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

Abstraction-based Safety Analysis of Linear Dynamical Systems with Neural Network Controllers

Ratan Lal[1] Pavithra Prabhakar[2]

Abstract— **We consider the safety verification problem of a closed-loop discrete-time linear dynamical system with a neural network controller. The crux of safety verification relies on computing output reachable sets of the dynamical system and the neural network. Reachable set computation time of the neural network grows with the network size. To address the scalability issue, our main approach consists of abstracting the neural network controller into a smaller *annotated interval neural network* (AINN), and using this to compute an over-approximation of the reachable set of the closed-loop system. We present a novel approach for output reachable set computation of an AINN by decomposing it into two reachable set computation problems on neural networks, which we then compute using star-sets. Our experimental analysis on two benchmarks demonstrate the trade-off in the precision and time for reachable set computation.**

I. Introduction

Neural networks are increasingly being deployed as controllers in safety-critical domains, which has necessitated rigorous analysis methods for guaranteeing the correct functioning of these systems. Of particular interest is safety analysis of closed loop dynamical systems with neural networks as controllers, which consists of checking if the states reached by the closed loop system after a given number of loop iterations do not intersect with an unsafe set. The crux of safety verification is the computation of the set of output valuations of the dynamical system and the neural network, which are then applied repeatedly to compute the reachable set of the closed loop system, and checked for intersection with an unsafe set.

Output reachable set computation of neural networks has received extensive attention in recent years [11]. Computation of the output set is computationally challenging (NP-complete); hence, efficient methods for computing precise over-approximations of the output set have been investigated (see related work for details). The large size of the network remains a challenge. We propose an orthogonal approach for over-approximating the input-output relation via a novel network reduction technique that is applicable to a wide range of activation functions.

A neural network is a deterministic system, that is, the output is unique given an input. To capture an over-approximation of the input-output, we require a non-deterministic system. We propose a novel data structure, *annotated interval neural network*, that allows interval weights and biases, and consist of a natural number annotation on the nodes for bookkeeping during the abstraction-refinement process. The choice of weights and biases from the corresponding intervals at each step of the computation allows multiple possible outputs for a given input. Our abstraction procedure consists of merging the nodes of a neural network based on a partitioning of the nodes. Biases and weights of the abstract system are interval hulls of the corresponding biases and weights in the concrete system. The abstraction procedure is sound for any continuous activation function. Our data structure is similar to the interval neural networks [15] with the addition of annotations on the nodes, however, the abstraction construction is simplified as we do not need scaling the interval hull of the weight by a factor equivalent to the number of nodes merged in the abstract source node. This is circumvented through the additional annotations. More interestingly, we can show that the abstraction is unique given a partition, irrespective of the sequence of abstractions performed to reach a certain partition, which is violated by the abstraction in [15]. Further, our abstraction technique is applicable to any continuous activation function; however, we need verification methodologies for analyzing the abstract INNs.

We present a novel algorithms for computing the output reachable set of the annotated interval neural network (AINN) $\mathcal{N}$, which consists of decomposing the reachable set computation problem on $\mathcal{N}$ into two reachable set computation problems on standard neural networks. More precisely, reachable set for AINN $\mathcal{N}$ is obtained by a convex combination of the reachable set of two neural networks $\mathcal{N}_l$ and $\mathcal{N}_u$, where $\mathcal{N}_l$ and $\mathcal{N}_u$ represent the neural networks with the same architecture as $\mathcal{N}$ but with weights and biases that are the lower and upper bounds of the corresponding intervals. We use the star-set based method [20] to compute the reachset of $\mathcal{N}_l$ and $\mathcal{N}_u$ and use that to compute the reachset of $\mathcal{N}$, and then use that to compute the reachset for the linear dynamics (linear transformations of star sets can be computed efficiently). We iterate the procedure k times to compute the k-step reachable set of the closed loop linear system and use that for safety analysis.

We have implemented our abstraction based safety analysis procedure in a Python toolbox, and evaluated on two benchmarks, namely, ACAS XU and the Translational Oscillators with Rotating Actuator (TORA). Our experimental results illustrate the trade-off between the size of the abstraction, the verification time, and the safety conclusions.

[1]Ratan Lal is a faculty in the school of computer science and information systems at Northwest Missouri State University, USA rlal@nwmissouri.edu

[2]Pavithra Prabhakar is a faculty in the department of computer science at Kansas State University, USA pprabhakar@ksu.edu

2023 62nd IEEE Conference on Decision and Control (CDC) | 979-8-3503-0124-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/CDC49753.2023.10383593

II. Related Work

Verification of neural networks has gained momentum in recent years [1], [2]. In recent decades, neural network has been popularly used for the controller of cyber physical systems. Although many safety verification of neural network controlled dynamical systems [10], [18], [19] have been proposed, the main challenge is the safety verification of neural networks. For the neural networks, satisfiability solving based approaches that encode the verification problem as satisfiability modulo theory (SMT) solving problem, have been investigated. In particular, Reluplex [12], Planet [6], NeVer [17], and VeriDeep/DLV [9] can perform satisfiability checking of SMT formulas with linear constraints and ReLU operations. An SMT based counter-example guided abstraction refinement (CEGAR) approach has been explored in [16]. A reduction to efficient mixed integer linear programming (MILP) for reachability analysis have been explored in [3]. Methods for over-approximation of the reachable set of neural networks [8], including abstract interpretation [5] based methods have been explored. Approaches based on convex optimization [14], interval analysis [21], and linear approximation [22] have been explored.

