2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Star based Reachability Analysis of Interval Neural Networks

Vishnu Bondalakunta[1] and Pavithra Prabhakar[2]

Abstract— **The paper explores the computation of output reachable sets for Interval Neural Networks (INNs) with ReLU activation functions. An INN is a generalization of a Neural Network (NN), where the weights and biases are intervals rather than numbers. We propose a novel algorithm for computing precise over-approximations of the output reachable sets for INNs by introducing a novel data structure called *interval star set* which is a generalized version of the star set. Specifically, when the INN is a traditional NN, our method reduces to the standard star-based verification of NNs. We present experimental results that demonstrate that our method outperforms the existing method based on mixed integer linear programming (MILP) for the problem of INN output reachable set computation.**

2023 62nd IEEE Conference on Decision and Control (CDC) | 979-8-3503-0124-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/CDC49753.2023.10383228

I. Introduction

Neural Networks (NN) are being extensively used in safety critical control systems such as airborne collision avoidance systems [1] and autonomous vehicles [2] to perform sophisticated tasks. For these systems, strong assurances of correctness of verification are desired and this has attracted a large body of work focusing on formal verification of neural networks [3], [4], [5], [6], [7]. Output range computation problem is a fundamental problem in the safety analysis of neural network controlled autonomous systems. However, the problem is known to be NP-hard [5], and becomes computationally expensive for large number of neurons.

Abstraction based analysis is a promising approach to deal with the scalability issue. More recently, such an approach has been employed for neural networks [8], where a neural network is abstracted into a small network albeit with interval weights and biases, called an *Interval Neural Network* (INN). The abstraction is sound in that the output range of the INN is an over-approximation of the original network and the safety of the abstract INN implies the safety of the original NN. The paper also provides an MILP based method for computation of the range of an INN.

The broad objective of this paper is to explore efficient algorithms for the output range analysis of INNs which are promising data structures for the abstraction based analysis of neural networks. We note that existing algorithms for neural network verification cannot be directly applied in the case of INNs. We, however, take inspiration from existing NN verification approaches, specifically, star based approach [7], to analyze the output range of INNs in this paper. Star set is an abstract domain [9] that has been successfully applied toward scalable verification of neural networks. Here, we propose a generalization of the star set, namely, *interval star set* as an abstract domain for reachable set computation of an interval neural network. The crux of reachable set computation of neural networks relies on being able to efficiently compute two types of functions on an abstract domain, namely, affine transformations and ReLU application. These two operations can be done efficiently using the star set data structure as opposed to, say, polyhedral set representations. To compute the reachable set of an interval neural network, we need to compute the output of a set with respect to an infinite set of affine transformations that are presented by an interval matrix and an interval vector. Our first result provides an elegant representation for an interval star set that abstracts the set of states obtained by the "interval" affine transformation of an input interval star set. The next step is the application of the ReLU function on an interval star set. This can be interpreted as the application of a series of intersections with half-planes followed by linear transformation. We show that intersections with half planes for interval star set can be performed exactly and efficiently.

We have implemented this interval star set based reachable set computation algorithm for interval neural networks. Our experimental evaluation highlights the benefits of this approach with respect to a previously known MILP based analysis [10]. Specifically, our algorithm is faster and is able to analyze large interval neural networks, whereas the MILP solver runs into technical issue due to large constants in the encoding.

II. Related Work

We discuss the existing body of literature concerned with the verification of neural networks. There is a large body of work on heuristic and dynamic analysis techniques to test robustness of neural networks [3], [11], [12], [13]. They are effective in finding adversarial examples as opposed to formally proving safety of a network or range computation of a network.

Earlier neural network verification makes use of constraint solving where a verification problem is reduced to solving constraints, existing SAT/SMT based verification techniques include [14], [15], [5], [16]. These techniques are often sound and complete, however they are limited in scalability.

To improve the scalability, several verification techniques have been proposed which utilize the idea of abstraction in the form of abstract interpretation [17]. The main idea is to consider well designed numerical abstract domains such as boxes [17], zonotopes [18] and polyhedra [19].

*This work was partially supported by NSF CAREER Grant No. 1552668, NSF Grant No. 2008957 and Amazon Research Award.

[1] Vishnu Bondalakunta is a PhD student of Department of Computer Science, Kansas State University, Manhattan, KS 66506, USA vishnub@ksu.edu

[2] Pavithra Prabhakar is a Profesor of Computer Science at the Department of Computer Science, Kansas State University, Manhattan, KS 66506, USA pprabhakar@ksu.edu

Some examples are *DeepZ* [4], *DeepPoly* [20], *β-Crown* [6], *NNV* [7] and *nnenum* [21]. These techniques have produced promising results in neural network verification.

