	2107.30	70.90	47.31	2.14	21.96	**1496.22** ×	**10.26** ×
	0.008	3298.00	2151.06	78.97	55.33	2.26	26.14	**1459.29** ×	**11.57** ×
	0.010	3398.84	2181.15	92.20	58.79	2.18	30.25	**1559.10** ×	**13.88** ×
	0.012	3455.41	2186.51	93.22	57.92	2.02	33.75	**1710.59** ×	**16.71** ×
	0.014	3527.98	2196.18	97.52	59.93	2.02	37.11	**1746.52** ×	**18.37** ×
Med TanH	0.004	−−	−−	1526.09	559.85	9.83	223.80		**22.77** ×
	0.005	−−	−−	1770.22	873.69	10.60	293.72		**27.71** ×
	0.006	−−	−−	1874.22	620.33	9.63	345.78		**35.91** ×
	0.007	−−	−−	1896.33	908.76	10.22	361.63		**35.38** ×
	0.008	−−	−−	1947.69	594.78	9.53	374.66		**39.31** ×
	0.009	−−	−−	1999.03	873.76	9.48	392.00		**41.35** ×
Big TanH	0.0001	−−	−−	14709.54	5620.24	31.37	845.36		**26.95** ×
	0.0005	−−	−−	9362.07	5703.62	31.30	1011.02		**32.30** ×
	0.0010	−−	−−	10408.27	6098.98	31.76	1133.33		**35.68** ×
	0.0015	−−	−−	11034.17	6296.57	31.75	1353.34		**42.62** ×
	0.0020	−−	−−	15664.43	6296.57	31.90	1428.80		**44.79** ×
	0.0025	−−	−−	11781.70	6880.26	31.17	1439.27		**46.17** ×
Small Sig.	0.004	6880.94	4669.78	81.06	53.03	1.95	25.44	**3528.68** ×	**13.05** ×
	0.006	6870.19	4770.13	96.80	62.33	2.03	26.65	**3384.33** ×	**13.13** ×
	0.008	6875.68	4835.67	113.62	72.54	2.11	29.43	**3258.61** ×	**13.95** ×
	0.010	6870.88	4897.19	129.60	80.57	2.15	32.22	**3195.75** ×	**14.99** ×
	0.012	6793.01	4927.98	137.37	81.83	2.17	37.86	**3130.41** ×	**17.45** ×
	0.014	6793.60	4952.63	142.23	86.61	2.18	40.30	**3116.33** ×	**18.49** ×
Med Sigmoid	0.002	−−	−−	1455.14	730.53	10.01	226.87		**22.66** ×
	0.003	−−	−−	1558.18	767.03	10.35	242.19		**23.40** ×
	0.004	−−	−−	1508.76	824.71	11.18	261.67		**23.41** ×
	0.005	−−	−−	1649.89	981.90	11.61	273.46		**23.55** ×
	0.006	−−	−−	1684.02	890.23	11.24	282.25		**25.11** ×
	0.007	−−	−−	1736.53	949.26	11.33	295.17		**26.05** ×
Big Sigmoid	0.0001	−−	−−	8323.97	5401.24	33.64	841.84		**25.02** ×
	0.0010	−−	−−	8840.37	5436.69	31.99	907.47		**28.37** ×
	0.0020	−−	−−	9097.82	5618.01	31.78	956.15		**30.09** ×
	0.0030	−−	−−	9295.44	5404.40	31.86	1063.38		**33.38** ×
	0.0040	−−	−−	9545.42	5517.04	32.58	1156.23		**35.49** ×
	0.0050	−−	−−	9762.18	5719.41	32.19	1192.44		**37.04** ×

Verification results. The experiments show that the star-based approach verifies the most images. However, due to the time restriction of the experiment, it could not verify all networks with architecture larger than a small size. For the rest of the networks, RStar4 verifies the most images. In Figures 4 and 5, we can see that RStar2 verifies almost the same number of images as AbsDom LP (N). On the MNIST dataset, AbsDom LP (N) verifies at most one more image than RStar2 as it uses MATLAB linprog optimizer instead of gurobi optimizer. If the state bounds of RStar2 are computed based on MATLAB linprog, RStar2 verifies the same number of images as AbsDom LP. On the MNIST dataset, the star set verifies at most 17% more images than RStar4, while on the CIFAR10 dataset, it verifies at most 1% more. These results show that RStar4 holds its precision on the larger dataset. RStar4 verifies up to 40% more images against L_∞ norm attack than DeepPoly. RStar0 performs very poorly on CIFAR10 neural networks with Sigmoid activation function. However, the precision of RStar0 is very close to the precision of DeepPoly for MNIST NNs and CIFAR10 TanH NNs (medium and big). Except for CIFAR10 Sigmoid NNs and small CIFAR10 TanH NN, RStar0 verifies at most 5% fewer images than DeepPoly in the worst case.