In this paper, we present a safety analysis of neural network-controlled linear dynamical systems based on a network reduction technique and reachability analysis of neural networks and linear dynamical systems.

III. Preliminaries

a) Numbers and Sets: Let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{IR}$ and $\mathbb{N}$ denote the set of real numbers, the set of positive real numbers, the set of all real intervals, and the set of natural numbers, respectively. We say that an interval $[a, b] \in \mathbb{IR}$ is singular if $a = b$. Given an interval $[a, b]$, we use $[a, b]_l$ to denote a and $[a, b]_u$ to denote b. We use $[n]$ and $(n]$ to denote the set $\{0, 1, \ldots, n\}$ and $\{1, 2, \ldots, n\}$, respectively. Given a set $\mathcal{S}$, we use $|\mathcal{S}|$ to denote the number of elements in the set.

b) Relations and Functions: Given three sets $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, and relations $R_1 \subseteq \mathcal{S}_1 \times \mathcal{S}_2$, $R_2 \subseteq \mathcal{S}_2 \times \mathcal{S}_3$, the composition of R_1 and R_2 denoted by $R_1 \diamond R_2$ is the subset $\{(s_1, s_3) \in \mathcal{S}_1 \times \mathcal{S}_3 \mid \exists\ s_2 \in \mathcal{S}_2,\ (s_1, s_2) \in R_1,\ (s_2, s_3) \in R_2\}$ of $\mathcal{S}_1 \times \mathcal{S}_3$. A function $f : \mathcal{S}_1 \to \mathcal{S}_2$ is bijection if it is one-one and onto function. Given functions $f : \mathcal{S}_1 \to \mathcal{S}_2$ and $g : \mathcal{S}_2 \to \mathcal{S}_3$, we use $g \circ f$ for the composition of the functions f and g, that is, $g \circ f : \mathcal{S}_1 \to \mathcal{S}_3$, where $g \circ f(s_1) = g(f(s_1))$. Given a set $\mathcal{S}$, a valuation on $\mathcal{S}$ is a function $f : \mathcal{S} \to \mathbb{R}$. We will use $Val(\mathcal{S})$ to denote the set of all valuations on $\mathcal{S}$.

c) Convex Hull: Let $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ be subsets of $\mathbb{R}^n$. Convex hull of $\mathcal{S}_i$s, $i \in (n]$ denoted by $\mathrm{CH}(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n)$ is the smallest convex set that contains all $\mathcal{S}_i$s, $i \in (n]$.

IV. Interval Neural Networks

Recall that a neural network (NN) is a computational model, that consists of an input layer, one or more hidden layers, and an output layer of neurons (or nodes). Each neuron in the hidden layers and the output layer is labeled with a bias and an activation function. Each edge between two neurons in two consecutive layers is labeled with a weight. In this paper, we consider interval neural networks (INNs), which extend NNs, by allowing interval values for biases and edges, and annotating the neurons in the hidden and output layers with a natural number, that allows to keep relationship between concrete and abstract neurons. Next, we formally define the INN.

Definition 1: An interval neural network (INN) is a tuple $\mathcal{N} = \big(k, \mathcal{A}ct, \{\mathcal{S}_i\}_{i\in[k]}, \{\mathcal{L}_i\}_{i\in[k]}, \{\mathcal{W}_i\}_{i\in(k]}, \{b_i\}_{i\in(k]}, \{\mathcal{A}_i\}_{i\in(k]}\big)$, where

- k is the number of layers;
- $\mathcal{A}ct$ is a set of activation functions;
- For each $i \in [k]$,
  - $\mathcal{S}_i$ is a set of neurons in layer i;
  - $\mathcal{L}_i : \mathcal{S}_i \to \mathbb{N}$ is a function that annotates each neuron in layer i with a natural number;
- For each $i \in (k]$,
  - $\mathcal{W}_i : \mathcal{S}_{i-1} \times \mathcal{S}_i \to \mathbb{IR}$ is a function that assigns a real number interval to each edge between layer $i-1$ and layer i;
  - $b_i : \mathcal{S}_i \to \mathbb{IR}$ is a function that assigns a real number interval to each neuron in layer i;
  - $\mathcal{A}_i : \mathcal{S}_i \to \mathcal{A}ct$ is a function that assigns one of the activation functions from the set $\mathcal{A}ct$ to each neuron in layer i.

In the rest of this paper, we will use $\mathcal{N}$ to denote an interval neural network. We consider the class of activation functions to be the class of continuous functions from $\mathbb{R}$ to $\mathbb{R}$, such as Relu, Sigmoid, and Tanh function.

We capture the semantics of INN as a set of pairs of input-output valuations, where the output valuation corresponds to a possible valuation of the output layer, given the input valuation.

The next definition formalizes the semantics of INN for a particular layer, which is then iteratively composed to obtain the input-output relation of the INN. Note that while the NN semantics is deterministic, the interval weights and biases render the semantics of INN non-deterministic. However, this is needed to express over-approximations of neural network semantics.