Our work is mainly concerned about Interval Neural Networks (INNs) which are a new abstract system that models the behavior of neural networks by representing the weights of a network as intervals [8]. An MILP based verification technique for analyzing these abstract networks is also provided in the original paper [8]. However, it is not efficient to verify neural networks or compute its output range using MILP-based techniques [22] and the same applies to INNs for similar reasons. It is important to note that existing verification engines for neural networks cannot be applied to work on INNs.

Our work generalizes and extends the state-of-the-art star based verification ideas [7], [21] for the scalable verification analysis and output range computation of the INNs.

III. Preliminaries

The set of all real numbers is represented by $\mathbb{R}$. The set of all non-negative real numbers is represented by $\mathbb{R}^+$. Given a non-negative integer k, let $[k]$ denote the set $\{1,2,\cdots,k\}$. A partition P over a set S is the set $P=\{P_1,\cdots,P_k\}$ such that $\bigcup_{i=1}^{k} P_i = S$ and $P_i \cap P_j = \emptyset, \quad \forall i,j \in [k]$ and $i \neq j$. The cardinality of a set S is represented as $|S|$. An ordered set S is a set whose elements are arranged in an order and where $S[i]$ represents the i-th element of S. The complement of a set S is represented as $\neg S$. We will use 'σ' to represent the ReLU activation function, where $\sigma(x)=x$ if $x \geq 0$ and $\sigma(x)=0$ if $x \leq 0$.

A. Intervals and operations

We review some standard definitions and results from interval analysis [23]. A closed interval is denoted by $[a,b]$ which represents the set $\{x \in \mathbb{R} : a \leq x \leq b\}$. The set of all closed (real-valued) intervals is denoted by $\mathbb{IR}$. We denote the left and right endpoints of an interval X by $\underline{X}$ and $\overline{X}$ respectively, i.e $X=[\underline{X},\overline{X}]$. An interval X is called a point interval if $\underline{X}=\overline{X}$. The sum of two intervals $X,Y \in \mathbb{IR}$ is given by $X+Y=\{x+y : x \in X, y \in Y\}$. In terms of endpoints, the sum $X+Y$ of two intervals is given by:

$$X+Y=[\underline{X}+\underline{Y},\overline{X}+\overline{Y}]$$

The product of two intervals $X,Y \in \mathbb{IR}$ is given by $X.Y=\{xy : x \in X, y \in Y\}$.
In terms of endpoints, the products $X.Y$ of two intervals is given by:

$$X.Y=[\min(S),\max(S)] \text{ where,}$$
$$S=\{\underline{XY},\underline{X}\overline{Y},\overline{X}\underline{Y},\overline{XY}\}$$

B. Interval Matrices and Vectors

A matrix M of dimension (m,n) over a set S is a function from $[m]\times[n]$ to S. We denote the (i,j)-th element of M, namely $M(i,j)$, as M_{ij} and the set of all matrices of dimension (m,n) as $S^{m\times n}$. When $n=1$, we write $S^{m\times 1}$ as simply S^m which represents the set of all vectors of dimension m over the set S. The j-th element of a vector V will be written as V_j instead of V_{j1}.

A real valued matrix is a matrix over real numbers. An *interval matrix* is a matrix whose elements are real intervals. Alternatively we represent an interval matrix $\mathscr{M}$ of dimensions (m,n) as a tuple of two (m,n) dimensional real-valued matrices $(\underline{\mathscr{M}},\overline{\mathscr{M}})$ where $\mathscr{M}_{ij}=[\underline{\mathscr{M}}_{ij},\overline{\mathscr{M}}_{ij}]$. $\underline{\mathscr{M}}$ and $\overline{\mathscr{M}}$ are referred to as the 'lower' and 'upper' matrices(limits) of $\mathscr{M}$ respectively. Given two real matrices M' and M'' having the same dimensions (m,n), we write $M' \leq M''$ if $\forall(i,j) \in [m]\times[n], \quad M'_{ij} \leq M''_{ij}$.

We also say that a real-valued matrix M is an *instance* of an interval matrix $\mathscr{M}$ of dimensions (m,n) if both have the same dimensions and $\forall(i,j) \in [m]\times[n]$, $\underline{\mathscr{M}}_{ij} \leq M_{ij} \leq \overline{\mathscr{M}}_{ij}$. We write $M \in \mathscr{M}$ to mean that M is an *instance* of $\mathscr{M}$.

Throughout this paper, bold font symbols will represent interval matrices (vectors) and their lower and upper limits and normal font symbols will represent real valued matrices (vectors) unless explicitly mentioned otherwise. For example, $\mathbf{M} \in \mathbb{IR}^{m\times n}$ is an interval matrix and $\overline{\mathbf{M}}, \underline{\mathbf{M}} \in \mathbb{R}^{m\times n}$ are real matrices. $M' \in \mathbf{M}$ is also a real matrix belonging to $\mathbb{R}^{m\times n}$.