6 RELATED WORK

Verification of DNNs receives great attention with many approaches proposed for feedforward neural networks (FNNs) and convolutional neural networks (CNNs) such as polytope [33, 40, 43], star [32, 34], ImageStar [30, 31], zonotope [4, 27], symbolic interval [36, 41], simulation-based [39], satisfiability modulo theories (SMT) [10, 14, 15], Mixed-Integer Linear Programming (MILP) [2, 20], CROWN [46], FastLin and FastLip [37], and abstraction [23]. Lately, verification of recurrent neural networks (RNNs) has been investigated using unrolling techniques [13] and invariant inference [12, 25, 45].

Approximation of sigmoidal activation functions. In 2010, Pulina at el. [24] verified Sigmoidal FNNs by splitting the sigmoidal function into n intervals. Due to the exponential explosion of abstraction, this approach is not scalable. Grundt et al. [6] converted an interval constraint propagator into the SMT solver iSAT problem. They also encapsulated the sigmoid function into multiple interval boxes with a fixed width for an approximating approach. Fast-Lin [37] and DeepZono [27] overapproximated non-linear activations by applying two parallel linear bounds. DeepPoly [28] and CROWN [46] proposed the back-substitution method to quickly estimate the state vector x without solving LPs. To allow the back-substitution work, they overapproximated functions by two independent linear bounds. Zhang et al. [47] implemented network-wise approximation, NeWise, by solving the optimization problems. They stated if the network is monotonous, the composition of all neuron-wise tightest approximations is as tight as the network-wise approximations. In comparison with VeriNet [8] and RobustVerifier [18], Zhang et al. mentioned in their paper that different methods of tight overapproximation by two linear bounds vary case by case. Henriksen et al. implemented VeriNet [8, 9], which has symbolic interval propagation and a branch and bound-based refinement phase. The authors proposed a unique upper linear bound for the tangent function. DeepCert [38] adopted a fine-grained linear approximation approach that considers the slopes of two linear constraints according to the bounds of the state vector. Lin et al. (RobustVerifier [18]) transformed sigmoid DNNs into equivalent LP problems with relaxation. Ivanov et al. [11] transformed the sigmoidal neural network into an equivalent hybrid system for verification and applied the Taylor model approach for better scalability.

Verification of ReLU DNNs. Unlike sigmoidal operations, ReLU functions have been extensively researched in neural network verification. The star set [32] verifies ReLU networks with exact and overapproximation approaches. It applied three linear constraints known as triangular relaxation to overapproximate ReLU with LP solvers. An extension of the star set, ImageStar [30] can prove the robustness of real-world convolutional neural networks (CNN) such as VGG19. The authors showed the formal method to verify the robustness of semantic segmentation neural networks. Recently, Yang et al. proposed the Facet-Vertex Incidence approach (FVI) [42–44] to verify ReLU DNNs. An FVI matrix has a containment relation of facets and vertices to compute a complete and efficient reachability analysis. Instead of overapproximation ReLU of a single neuron, Singh et al. [26] initiated a parametric framework that can approximate the output of multiple ReLUs jointly.

7 CONCLUSION AND FUTURE WORKS

We propose the star and the relaxed star-based overapproximate reachability analysis for FNNs with Sigmoid or TanH activation functions. Although the star set is the least conservative approach, it could not handle large neural networks as it used LP solver to compute the state bounds. The experiments show that RStar2 and RStar4 are scalable and more precise than the existing polytope approach to verifying large neural networks against adversarial attacks. For a bigger dataset, CIFAR10, the relaxed star approaches verify fairly close to the number of images the star set verified. We show that the symbolic intervals using back-substitution are a promising method to substitute LP optimization for DNNs with bounded nonlinear activation functions such as TanH and Sigmoid. Therefore, our relaxed star methods are very promising for verifying the robustness of deeper neural networks and the star method for smaller neural networks.

Future works. We are interested in applying the proposed methods to verify real-time learning-based cyber-physical systems and extending our work to verify long short-term memory (LSTM) and gated recurrent unit (GRU) neural networks. To increase scalability, we plan to exploit the benefit of GPU computing and parallel processing of CPUs. We plan to explore and improve overapproximation reachability by improving the quality of upper and lower linear constraints of nonlinear activation functions. We are also considering splitting methods to reduce accumulated overapproximation errors.

8 ACKNOWLEDGEMENT

The material presented in this paper is based upon work supported through the National Science Foundation (NSF) grant numbers 2220418, 2331937, and 2245853. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF.