Definition 2: Given an INN $\mathcal{N}$ and a layer index $i \in (k]$, we define the semantics of $\mathcal{N}$ for the layer i, denoted by $[\![\mathcal{N}]\!]_i$ as follow:

$$[\![\mathcal{N}]\!]_i = \{(v, v') \in Val(\mathcal{S}_{i-1}) \times Val(\mathcal{S}_i) \mid \forall\ s' \in \mathcal{S}_i,$$
$$\exists\ \{w_{s,s'}\}_{s\in\mathcal{S}_{i-1}} \in \{\mathcal{W}_i(s, s')\}_{s\in\mathcal{S}_{i-1}},\quad b_{s'} \in b_i(s') \text{ s.t.}$$
$$v'(s') = \mathcal{A}_i(s')\Big(\sum_{s\in\mathcal{S}_{i-1}} w_{s,s'}\mathcal{L}_{i-1}(s)v(s)\Big) + b_{s'}\Big)\}$$

We define the semantics of INN $\mathcal{N}$ as a composition of the semantics $[\![\mathcal{N}]\!]_i$s, $i \in (n]$, where $[\![\mathcal{N}]\!]_i$ captures all valid pairs of valuations for the neurons in layer $i-1$ and i.

Definition 3: Given an INN $\mathcal{N}$, its semantics denoted by $[\![\mathcal{N}]\!]$, is defined as:

$$[\![\mathcal{N}]\!] = [\![\mathcal{N}]\!]_1 \diamond [\![\mathcal{N}]\!]_2 \diamond \ldots \diamond [\![\mathcal{N}]\!]_k$$

V. Linear Dynamical Systems

In this section, we formalize an n-dimensional discrete-time linear dynamical system given as:

$$x(k+1) = Ax(k) + Bu(k), \tag{1}$$

using explicit state and input variables. Here, $x(k)$ and $u(k)$ denote the state and input at time k, and A and B are matrices. So, the state at time $k+1$ is a linear function of the state and input at time k. Instead of treating state and inputs at time k as vectors, we treat them in the following formalization as valuations for state and input variables.

Definition 4: A linear dynamical system (LDS) is a tuple $\mathcal{D} = (\mathcal{S}_\mathcal{X}, \mathcal{S}_\mathcal{I}, \mathcal{M}_A, \mathcal{M}_B, \mathbb{S}, \mathbb{I})$, where

- $\mathcal{S}_\mathcal{X}$ is a set of state variables;
- $\mathcal{S}_\mathcal{I}$ is a set of input variables;
- $\mathcal{M}_A : \mathcal{S}_\mathcal{X} \times \mathcal{S}_\mathcal{X} \to \mathbb{R}$ is a function that assign a real value for each pair of state variables;
- $\mathcal{M}_B : \mathcal{S}_\mathcal{X} \times \mathcal{S}_\mathcal{I} \to \mathbb{R}$ is a function that assign a real value for each pair of state and input variable;
- $\mathbb{S} \subseteq \{v \mid v : \mathcal{S}_\mathcal{X} \to \mathbb{R}\}$ is a set of initial valuations for the state variables in $\mathcal{S}_\mathcal{X}$.
- $\mathbb{I} \subseteq \{v \mid v : \mathcal{S}_\mathcal{I} \to \mathbb{R}\}$ is a set of valuations for the input variables in $\mathcal{S}_\mathcal{I}$.

Next, we define the semantics of LDS that captures the next state valuation given an initial state and an input valuation.

Definition 5: Given an n-dimensional LDS $\mathcal{D} = (\mathcal{S}_\mathcal{X}, \mathcal{S}_\mathcal{I}, \mathcal{M}_A, \mathcal{M}_B, \mathbb{S}, \mathbb{I})$, we define the semantics of $\mathcal{D}$ as follow: $[\![\mathcal{D}]\!] = \{(v_1, v, v_2) \mid v_1 \in \mathbb{S},\ v \in \mathbb{I},\ \forall\ x \in \mathcal{S}_\mathcal{X},$

$$v_2(x) = \sum_{x' \in \mathcal{S}_\mathcal{X}} \mathcal{M}_A(x, x') v_1(x') + \sum_{u \in \mathcal{S}_\mathcal{I}} \mathcal{M}_B(x, u) v(u)\}$$

VI. Linear Dynamical Systems with INN

In this section, we introduce the linear dynamical systems controlled by interval neural networks. Here, the input to the linear dynamical system $x(k+1) = Ax(k) + Bu(k)$ is provided by the output of the neural network $\mathcal{N}$ which in turn takes the state of the linear dynamical system as input as shown in Figure 1. Our broad objective is to analyze systems in which neural networks control the dynamical systems. However, the analysis can be expensive when we have large neural networks. We will present an abstraction of a neural network into a small interval neural network in order to scale the analysis. Next, we formally define syntax and semantics of a linear dynamical system with an INN.