C. Interval Constraints

We refer to a constraint of the form $\mathbf{W}*x+\mathbf{b} \leq 0$, where $\mathbf{W} \in \mathbb{IR}^{p\times n}$, $\mathbf{b} \in \mathbb{IR}^{p\times 1}$ and $x \in \mathbb{R}^{n\times 1}$ as an *interval constraint* over variables x. A valuation $v \in \mathbb{R}^{n\times 1}$ is said to satisfy the above interval constraint if $\exists W \in \mathbf{W}$ and $\exists b \in \mathbf{b}$ such that $W*v+b \leq 0$. When $\mathbf{W},\mathbf{b}$ are point intervals, the interval constraints are just linear constraints.

Reasoning about such quantified constraints can be hard in general as opposed to linear constraints. We observe that when the variables belong to the domain of non-negative real numbers, the problem of finding the set of values satisfying an interval constraint reduces to finding the set of values satisfying a linear constraint corresponding to the lower or upper matrices of the interval matrices. This is captured by the following lemma.

Lemma 1: The set of satisfying valuations of an interval constraint $\mathbf{W}*x+\mathbf{b} \leq 0$ and that of the linear constraint $\underline{\mathbf{W}}*x+\underline{\mathbf{b}} \leq 0$ over $x \in \mathbb{R}^{+n}$ are equal. The set of satisfying valuations of an interval constraint $\mathbf{W}*x+\mathbf{b} \geq 0$ and that of the linear constraint $\overline{\mathbf{W}}*x+\overline{\mathbf{b}} \geq 0$ over $x \in \mathbb{R}^{+n}$ are equal.

Proof: We prove the above by showing that if a valuation v satisfies the the linear constraint $\underline{\mathbf{W}}*x+\underline{\mathbf{b}} \leq 0$ then it also satisfies the interval constraint $\mathbf{W}*x+\mathbf{b} \leq 0$ and vice-versa.

For a valuation v satisfying $\underline{\mathbf{W}}*x+\underline{\mathbf{b}} \leq 0$, it follows from the definition of an interval constraint that it satisfies $\mathbf{W}*x+\mathbf{b} \leq 0$.

For a valuation v satisfying $\mathbf{W}*x+\mathbf{b} \leq 0$, there exists some W,b such that $W*v+b \leq 0$. Observe that $\underline{\mathbf{W}} \leq W$ and $\underline{\mathbf{b}} \leq b$ and v is non-negative. This implies that $\underline{\mathbf{W}}*v+\underline{\mathbf{b}} \leq W*v+b \leq 0$.

This proves that $v \vdash \mathbf{W}*x+\mathbf{b} \leq 0 \iff v \vdash \underline{\mathbf{W}}*x+\underline{\mathbf{b}} \leq 0$.

The proof of the statement $v \vdash \mathbf{W}*x+\mathbf{b} \geq 0 \iff v \vdash \overline{\mathbf{W}}*x+\overline{\mathbf{b}} \geq 0$ follows in a similar manner.

■

D. Interval Affine Transformation

Given a set of linear transformations which are instances of the interval matrix $\mathbf{W}$ i.e $\{W \mid W \in \mathbf{W}\}$, a set of vectors which are instances of the interval vector $\mathbf{b}$ and a set of real vectors X, we refer to $\langle \mathbf{W}, \mathbf{b} \rangle(X)$ as the *interval affine transformation* of X with respect to $\mathbf{W}$ and $\mathbf{b}$. The set $\langle \mathbf{W}, \mathbf{b} \rangle(X)$ represents the union of all the sets which are obtained by the linear transformation of X by some $W \in \mathbf{W}$ followed by a translation along some vector $b \in \mathbf{b}$.

Definition 1: Given a set $S \subseteq \mathbb{R}^n$, an interval matrix $\mathbf{W} \in \mathbb{IR}^{m \times n}$ and an interval vector $\mathbf{b} \in \mathbb{IR}^n$, the *interval affine transformation* of X w.r.t $\mathbf{W}$ and $\mathbf{b}$ is:
$\langle \mathbf{W}, \mathbf{b} \rangle(X) = \{W * x + b \mid W \in \mathbf{W}, b \in \mathbf{b}, x \in X\}$

E. Interval Neural Network

An interval neural network (*INN*) [8] is a new computational model which is a generalized version of the standard neural network (*NN*). Recall that, a neural network is a computational model that consists of an input layer, an output layer and multiple hidden layers. Each layer consists of nodes also called neurons. Each layer of neurons is connected to the preceding layer of neurons by edges that are ascribed with some weight values. Given some input values to the input layer neurons, the values of the neurons at the next layer are obtained in two steps. First, a weighted sum is computed using the weighted edges followed by an addition of a bias associated with the output node. We refer to the above computation as the affine transformation across a layer. Then, this is followed by an application of an activation operation. We will only consider the ReLU activation function. An INN is structurally identical to a neural network with the only difference being that the ascribed weights and biases assume a range of values from an interval instead of a single number. Consequently, the output of an INN is a set of values. An INN can also be viewed as a collection of an uncountable number of neural networks (having the same structure), where the weights and biases lie in the corresponding weight and bias intervals of the given INN. Therefore, the output range of the INN can be understood as the union of the output ranges of all these neural networks it represents.