REFERENCES

[1] Nicholas Carlini and David A. Wagner. 2016. Towards Evaluating the Robustness of Neural Networks. *CoRR* abs/1608.04644 (2016). arXiv:1608.04644

[2] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. 2017. Output Range Analysis for Deep Neural Networks. *CoRR* abs/1709.09130 (2017). arXiv:1709.09130

[3] Ruediger Ehlers. 2017. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 269–286.

[4] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572 [stat.ML]

[6] Dominik Grundt, Sorin Liviu Jurj, Willem Hagemann, Paul Kröger, and Martin Fränzle. 2022. Verification of Sigmoidal Artificial Neural Networks using iSAT. *Electronic Proceedings in Theoretical Computer Science* 361 (jul 2022), 45–60. https://doi.org/10.4204/eptcs.361.6

[7] Gurobi Optimization, LLC. 2020. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[8] P. Henriksen and A. Lomuscio. 2020. Efficient neural network verification via adaptive refinement and adversarial search. *In Proceedings of the 24th European Conference on Artificial Intelligence (ECAI20)* (2020).

[9] P. Henriksen and A. Lomuscio. 2021. An efficient splitting method for neural network verif ication via indirect effect analysis. *In Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI21)* (2021).

[10] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2016. Safety Verification of Deep Neural Networks. *CoRR* abs/1610.06940 (2016). arXiv:1610.06940

[11] Radoslav Ivanov, Taylor J. Carpenter, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. 2020. Verifying the Safety of Autonomous Systems with Neural Network Controllers. *ACM Trans. Embed. Comput. Syst.* 20, 1, Article 7 (dec 2020), 26 pages. https://doi.org/10.1145/3419742

[12] Yuval Jacoby, Clark Barrett, and Guy Katz. 2020. Verifying recurrent neural networks using invariant inference. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 57–74.

[13] Yuval Jacoby, Clark W. Barrett, and Guy Katz. 2020. Verifying Recurrent Neural Networks using Invariant Inference. *CoRR* abs/2004.02462 (2020). arXiv:2004.02462

[14] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. *CoRR* abs/1702.01135 (2017). arXiv:1702.01135

[15] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. 2019. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In *Computer Aided Verification*, Isil Dillig and Serdar Tasiran (Eds.). Springer International Publishing, Cham, 443–452.

[16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. CIFAR-10 (Canadian Institute for Advanced Research). (2014).

[17] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. (2010).

[18] Wang Lin, Zhengfeng Yang, Xin Chen, Qingye Zhao, Xiangkun Li, Zhiming Liu, and Jifeng He. 2019. Robustness Verification of Classification Deep Neural Networks via Linear Programming. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11410–11419. https://doi.org/10.1109/CVPR.2019.01168

[19] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2017. Towards Robust Neural Networks via Random Self-ensemble. *CoRR* abs/1712.00673 (2017). arXiv:1712.00673

[20] Alessio Lomuscio and Lalit Maganti. 2017. An approach to reachability analysis for feed-forward ReLU neural networks. *CoRR* abs/1706.07351 (2017). arXiv:1706.07351

[21] Mark Niklas Müller, Gagandeep Singh, Mislav Balunovic, Gleb Makarchuk, Anian Ruos, François Serre, Maximilian Baader, Dana Drachsler Cohen, Timon Gehr, Adrian Hoffmann, Jonathan Maurer, Matthew Mirman, Markus Püschel, and Martin Vechev. 2021. *ERAN: ETH Robustness Analyzer for Neural Networks*. https://github.com/eth-sri/eran

[22] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2016. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples. *CoRR* abs/1602.02697 (2016). arXiv:1602.02697

[23] Pavithra Prabhakar and Zahra Rahimi Afzal. 2019. Abstraction based output range analysis for neural networks. *Advances in Neural Information Processing Systems* 32 (2019).

[24] Luca Pulina and Armando Tacchella. 2010. An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In *Computer Aided Verification*, Tayssir Touili, Byron Cook, and Paul Jackson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 243–257.

[25] Wonryong Ryou, Jiayu Chen, Mislav Balunovic, Gagandeep Singh, Andrei Dan, and Martin Vechev. 2021. Scalable Polyhedral Verification of Recurrent Neural Networks. arXiv:2005.13300 [cs.LG]

[26] Gagandeep Singh, Rupanshu Ganvir, Markus Püschel, and Martin Vechev. 2019. Beyond the Single Neuron Convex Barrier for Neural Network Certification. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/0a9fdbb17feb6ccb7ec405cfb85222c4-Paper.pdf

[27] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. 2018. Fast and Effective Robustness Certification. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc., 10802–10813.