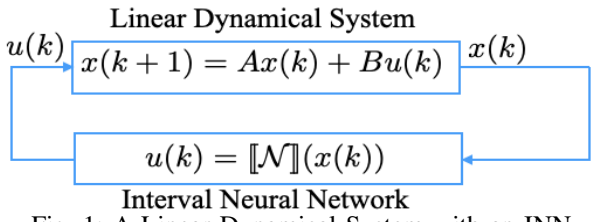

Fig. 1: A Linear Dynamical System with an INN

Definition 6: A linear dynamical system with an interval neural network (LDSINN) is a tuple $\mathcal{C} = (\mathcal{D}, \mathcal{N})$ where

- $\mathcal{D} = (\mathcal{S}_\mathcal{X}, \mathcal{S}_\mathcal{I}, \mathcal{M}_A, \mathcal{M}_B, \mathbb{S}, \mathbb{I})$ is a linear dynamical system;
- $\mathcal{N} = \big(k, \mathcal{A}ct, \{\mathcal{S}_i\}_{i\in[k]}, \{\mathcal{L}_i\}_{i\in[k]}, \{\mathcal{W}_i\}_{i\in(k]}, \{b_i\}_{i\in(k]}, \{\mathcal{A}_i\}_{i\in(k]}\big)$ is an interval neural network;
- $\mathcal{S}_\mathcal{X} = \mathcal{S}_0$ and $\mathcal{S}_\mathcal{I} = \mathcal{S}_k$.

Definition 7: Given an LDSINN $\mathcal{C} = (\mathcal{D}, \mathcal{N})$, we define the semantics of $\mathcal{C}$ as follows: $[\![\mathcal{C}]\!] = \{(x_1, x_2) \mid$
$\exists\ u \in Val(\mathcal{S}_k),\ (x_1, u) \in [\![\mathcal{N}]\!],\ (x_1, u, x_2) \in [\![\mathcal{D}]\!]\}$.

Our broad objective is to perform safety analysis of linear dynamics systems with neural network controllers for a finite time horizon $k \in \mathbb{N}$. This require us to define the semantics of k-composition of a LDSINN $\mathcal{C} = (\mathcal{D}, \mathcal{N})$ denoted by $[\![(\mathcal{C}, k)]\!]$ that captures all pairs of initial state valuation and state valuation at k^{th} iteration of evolution through $\mathcal{D}$ and $\mathcal{N}$. $[\![(\mathcal{C}, k)]\!]$ is formally defined as follows:

$$[\![(\mathcal{C}, k)]\!] = \underset{1}{[\![\mathcal{C}]\!]} \diamond \underset{2}{[\![\mathcal{C}]\!]} \diamond \ldots \diamond \underset{k}{[\![\mathcal{C}]\!]}.$$

Problem 1: [Safety Problem] Given an LDSINN $\mathcal{C} = (\mathcal{D}, \mathcal{N})$, a set of unsafe state valuations $\mathbb{U}$, and number of steps $k \in \mathbb{N}$, verify whether there exist initial state valuation $v \in \mathbb{S}$ and $i \in (k]$ such that for some v', $(v, v') \in [\![(\mathcal{C}, i)]\!]$, and $v' \in \mathbb{U}$.

The crux for the safety analysis lies in computing all valid pairs of initial state and state valuation for k-composition of LDSINN that consists of two computational steps, namely, (a) computation of control inputs from INN $\mathcal{N}$; (b) computation of the set of next state valuations with respect to the set of current state valuations and the set of control input valuations obtained from $\mathcal{N}$. The step (a) is computationally expensive, specifically, for the large neural networks. Hence, we present an abstraction technique based on a partitioning of the neuron-space that is sound. The abstraction technique reduces the large neural network into a smaller neural network expressed in the form of INN. The step (b) can be performed by linear transformation operation over current state and control input valuations.

VII. Abstraction of INN

In this section, we present an abstraction procedure that reduces a large INN $\mathcal{N}$ (or neural network) into a small INN $\widehat{\mathcal{N}}$ such that $\widehat{\mathcal{N}}$ is an over-approximation of $\mathcal{N}$. The procedure is based on partitioning the nodes in each layer of INN into a finite number of sets, and considering these sets as abstract nodes. We start with some preliminaries.

Definition 8 (Partition): A partition of an INN $\mathcal{N}$ is a sequence $\mathcal{P} = \{\mathcal{P}_i\}_{i\in[k]}$, where $\mathcal{P}_i$ is a partition of $\mathcal{S}_i$ satisfying the following conditions:

- for each $\mathcal{S} \in \mathcal{P}_i$, if $s_1, s_2 \in \mathcal{S}$, then $\mathcal{A}_i(s_1) = \mathcal{A}_i(s_2)$;
- $\mathcal{P}_0 = \{\{s\} \mid s \in \mathcal{S}_0\}$, $\mathcal{P}_k = \{\{s\} \mid s \in \mathcal{S}_k\}$.

The broad abstraction procedure consists of merging the nodes of a particular set of a partition into an abstract node and instantiating the biases and the weights such that over-approximation is guaranteed. For the bias on an abstract node, we consider the convex hull of the biases associated with the nodes in the set corresponding to the abstract node. The weights on abstract edge is a bit tricky. As shown in [15], a convex hull of the weight intervals associated with the concrete edges corresponding to the abstract edges does not provide an over-approximation; however, "scaling"

the interval by the number of nodes being merged in the source provides an over-approximation. However, such an over-approximation is not conducive to refinement, since, depending on the sequence of partitions used to construct the abstraction (with the "same" final partition), one ends up with different abstractions. Hence, we avoid the scaling, and instead store this information in the annotations, which provides a unique abstraction corresponding to a partition, irrespective of the sequence of node mergings applied to arrive at that abstraction. More precisely, for the abstract edge weight, we consider only the convex hull of the corresponding concrete edge weights (without any scaling) and annotate the abstract node with the number of merged nodes. Note that our semantics for INN has the effect of scaling with respect to the number of merged incoming nodes. Next, we provide the formal definition of the abstraction construction.