Definition 2 (Interval Neural Network): [8]
An interval neural network (INN) is a tuple, $(k, \{S^{(i)}\}_{i \in [k] \cup \{0\}}, \{\mathbf{W}^{(i)}\}_{i \in [k]}, \{\mathbf{b}^{(i)}\}_{i \in [k]})$ where,

- $k + 1 \in \mathbb{N}$ refers to the number of layers
- $\forall i \in [k] \cup \{0\}$, $S^{(i)}$ is an ordered set of nodes of i-th layer in the interval neural network. $S^{(0)}$ is the input layer, $S^{(k)}$ is the output layer and $S^{(i)}, \forall i \in [k]/\{k\}$ is a hidden layer.
- $\forall i \in [k]$, $\mathbf{W}^{(i)} \in \mathbb{IR}^{m \times n}$, where $m = |S^{(i)}|$ and $n = |S^{(i-1)}|$ represents the weights of the edges between the $i-1$-th and i-th layer. The weight of the edge connecting $S^{(i)}[p]$ and $S^{(i-1)}[q]$ is given by $\mathbf{W}^{(i)}_{pq}$.
- $\forall i \in [k]$, $\mathbf{b}^{(i)} \in \mathbb{IR}^{m \times 1}$, where $m = |S^{(i)}|$ represents the biases associated with the nodes in the i-th layer. The bias of the p-th element of $S^{(i)}$ i.e $S^{(i)}[p]$ is given by $\mathbf{b}^{(i)}_p$.

A neural network (NN) is a special case of an INN when all intervals in $\mathbf{W}^{(i)}$ and $\mathbf{b}^{(i)}$ are point intervals.

Now, we define the reachable set of an INN. First we explain what a valuation of a layer of an INN is. Formally, a valuation of a layer $S^{(i)}$ of an INN is a vector $v \in \mathbb{R}^{|S^{(i)}|}$. The interpretation being that the jth node of the ordered set $S^{(i)}$ is assigned the value v_j. Given a set of valuations of input layer $\mathscr{I}$ of an INN, we compute the set obtained by propagating the input valuations across the first hidden layer. This is done by computing a weighted sum dictated by the interval weights of the layer and the addition of the interval bias associated with the output node followed by a transformation under the ReLU function. Alternatively, this set is simply the ReLU transformation of the interval affine transformation of $\mathscr{I}$ with respect to $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$. This set is referred to as the reachable set across the first layer. This process is continued iteratively across the subsequent layers. The reachable set of the INN is defined to be the reachable set across the final layer.

Definition 3: [Reachable set of *INN*] Given a set of input valuations $\mathscr{I}$, an interval neural network (INN) $\mathscr{T} = (k, \{S^{(i)}\}_{i \in [k] \cup \{0\}}, \{\mathbf{W}^{(i)}\}_{i \in [k]}, \{\mathbf{b}^{(i)}\}_{i \in [k]})$, the reachable set $\mathscr{R}_i$ across the i-th layer is defined inductively as follows:

$$\mathscr{R}_0 = \mathscr{I},$$
$$\forall i \in [k], \quad \mathscr{R}_i := \sigma(\langle \mathbf{W^{(i)}}, \mathbf{b^{(i)}} \rangle(\mathscr{R}_{i-1}))$$

The reachable set $\mathscr{R}(\mathscr{T}, \mathscr{I})$ is given by $\mathscr{R}_k$. In this work, we restrict the input set $\mathscr{R}_0$ to be a bounded polyhedron set.

We now explore the notion of safety of an INN. An INN is 'safe' for an input $\mathscr{I}$ if all the elements of the reachable set belong to a specified 'safe' set. Specifications are encoded as a set of linear constraints over the set of valuations of the output layer of the INN. An INN is safe for a given input set with respect to a safety specification if the reachable set obtained by propagating the input valuations satisfies the safety specification.

Problem 1 (Safety Verification of INN): Given an INN $\mathscr{T}$, an initial set $\mathscr{I}$ and a safety specification $\mathscr{X}$ (a set of safe output layer valuations), we say that $\mathscr{T}$ is safe with respect to $\mathscr{I}$ and $\mathscr{X}$, written $(\mathscr{T}, \mathscr{I}) \models \mathscr{X}$, if $\mathscr{R}(\mathscr{T}, \mathscr{I}) \subseteq \mathscr{X}$. Otherwise the INN is said to be unsafe.

IV. Interval Star

A popular method for obtaining the reachable set of a neural network involves representing the set of input valuations by a data structure and then propagating this structure across the layers of the network. The *Star Set* representation [9] has emerged as a promising representation for this problem owing to its cheap and fast affine transformation and inexpensive half-space intersections. However, in the case of INN, obtaining affine transformations of a star set is neither straightforward nor efficient. We introduce a novel set representation *Interval Star Set* which is a generalization of the star set for obtaining the reachable set of an INN.