[28] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An Abstract Domain for Certifying Neural Networks. *Proc. ACM Program. Lang.* 3, POPL, Article 41 (Jan. 2019), 30 pages.

[29] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. arXiv:1312.6199 [cs.CV]

[30] Hoang-Dung Tran, Stanley Bak, Weiming Xiang, and Taylor T. Johnson. 2020. Verification of Deep Convolutional Neural Networks Using ImageStars. *CoRR* abs/2004.05511 (2020). arXiv:2004.05511 https://arxiv.org/abs/2004.05511

[31] Hoang-Dung Tran, Neelanjana Pal, Patrick Musau, Diego Manzanas Lopez, Nathaniel Hamilton, Xiaodong Yang, Stanley Bak, and Taylor Johnson. 2021. Robustness Verification of Semantic Segmentation Neural Networks using Relaxed Reachability.

[32] Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. 2019. Star-Based Reachability Analysis of Deep Neural Networks. In *Formal Methods – The Next 30 Years*, Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira (Eds.). Springer International Publishing, Cham.

[33] Hoang-Dung Tran, Patrick Musau, Diego Manzanas Lopez, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T Johnson. 2019. Parallelizable Reachability Analysis Algorithms for Feed-Forward Neural Networks. In *2019 IEEE/ACM 7th International Conference on Formal Methods in Software Engineering (FormaliSE)*. 51–60.

[34] Hoang-Dung Tran, Neelanjana Pal, Diego Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor Johnson. 2021. Verification of piecewise deep neural networks: a star set approach with zonotope pre-filter. *Formal Aspects of Computing* (2021).

[35] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. 2020. NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. In *Computer Aided Verification*, Shuvendu K. Lahiri and Chao Wang (Eds.). Springer International Publishing, Cham, 3–17.

[36] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Formal Security Analysis of Neural Networks using Symbolic Intervals. *CoRR* abs/1804.10829 (2018). arXiv:1804.10829

[37] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S. Dhillon, and Luca Daniel. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. arXiv:1804.09699 [stat.ML]

[38] Yiting Wu and Min Zhang. 2021. Tightening Robustness Verification of Convolutional Neural Networks with Fine-Grained Linear Approximation. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 13 (May 2021), 11674–11681. https://doi.org/10.1609/aaai.v35i13.17388

[39] Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. 2017. Output Reachable Set Estimation and Verification for Multi-Layer Neural Networks. *CoRR* abs/1708.03322 (2017). arXiv:1708.03322

[40] Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. 2017. Reachable Set Computation and Safety Verification for Neural Networks with ReLU Activations. *CoRR* abs/1712.08163 (2017). arXiv:1712.08163

[41] Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. 2018. Specification-Guided Safety Verification for Feedforward Neural Networks. *CoRR* abs/1812.06161 (2018). arXiv:1812.06161

[42] Xiaodong Yang, Taylor T Johnson, Hoang-Dung Tran, Tomoya Yamaguchi, Bardh Hoxha, and Danil Prokhorov. 2021. Reachability Analysis of Deep ReLU Neural Networks Using Facet-Vertex Incidence. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control* (Nashville, Tennessee) *(HSCC '21)*. Association for Computing Machinery, New York, NY, USA, Article 18, 7 pages. https://doi.org/10.1145/3447928.3456650

[43] Xiaodong Yang, Hoang-Dung Tran, Weiming Xiang, and Taylor T. Johnson. 2020. Reachability Analysis for Feed-Forward Neural Networks using Face Lattices. *CoRR* abs/2003.01226 (2020). arXiv:2003.01226

[44] Xiaodong Yang, Tomoya Yamaguchi, Hoang-Dung Tran, Bardh Hoxha, Taylor T. Johnson, and Danil V. Prokhorov. 2021. Reachability Analysis of Convolutional Neural Networks. *CoRR* abs/2106.12074 (2021). arXiv:2106.12074 https://arxiv.org/abs/2106.12074

[45] Hongce Zhang, Maxwell Shinn, Aarti Gupta, Arie Gurfinkel, Nham Le, and Nina Narodytska. 2020. Verification of recurrent neural networks for cognitive tasks via reachability analysis. In *ECAI 2020*. IOS Press, 1690–1697.

[46] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. 2018. Efficient Neural Network Robustness Certification with General Activation Functions. arXiv:1811.00866 [cs.LG]

[47] Zhaodi Zhang, Yiting Wu, Si Liu, Jing Liu, and Min Zhang. 2022. Provably Tightest Linear Approximation for Robustness Verification of Sigmoid-like Neural Networks. https://doi.org/10.48550/ARXIV.2208.09872