Definition 9: Given an INN $\mathcal{N}$ and its partition $\mathcal{P} = \{\mathcal{P}_i\}_{i\in[k]}$, we define an abstract INN $\mathcal{N}/\mathcal{P} = \big(k, \mathcal{A}ct, \{\widehat{\mathcal{S}}_i\}_{i\in[k]}, \{\widehat{\mathcal{L}}_i\}_{i\in[k]}, \{\widehat{\mathcal{W}}_i\}_{i\in(k]}, \{\widehat{b}_i\}_{i\in(k]}, \{\widehat{\mathcal{A}}_i\}_{i\in(k]}\big)$, where

- for each $i \in [k]$, (a) $\widehat{\mathcal{S}}_i = \mathcal{P}_i$; (b) for each $\widehat{s} \in \widehat{\mathcal{S}}_i$, $\widehat{\mathcal{L}}_i(\widehat{s}) = \sum_{s\in\widehat{s}} \mathcal{L}_i(s)$;
- for each $i \in (k]$,
  - for each $\widehat{s} \in \widehat{\mathcal{S}}_{i-1}$, $\widehat{s}' \in \widehat{\mathcal{S}}_i$, $\widehat{\mathcal{W}}_i(\widehat{s},\widehat{s}') = \mathrm{CH}(\{\mathcal{W}_i(s,s') \mid s \in \widehat{s},\ s' \in \widehat{s}'\})$;
  - for each $\widehat{s} \in \widehat{\mathcal{S}}_i$, $\widehat{b}_i(\widehat{s}) = \mathrm{CH}(\{b_i(s) \mid s \in \widehat{s}\})$;
  - for each $\widehat{s} \in \widehat{\mathcal{S}}_i$, $\widehat{\mathcal{A}}_i(\widehat{s}) = \mathcal{A}_i(s)$ for some $s \in \widehat{s}$.

In the rest of this paper, we use $\widehat{\ }$ to denote the components of the abstract INN $\mathcal{N}/\mathcal{P}$ as defined in Definition 9 in order to distinguish from the components of $\mathcal{N}$. Next, we define an abstraction function for establishing a relation between the valuations of the concrete and abstract systems. The value for an abstract node is a weighted average of the values of concrete nodes, where weights are annotated values associated with the concrete nodes.

Definition 10: Given an INN $\mathcal{N}$ and its abstraction $\mathcal{N}/\mathcal{P}$, we define an abstraction function $\alpha_{(\mathcal{N},\mathcal{P})} = \{\alpha^i_{(\mathcal{N},\mathcal{P})}\}_{i\in[k]}$, where $\alpha^i_{(\mathcal{N},\mathcal{P})} : Val(\mathcal{S}_i) \to Val(\widehat{\mathcal{S}}_i)$ such that

$$\alpha^i_{(\mathcal{N},\mathcal{P})}(v)(\widehat{s}) = \frac{\sum_{s\in\widehat{s}} \mathcal{L}_i(s)v(s)}{\widehat{\mathcal{L}}_i(\widehat{s})}.$$

Remark 1: We avoid subscripts $\mathcal{N}$, $\mathcal{P}$ and superscript i from $\alpha^i_{(\mathcal{N},\mathcal{P})}$, when it is clear from the context.

Next, we show that given an input-output valuation (v, v') of an INN $\mathcal{N}$, its mapping under the abstraction function α, that is, $(\alpha(v), \alpha(v'))$ will be an input-output valuation of the abstract INN $\mathcal{N}/\mathcal{P}$, which is stated in the following theorem.

Theorem 1: Given an INN $\mathcal{N}$ and its abstraction $\mathcal{N}/\mathcal{P}$, we have the following:

$$\text{if } (v,v') \in [\![\mathcal{N}]\!], \text{ then } (\alpha(v),\alpha(v')) \in [\![\mathcal{N}/\mathcal{P}]\!].$$

In order to compare the input-output relations of two INNs, we need to be able to map input and output valuations, of the two systems. We do this by creating bijections between the input and the output neurons of the two INNs.

Definition 11: Let $F_I : \mathcal{S}_I \to \mathcal{S}'_I$ and $F_O : \mathcal{S}_O \to \mathcal{S}'_O$ be two bijective functions, and $R \subseteq Val(\mathcal{S}'_I) \times Val(\mathcal{S}'_O)$ be a binary relation. We define $R/(F_I, F_O) = \{(v_1 \circ F_I, v_2 \circ F_O) \mid (v_1, v_2) \in R\}$.

We define an ordering on INNs using over-approximation relationship between two INNs.

Definition 12: Given two INNs $\mathcal{N}_1$ and $\mathcal{N}_2$, we say $\mathcal{N}_2$ is an over-approximation of $\mathcal{N}_1$, denoted by $\mathcal{N}_1 \leqslant \mathcal{N}_2$ if there are bijection functions $F_I : \mathcal{S}^1_0 \to \mathcal{S}^2_0$ and $F_O : \mathcal{S}^1_k \to \mathcal{S}^2_k$ such that $[\![\mathcal{N}_1]\!] \subseteq [\![\mathcal{N}_2]\!]/(F_I, F_O)$.