An Interval Star Θ has three components: a center interval vector $\mathbf{c}$, a set of basis interval vectors $\mathbf{V}$ and a predicate P. The set of points represented by the Interval Star Θ is given by the linear combination of the basis interval vectors in $\mathbf{V}$ with scalars 'α' satisfying the predicate P followed by a translation along the center interval vector $\mathbf{c}$.

Definition 4 (Interval Star): An *Interval Star* Θ is a tuple $\langle \mathbf{c}, \mathbf{V}, P \rangle$ where $\mathbf{c} \in \mathbb{IR}^{n \times 1}$, $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m\}$ is a set of m interval vectors in $\mathbb{IR}^n$ which is represented as an interval matrix having $\mathbf{v}_i$ for $i \in [m]$ as column vectors and $P : \mathbb{R}^{+m} \rightarrow \{\top, \bot\}$ is a predicate.

Definition 5: [Set represented by an Interval Star] The semantics of the set represented by the interval star $\Theta = \langle \mathbf{c}, \mathbf{V}, P \rangle$ is given as:

$$[\![\Theta]\!] = \{\ c + \sum_{i=1}^{m} (\alpha_i v_i) \mid c \in \mathbf{c}, V \in \mathbf{V} \, \text{and} \, P(\alpha_1, \cdots, \alpha_m) = \top \}$$

We will sometimes refer to the tuple Θ and set of states $[\![\Theta]\!]$ as Θ interchangeably. For this work, we only consider the predicate P to be a conjunction of linear constraints, $P(\alpha) := C\alpha + d \leq 0$ where for p linear constraints, $C \in \mathbb{R}^{p \times n}$, α is the vector of n non-negative variables, i.e $\alpha \in \mathbb{R}^{+n}$ and $d \in \mathbb{R}^{p \times 1}$. An interval star is empty if and only if $P(\alpha)$ is empty.

We note that an interval star set and star set are identical representations when all intervals in $\mathbf{c}$ and $\mathbf{V}$ of the interval star are point intervals.

Given a set of valuations represented as an interval star Θ at a layer of an INN, in order to compute the reach set across this layer, we first need to compute the interval affine transformation of $[\![\Theta]\!]$ with respect to the weight and bias of the layer. Obtaining the interval star representation of this set is challenging and may contain a large number of basis vectors and a large number of constraints in the predicate. Instead, we compute an interval star which is an over-approximation of the exact interval affine transformed set. This over-approximate interval star has the advantage of being easy to obtain in a fast manner as it only involves interval matrix arithmetic.

Definition 6: [Over-Approximate Interval Affine Transformation of an Interval Star]. Given an *Interval Star* $\Theta = \langle \mathbf{c}, \mathbf{V}, P \rangle$, we give an over-approximate interval affine transformation of Θ with respect to $\mathbf{M}$ and $\mathbf{b}$ represented as $\langle \mathbf{W}, \mathbf{b} \rangle^*(\Theta)$ where the semantics of the set it represents is given by:

$$[\![\langle \mathbf{W}, \mathbf{b} \rangle^*(\Theta)]\!] = \langle \mathbf{Mc} + \mathbf{b}, \mathbf{MV}, P \rangle$$

A. Properties of Interval Stars

We discuss some properties of interval stars in the following propositions. Some of the proofs are included in the appendix section.

Proposition 1: Any bounded convex polyhedron set can be represented as an *Interval Star*

Proof: Consider a bounded convex polyhedron set $\mathscr{P} := \{x \mid Cx + d \leq 0, x \in \mathbb{R}^n\}$. Let $\mathscr{B} = \mathscr{I}_1 \times \mathscr{I}_2 \times \cdots \times \mathscr{I}_n$ be any box containing $\mathscr{P}$ where $\mathscr{I}_i \in \mathbb{IR}$ for $i \in [n]$ and let b be the vertex of $\mathscr{B}$ such that $b = [\underline{\mathscr{I}}_1, \underline{\mathscr{I}}_2, \cdots, \underline{\mathscr{I}}_n]^T$.

The interval star Θ with the center $\mathbf{c}$ such that $\underline{\mathbf{c}} = \overline{\mathbf{c}} = b$, the basis vectors $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\}$ such that $\underline{\mathbf{v}}_i = \overline{\mathbf{v}}_i = e_i$ where e_i is the ith canonical basis vector of $\mathbb{R}^n$ and predicate $P(\alpha) := C\alpha + Cb + d \leq 0$ is equivalent to the polyhedron $\mathscr{P}$. The above is true as $x \in \Theta$ can be written as $x = b + \alpha$. If this $x \in \mathscr{P}$ then this implies that $C(b + \alpha) + d \leq 0$ which is exactly the predicate P of Θ. Also given a point $x \in \mathscr{P}$, choosing α to be $x - b$ will result in x belonging to Θ as well.

We also need to check that $\alpha \in \mathbb{R}^{+n}$ to be consistent with the definition of an interval star. We observe that for $x \in \Theta$, $x = b + \alpha$. Since Θ and $\mathscr{P}$ represent the same set, $x \in \mathscr{P}$ which implies that $\alpha = x - b \geq 0$ as b is the lowest vertex of the box containing $\mathscr{P}$. ■

Given an input set represented by an interval star set, to obtain the reach set of an INN we need to compute a series of interval affine transformations and ReLU transformations. Instead of computing the exact interval affine transformation of an interval star, we compute the interval star as described in Definition 6 and show that it is an over-approximation of the exact interval affine transformation of an interval star. For performing the ReLU transformation of an interval star set, we need to be able to compute the intersection of an interval star set with a half-space which we explain below in Proposition 3.

Proposition 2: Given an interval star Θ, an interval matrix $\mathbf{M}$ and an interval offset vector $\mathbf{b}$,

$$\langle \mathbf{M}, \mathbf{b} \rangle([\![\Theta]\!]) \subseteq [\![\langle \mathbf{M}, \mathbf{b} \rangle^*(\Theta)]\!]$$

Proof: The above proposition follows as a consequence of Definitions 1, 5, 6 and the fact that $A' * B' \in \mathbf{A} * \mathbf{B}$ where $A' \in \mathbf{A}, B' \in \mathbf{B}$. In general, we have that,

$$\{A' * B' \mid A' \in \mathbf{A}, B' \in \mathbf{B}\} \subseteq \{X \in \mathbf{A} * \mathbf{B}\}$$

Refer to section 7.2 of [23] ■

Proposition 3: [Interval Star and Half Space Intersection] The intersection of a set represented by an *Interval Star* $\Theta := \langle \mathbf{c}, \mathbf{V}, P \rangle$ and a half space $\mathscr{H} := \{x \mid H * x + g \leq 0\}$ is given by the set represented by another star $\hat{\Theta}$ with the following characteristics:

$$\hat{\Theta} = \langle \hat{\mathbf{c}}, \hat{\mathbf{V}}, \hat{P} \rangle, \, \hat{\mathbf{c}} = \mathbf{c}, \, \hat{\mathbf{V}} = \mathbf{V}, \, \hat{P} = P \wedge P',$$

$$P'(\alpha) := \underline{(H \times \mathbf{V})}\alpha + \underline{(H \times \mathbf{c})} + g \leq 0$$

.

Proof: For any $x \in [\![\Theta]\!] \cap \mathscr{H}$, we have,

$$x \in \mathbf{y} = \mathbf{c} + \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \bigwedge (Hx + g \leq 0)$$

Or equivalently,

$$\begin{aligned} & x \in \mathbf{y} = \mathbf{c} + \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \bigwedge \\ & \quad \exists c \in \mathbf{c} \; \exists V \in \mathbf{V} : (H \times V) \times \alpha + (H \times c) + g \leq 0 \end{aligned}$$

Which can be represented as,

$$x \in \mathbf{y} = \mathbf{c} + \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \bigwedge (H \times \mathbf{V}) \times \alpha + (H \times \mathbf{c}) + g \leq 0$$

Using Lemma 1, we can reduce the quantified linear constraints to linear constraints and get,

$$x \in \mathbf{y} = \mathbf{c} + \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \bigwedge (\underline{H \times \mathbf{V}}) \times \alpha + (\underline{H \times \mathbf{c}}) + g \leq 0$$

Which implies that the intersection is another *Interval Star* with the same center $\mathbf{c}$ and basis vectors $\mathbf{v}_i$ as Θ and an updated predicate $\hat{P} = P \wedge P'$, $P'(\alpha) := \underline{(H \times \mathbf{V})}\alpha + \underline{(H \times \mathbf{c})} + g \leq 0$ ■

V. Star-Based Reachability Analysis of INNs

Given an INN, and an input bounded polyhedron set $\mathscr{I}$, our main goal is to compute an over-approximation of the reachable set of $\mathscr{I}$ across the INN as per Definition 3. We first begin by converting a given bounded convex polyhedron input set $\mathscr{I}$ into an interval star using Proposition 1. The reach set is computed layer-by-layer, wherein each layer we first compute the over-approximate interval affine transformation(with respect to the layer's weights and biases) of the reach set across the preceding layer and follow it by a ReLU transformation. The over-approximate interval affine transformation of the interval star $\Theta = \langle \mathbf{c}, \mathbf{V}, P \rangle$ with respect to the weight $\mathbf{W}$ and bias $\mathbf{b}$ is quickly computed as another interval star $\Theta' = \langle \mathbf{W}\mathbf{c} + \mathbf{b}, \mathbf{W}\mathbf{V}, P \rangle$ as per Definition 6.