Given a partition $\mathcal{P}$, abstract INN $\mathcal{N}/\mathcal{P}$ is always an over-approximation of $\mathcal{N}$, which is stated in the following theorem.

Theorem 2: Given an INN $\mathcal{N}$ and its partition $\mathcal{P}$, we have $\mathcal{N} \leqslant \mathcal{N}/\mathcal{P}$.

VIII. Construction of $\mathcal{N}_l$ and $\mathcal{N}_u$

In this section, we present a new approach for computing the reachable set of an interval neural network by reducing it to the verification of the traditional neural networks. Our broad idea is that the reach set of an INN can be represented as a convex hull of reach sets of two neural networks, namely N_l and $\mathcal{N}_u$, referred to as the lower and upper neural networks. This holds as long as the inputs are all positive, or more generally, when all the inputs for a particular node have the same sign. We decompose the input set into a finite number of sets based on the signs of the inputs to generalize this observation.

For simplicity, first we consider the case where the state valuations are all positive. $\mathcal{N}_l$ is the neural network obtained by considering the lower bound of weights and biases of INN $\mathcal{N}$. $\mathcal{N}_u$ is the neural network constructed by considering the upper bound of weights and biases of INN $\mathcal{N}$.

Definition 13: Given an INN $\mathcal{N}$, lower/upper neural network $\mathcal{N}_l/\mathcal{N}_u$ is defined as the same structure of $\mathcal{N}$ except:

- For $a \in \{l, u\}$,
  - For each $i \in (k]$, for $s \in \mathcal{S}^a_{i-1}$, $s' \in \mathcal{S}^a_i$, $\mathcal{W}^a_i(s,s') = [\mathcal{L}_i(s) * \mathcal{W}_i(s,s')_a, \mathcal{L}_i(s) * \mathcal{W}_i(s,s')_a]$;
  - For each $i \in (k]$, for $s \in \mathcal{S}^a_i$, $b^a_i(s) = [b_i(s)_a, b_i(s)_a]$.

Next, we establish the relationship between output valuation of the INN $\mathcal{N}$, $\mathcal{N}_l$ and $\mathcal{N}_u$ with respect to an input valuation, which is stated in the following theorem.

Theorem 3: Let $\mathcal{N}$ be an INN, and $v \in Val(\mathcal{S}_0)$ such that for all $s \in \mathcal{S}_0$, $v(s) \geq 0$. Then, we have

$$(v,v') \in [\![\mathcal{N}]\!] \text{ iff } \exists\ v^l \in Val(\mathcal{S}^l_k),\ v^u \in Val(\mathcal{S}^u_k) \text{ such that } (v,v^l) \in [\![\mathcal{N}_l]\!], (v,v^u) \in [\![\mathcal{N}_u]\!] \text{ and for all } s \in \mathcal{S}_k,\ v^l(s) \leq v'(s) \leq v^u(s).$$

Next, we provide a construction of $\mathcal{N}_l$ and $\mathcal{N}_u$ for the general case. For the ReLU activation function, we know that all the valuations of neurons in layer 1 or higher are positive. Hence, for N_l and $\mathcal{N}_u$, to compute weights and biases from layer 1, we use Definition 13. To find weights between input neurons and neurons in the first layer for both N_l and N_u, we first divide the input set into regions such that the sign of all the valuations in the same region for any input neuron is same, that is, $\forall\ s \in \mathcal{S}_0, \mathbb{I}(s) \cap \mathbb{R}_{\geq 0} = \emptyset$ or $\forall\ s \in \mathcal{S}_0, \mathbb{I}(s) \cap \mathbb{R}_{<0} = \emptyset$. The upper and lower bound

of the weight are chosen based on the sign of the valuation of the incoming node.

Let us define for any $x \in \mathbb{R}$, $sign(x)$ is positive if $x >= 0$ and negative otherwise. We define an equivalence relation on the input set $\mathbb{I}$ as follows. $v_1, v_2 \in \mathbb{I}$ are equivalent if for every input neuron $s \in \mathcal{S}_0$, $sign(v_1(s)) = sign(v_2(s))$. Let $\mathbb{I}_1, \ldots, \mathbb{I}_m$ be the partition of $\mathbb{I}$ corresponding to the equivalence relation. Note that m could be $2^{|\mathcal{S}_0|}$ in the worst case. Next we define for each $\mathbb{I}_j$, upper and lower bound neural networks, $\mathcal{N}_l^j$ and $\mathcal{N}_u^j$. The neural networks $\mathcal{N}_l^j$ and $\mathcal{N}_u^j$ are the same except the weight on layer 1, that is, $\mathcal{W}_1^{l,j}$ and $\mathcal{W}_1^{u,j}$. For $s \in \mathcal{S}_0$, let $sign(\mathbb{I}_j, s)$ be the sign of $v(s)$ for some $v \in \mathbb{I}_j$. Note that for all the valuations $v \in \mathbb{I}_j$, $v(s)$ always have the same sign. Then, we use the following for $\mathcal{W}_1^{l,j}$ and $\mathcal{W}_1^{u,j}$. For $s \in \mathcal{S}_0$, for $s' \in \mathcal{S}_1$,
(a) if $sign(\mathbb{I}_j, s)$ is positive, then $\mathcal{W}_1^{l,j}$ and $\mathcal{W}_1^{u,j}$ are the same as defined in Definition 13;
(b) if $sign(\mathbb{I}_j, s)$ is negative, then $\mathcal{W}_1^{l,j}(s, s')$ and $\mathcal{W}_1^{u,j}(s, s')$ will be interchanged, that is, lower and upper weight for the edge (s, s') will be $\mathcal{W}_1^{u,j}(s, s')$ and $\mathcal{W}_1^{l,j}(s, s')$, respectively.