The crux of the problem lies in computing the exact image of an interval star Θ across the ReLU layer. As in the existing star based verification method [7], we consider the ReLU function as a series of σ_i functions applied one after the other, i.e for a layer L of n neurons, the reachable set is

$$\mathscr{R}_L = \sigma_n(\sigma_{n-1}(\cdots \sigma_1(\Theta)) \cdots)$$

The $\sigma_i(.)$ function applies the ReLU activation on the input for only that particular neuron i. To do so, first the input star $\Theta = \langle \mathbf{c}, \mathbf{V}, P \rangle$ is split into two stars $\Theta_1 = \langle \mathbf{c}, \mathbf{V}, P_1 \rangle$ and $\Theta_2 = \langle \mathbf{c}, \mathbf{V}, P_2 \rangle$ according to Proposition 3 where $\Theta_1 = \Theta \wedge x_i \geq 0$ and $\Theta_2 = \Theta \wedge x_i \leq 0$. Then we check whether Θ_1 and Θ_2 are empty using linear programming(LP) feasibility checking. If Θ_2 is non-empty, since all $x \in \Theta_2$ have $x_i \leq 0$, the activation of ReLU on the neuron i must map all x_i to 0. This is done by multiplying the center and basis vectors of Θ_2 with the mapping matrix $M = [e_1, \cdots, e_{i-1}, 0, e_{i+1}, \cdots, e_n]$. That is,

$$\Theta_2 \leftarrow M * \Theta = \langle M\mathbf{c}, M\mathbf{V}, P_2 \rangle$$

If Θ_1 is non-empty, since $x_i \geq 0$, the ReLU activation on neuron i is the identity transformation that is,

$$\Theta_1 \leftarrow \Theta = \langle \mathbf{c}, \mathbf{V}, P_1 \rangle$$

Therefore, $\sigma_i(\Theta) = \langle \mathbf{c}, \mathbf{V}, P_1 \rangle \cup \langle M\mathbf{c}, M\mathbf{V}, P_2 \rangle$.

Our reachability algorithm works as follows. A given set of inputs to an INN $\mathscr{T}$ is represented by an interval star Θ. We propagate interval stars across the INN layer-by-layer. We first compute the over-approximate interval affine transformation of the input star Θ with respect to the starting layer's weights $\mathbf{W}^{(1)}$and biases $\mathbf{b}^{(1)}$. The ReLU image of of the resultant star is obtained by computing a series of σ_i images. Computing a σ_i image across a neuron i involves intersecting the star with two half-spaces $x_i \geq 0$ and $x_i \leq 0$ to produce two intermediate stars Θ_j^+ and Θ_j^- respectively. These stars are checked for emptiness using an LP feasibility solver. This is done by checking if the predicates associated with the stars admit any feasible solutions. The intermediate non-empty stars then undergo a ReLU transformation across the neuron i. In subsequent layers, the input to the layers may be a collection of interval stars. We perform the above operations on each of the individual stars and obtain the reachable set across the layer as a collection of all the output stars computed. The set of stars obtained at the final layer $\mathscr{T}(\Theta)'$ represent an over-approximation of the reachable set of the INN. This is explained below.

Lemma 2: For a given input set Θ and INN $\mathscr{T}$, The output of the reachability algorithm, $\mathscr{T}(\Theta)'$ is an over-approximation of $\mathscr{T}(\Theta)$ (the reach set).

Proof: Consequence of Proposition 2. ■

We also discuss about the verification complexity of determining the safety of an INN $\mathscr{T}$ for a given input set Θ and a safety specification $\mathscr{X}$ in the below theorem.

Theorem 1 (Verification Complexity): Let $\mathscr{T}$ be a N-neuron INN , Θ be a *Interval Star* with p linear constraints and m-variables in the predicate, $\mathscr{X}$ be a safety specification with s linear constraints. In the worst case, the safety verification or falsification problem $\mathscr{T}(\Theta)' \models \mathscr{X}$ is equivalent to solving 2^N feasibility problems in which each has $N + p + s$ linear constraints and m variables.

Proof: Proof is same as that in [7] where Θ is a star set and $\mathscr{T}$ is a *NN.* ■

VI. Experiments

In this section, we perform a series of experiments to evaluate the scalability and effectiveness of our method in computing the reach set of INNs and compare it with the existing MILP based verification method for INNs [8].

Setup: Our tool is written using Python 3.8. We use the Gurobi solver for solving LP and MILP problems. We set up a python interface to MATLAB-R2022a and used CORA-2022 toolbox for all interval matrix arithmetic. All experiments were run on a virtual Ubuntu 20.04 with single core on a host Mac machine.

Range computation times for large synthetic INNs: In the following experiments, we generated INNs having 2 input neurons and 2 output neurons. The input range for both the neurons was set to $[0.5, 1.0]$. The values of the weights and biases of the INN were uniformly randomly generated from the range $[-1.0, 1.0]$. We obtained different INNs by varying the number of hidden layers and the number of neurons per hidden layer. For larger and more practical networks, the MILP method runs into issues whereas the interval star method does not.

For large networks, the MILP encoding contains large constants and the solver is unable to handle such large constants and reports that the model is unbounded. The star method does not have such a limitation and can be used on very large networks. We demonstrate few examples of this in Table I. In, Table I, 'Nodes' refers to the number of neurons per hidden layer, 'Layers' refers to number of hidden

Nodes	Layers	Star	MILP
15	15	1.311	0.291
20	20	2.283	MU
50	20	10.048	MU

TABLE I

OUTPUT RANGE COMPUTATION TIMES BETWEEN OUR METHOD AND THE MILP BASED METHOD ON RANDOMLY GENERATED INNS.

Network	Num Stars	Star time	MILP time
2_9	36	36.292	320.925
3_7	151	113.589	583.817
5_7	378	91.593	595.197

TABLE II

RANGE COMPUTATION TIMES ON INNS DERIVED FROM ACAS XU NETWORKS BY PERTURBING THE WEIGHTS AND BIASES BY 0.00001.

layers, 'Star' refers to the time (in seconds) taken by our method to compute the reach set, 'MILP' refers to the time (in seconds) taken by the MILP method. 'MU' refers to the 'Model Unbounded' message returned by the MILP solver.

Range computation time comparison on INNs derived from ACAS XU benchmarks: We consider the ACAS XU benchmarks which are neural networks with 6 hidden layers with 50 neurons each [1]. Specifically we considered the networks 2_9, 5_7, 3_7 from the VNN-COMP 2021 benchmark repository [24]. To convert these networks to INNs, we added a perturbation of 0.00001 to all the weights and biases of the network. We compute the output range of these networks using the input range $X_0 = [0.0, 0.0]$, $X_1 = [0.0, 0.009549297]$, $X_2 = [0.493380324, 0.5]$, $X_3 = [0.45, 0.5]$, $X_4 = [0.45, 0.5]$. We observe that the MILP method is considerably slower than our method on ACAS XU derived INNs. Table II presents some examples. In Table II, 'Network' refers to the index number of the network in the repository. 'Num Stars' refers to the number of stars output by the our method. 'Star time' refers to the time (in seconds) taken by our method to compute the range. 'MILP time' refers to the time (in seconds) taken by the MILP method to compute the range.

We also point out that the verification time of our approach is positively correlated with the number of stars output by the method. Our implementation is baseline and does not include any optimizations present in the star based NN verification tool nnenum [21] which can further reduce the verification time of our approach.

VII. CONCLUSIONS

We have implemented a reachability method which performs better than the existing verification method for verifying large and practical INNs. For future work, we will extend the optimizations for star based reachability used in the state-of-the-art NN verification tool nnenum [21] to our method.

REFERENCES

[1] K. D. Julian, M. J. Kochenderfer, and M. P. Owen, "Deep neural network compression for aircraft collision avoidance systems," *Journal of Guidance,Control and Dynamics*, vol. 42, no. 3, pp. 598–608, 2019.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.

[3] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, Ieee, 2017.

[4] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," *Advances in neural information processing systems*, vol. 31, 2018.

[5] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International conference on computer aided verification*, pp. 97–117, Springer, 2017.

[6] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter, "Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification," *arXiv preprint arXiv:2103.06624*, 2021.

[7] H.-D. Tran, D. Manzanas Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Star-based reachability analysis of deep neural networks," in *International symposium on formal methods*, pp. 670–686, Springer, 2019.

[8] P. Prabhakar and Z. Rahimi Afzal, "Abstraction based output range analysis for neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[9] S. Bak and P. S. Duggirala, "Simulation-equivalent reachability of large linear systems with inputs," in *International Conference on Computer Aided Verification*, pp. 401–420, Springer, 2017.

[10] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *arXiv:1706.07351*, 2017.

[11] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18, 2017.

[12] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, "Concolic testing for deep neural networks," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pp. 109–119, 2018.

[13] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, pp. 303–314, 2018.

[14] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *International Symposium on Automated Technology for Verification and Analysis*, pp. 269–286, Springer, 2017.

[15] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *International conference on computer aided verification*, pp. 3–29, Springer, 2017.

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

[17] P. Cousot, "Abstract interpretation," *ACM Computing Surveys (CSUR)*, vol. 28, no. 2, pp. 324–328, 1996.

[18] K. Ghorbal, E. Goubault, and S. Putot, "The zonotope abstract domain taylor1+," in *International conference on computer aided verification*, pp. 627–633, Springer, 2009.

[19] P. Cousot and N. Halbwachs, "Automatic discovery of linear restraints among variables of a program," in *Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pp. 84–96, 1978.

[20] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[21] S. Bak, "nnenum: Verification of relu neural networks with optimized abstraction refinement," in *NASA Formal Methods Symposium*, pp. 19–36, Springer, 2021.

[22] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[23] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to interval analysis*. SIAM, 2009.

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