IX. Safety Analysis

The broad approach to safety analysis consists of computing the reachable set of the closed loop system and checking if the reachable set has an empty intersection with the unsafe set. The reach set computation consists of the following two steps: compute the set of all control inputs that are output by the neural network for a given set of state valuations $\mathbb{S}$ given as input to the neural network, that is, compute $\mathbb{O}_{\mathcal{N}}^{\mathbb{S}} = \{(v' \mid \exists\ v \in \mathbb{S}, (v, v') \in [\![\mathcal{N}]\!]\}$; and compute the reach set of $\mathcal{D}$ for a given set of initial state valuations $\mathbb{S}$ and a set of input valuations $\mathbb{I}$ output by the neural network, denoted by $\mathbb{O}_{\mathcal{D}}^{\mathbb{S},\mathbb{I}}$, and given by: $\mathbb{O}_{\mathcal{D}}^{\mathbb{S},\mathbb{I}} = \mathcal{M}_A\mathbb{S} + \mathcal{M}_B\mathbb{I}$. One of the efficient methods to compute $\mathbb{O}_{\mathcal{N}}^{\mathbb{S}}$ is based on star-set representations of polyhedra [20], where both exact and over-approximate methods are provided as a trade-off between precision and computation time. $\mathbb{O}_{\mathcal{D}}^{\mathbb{S},\mathbb{I}}$ can be computed by taking linear transformation and Minkowski sum of the star-sets which can again be performed efficiently.

Our broad approach for safety analysis is shown in Figure 2. It relies on the basic idea of iteratively propagating star-sets through the neural network and linear dynamics. However, to address the scalability challenges arising with large neural networks, it incorporates the abstraction based approach proposed in the paper in Section VII. First, a neural network is abstracted, potentially resulting in a small interval neural network, which is further decomposed into upper and lower neural networks. Exact or over-approximate reachability computation methods are used to compute the individual reachable sets and their convex hull is computed that provides the exact or over-approximate reachable set of the interval neural network. Then the reachable sets with respect to the linear dynamics is computed. These steps are repeated for the desired number of iterations.

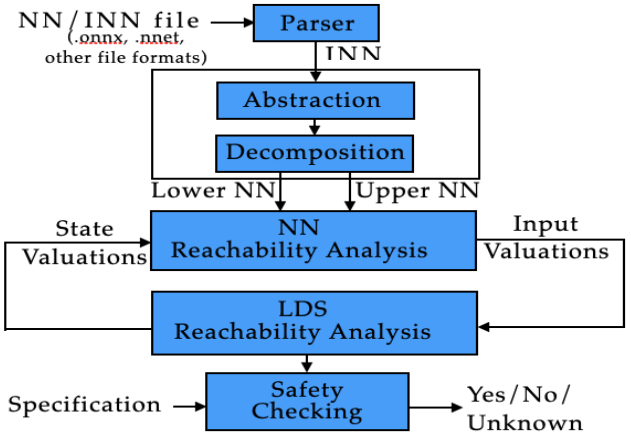

Fig. 2: Safety Analysis Framework

X. Experimental Analysis

In this section, we report our experimental evaluation on two benchmarks, namely ACAS Xu and the Translational Oscillators with Rotating Actuator (TORA). We have implemented the star-set based approach in Figure 2 in a Python toolbox. All the experiments were performed with Ubuntu 18.04 OS, Intel® core™ i7-4870HQ CPU @2.50GHz, 8GB RAM.

ACAS Xu. ACAS Xu is a neural network controller for an unmanned aircraft [13] designed for collision avoidance. It consists of 6 hidden layers with 50 neurons in each layer. We have performed safety analysis of the property that the output neuron corresponding to the control input Clear of Conflict (COC) is less than 10^{12} [12] on 4 neural networks of ACAS Xu benchmarks which is reported in Table I. We start with an abstraction, where each abstract neuron corresponds to merging 8 neurons. Next, we systematically construct refinements, that is, more precise abstractions refinement-1 and refinement-2, where refinement-1 corresponds to merging 4 neurons into an abstract neuron, and refinement-2 corresponds to merging 2 neurons into an abstract neuron. We report for each of the fours neural networks, *N1*, *N2*, *N3*, and *N4*, (1) *Time*, the time taken in seconds for computing the reach set and determining if the property is satisfied, and (2) *Range*, that represents the output range for the neuron Clear of Conflict (COC). We observe that subsequent refinements take more time to compute the reach set as the size of the abstract networks grows, however, the computed ranges (which represent the reachable values of a specific output neuron) become more precise. We note that Abstraction and Refinement-1 are not sufficient to prove the property (COC is less than 10^{12}), while Refinement-2 is more precise and is sufficient to prove the property in all the cases. This demonstrates the trade-off between reachable set computation time and its precision.

		N1		*N2*		*N3*		*N4*	
	Size	*Time*	*Range*(x_1)	*Time*	*Range* (x_1)	*Time*	*Range* (x_1)	*Time*	*Range* (x_1)
Abstraction	8	1.20	2.97e+15	1.19	2.87+e14	1.17	1.09e+15	1.16	1.49e+14
Refinement-1	13	1.98	7.67e+13	2.03	1.03e+13	2.02	4.19e+13	2.02	4.62e+12
Refinement-2	25	3.78	3.26e+11	3.71	1.54e+11	5.62	2.99e+11	3.74	3.92e+10

TABLE I: Verification of ϕ_1 over ACASXU Neural Networks

The Translational Oscillators with Rotating Actuator (TORA). TORA is an underactuated system which has one actuated rotor and one unactuated translational cart [7]. The

dynamics consists of 4 state variables and one input variable, and is given below:

$$x_1[k+1] = x_2[k], \quad x_2[k+1] = -x_1[k] + x_3[k],$$
$$x_3[k+1] = x_4[k], \quad x_4[k+1] = u.$$

The input u is computed by a neural network that consists of 4 input neurons corresponding to the state variables, 3 hidden layers each with 100 neurons, and 1 output neuron corresponding to the input variable. We report the details of the verification time in Table II for the original, abstract and refined neural network, for different number of steps, where abstract system has 7 neurons and refined system has 13 neurons in each hidden layer.

	Original(100 Neurons)		Abstraction (7 Neurons)		Refinement (13 Neurons)	
K	*Time*	*Range*(x_4)	*Time*	*Range*(x_4)	*Time*	*Range*(x_4)
2	13.73	[2.64, 2.84]	0.11	[6.0, 3.11e+07]	0.23	[5.0, 6.8e+06]
4	19.67	[3.25, 3.38]	0.47	[0, 2.28e+14]	2.21	[0, 1.06e+13]
6	16.86	[2.80, 2.94]	0.96	[0, 1.67e+21]	3.42	[0, 1.66e+19]
8	14.29	[3.64,2.84]	1.46	[0, 7.79e+20]	4.20	[0, 1.23e+18]

TABLE II: Computational Analysis over Different Abstractions

In Table II, K represents the number of iterations through the control loop, *Time* represents the average time in seconds for propagating the reach set through the control loop and *Range* represents the range of values of the output neuron corresponding to the state variable x_4 after k iterations.

We observe that when we increase the value of K, the average time per iteration increases as a result of the growth in the number of star-sets representing the reachable sets in subsequent iterations due to splits caused by the ReLU operations. Most of this time is consumed in propagating the reach set through the neural network, and the propogation of the reach set through the dynamics is fast. As before the average time increases and the range becomes smaller with subsequent refinements.

XI. Conclusions

In this paper, we present an abstraction based safety analysis framework for neural network controlled dynamical systems. We present a novel abstraction technique and a novel reachable set computation algorithm for the annotated interval neural networks. Our experimental analysis demonstrates the benefits of the abstraction based safety analysis approach. We noticed that the success of the approach critically relies on the specific abstractions. Our future work will investigate mechanisms for exploring the abstractions in a systematic manner, for instance, using methods such as counter-example guided abstraction refinement [4].

XII. Acknowledgement

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

References

[1] Rudy Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. Piecewise linear neural network verification: A comparative study. *CoRR*, 2017.

[2] Rudy Bunel, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. Piecewise linear neural networks verification: A comparative study. 2018.

[3] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA*, 2017.

[4] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *Proceedings of the International Conference on Computer Aided Verification*, pages 154–169. Springer, 2000.

[5] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Symposium on Principles of Programming Languages*, 1977.

[6] Rüdiger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis*, pages 269–286, Cham, 2017. Springer International Publishing.

[7] Bingtuan Gao, Honggang Song, Jianguo Zhao, and Chengya Gong. Dynamics and energy-based control of tora system on a slope. In *2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*. IEEE, 2013.

[8] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy, SP, year = 2018*.

[9] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 1989.

[10] Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Transactions on Embedded Computing Systems*, 2019.

[11] Xiaowei Huang, Daniel Kroening, Marta Z. Kwiatkowska, Wenjie Ruan, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. Safety and trustworthiness of deep neural networks: A survey. *CoRR*, abs/1812.08342, 2018.

[12] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 2017.

[13] Mykel J Kochenderfer, Jessica E Holland, and James P Chryssanthacopoulos. Next-generation airborne collision avoidance system. Technical report, Massachusetts Institute of Technology-Lincoln Laboratory Lexington United States, 2012.

[14] J. Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017.

[15] Pavithra Prabhakar and Zahra Rahimi Afzal. Abstraction based output range analysis for neural networks. 2019.

[16] Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *Computer Aided Verification*, pages 243–257, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[17] Luca Pulina and Armando Tacchella. Never: a tool for artificial neural networks verification. *Annals of Mathematics and Artificial Intelligence*, 2011.

[18] Jagannathan Sarangapani. *Neural network control of nonlinear discrete-time systems*. CRC press, 2018.

[19] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 147–156, 2019.

[20] Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T Johnson. Star-based reachability analysis of deep neural networks. In *International Symposium on Formal Methods*. Springer, 2019.

[21] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *27th USENIX Security Symposium*, 2018.

[22] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*. 2018.