Output Reachable Set Estimation and Verification for Multi-Layer Neural Networks

Weiming Xiang, *Senior Member, IEEE*, Hoang-Dung Tran *Member, IEEE*, and Taylor T. Johnson *Member, IEEE*

***Abstract*—In this paper, the output reachable estimation and safety verification problems for multi-layer perceptron neural networks are addressed. First, a conception called maximum sensitivity in introduced and, for a class of multi-layer perceptrons whose activation functions are monotonic functions, the maximum sensitivity can be computed via solving convex optimization problems. Then, using a simulation-based method, the output reachable set estimation problem for neural networks is formulated into a chain of optimization problems. Finally, an automated safety verification is developed based on the output reachable set estimation result. An application to the safety verification for a robotic arm model with two joints is presented to show the effectiveness of proposed approaches.**

***Index Terms*—Multi-layer perceptron, reachable set estimation, simulation, verification.**

I. Introduction

Artificial neural networks have been widely used in machine learning systems. Applications include adaptive control [1]–[4], pattern recognition [5], [6], game playing [7], autonomous vehicles [8], and many others. Though neural networks have been showing the effectiveness and powerful ability in dealing with complex problems, they are confined to systems which comply only to the lowest safety integrity levels since, in most of the time, a neural network is viewed as a *black box* without effective methods to assure safety specifications for its outputs. Verifying neural networks is a hard problem, even simple properties about them have been proven NP-complete problems [9]. The difficulties mainly come from the presence of activation functions and the complex structures, making neural networks large-scale, nonlinear, non-convex and thus incomprehensible to humans. Until now, only few results have been reported for verifying neural networks. The verification for feed-forward multi-layer neural networks is investigated based on *Satisfiability Modulo Theory* (SMT) in [10], [11]. In [12] an Abstraction-Refinement approach is proposed. In [9], [13], a specific kind of activation functions called *Rectified Linear Unit* is considered for verification of neural networks. Additionally, some recent reachable set estimation results are reported for neural networks [14]–[16], these results that are based on Lyapunov functions analogous to stability [17]–[20] and reachability analysis of dynamical systems [21], [22], have potentials to be further extended to safety verification.

In this work, we shall focus on a class of neural networks called *Multi-Layer Perceptron* (MLP). Due to the complex structure, manual reasoning for an MLP is impossible. Inspired by some simulation-based ideas for verification [23]–[25], the information collected from a finitely many simulations will be exploited to estimate the output reachable set of an MLP and, furthermore, to do safety verification. To bridge the gap between the finitely many simulations and the output set generated from a bounded input set which essentially includes infinite number of inputs, a conception called maximum sensitivity is introduced to characterize the maximum deviation of the output subject to a bounded disturbance around a nominal input. By formulating a chain of optimizations, the maximum sensitivity for an MLP can be computed in a layer-by-layer manner. Then, an exhaustive search of the input set is enabled by a discretization of input space to achieve an estimation of output reachable set which consists of a union of reachtubes. Finally, by the merit of reachable set estimation, the safety verification for an MLP can be done via checking the existence of intersections between the estimated reachable set and unsafe regions. The main benefits of our approach are that there are very few restrictions on the activation functions except for the monotonicity which is satisfied by a variety of activation functions, and also no requirement on the bounded input sets. All these advantages are coming from the simulation-based nature of our approach.

The remainder of this paper is organized as follows. Preliminaries and problem formulation are given in Section II. The maximum sensitivity analysis for an MLP is studied in Section III. Output reachable set estimation and safety verification results are given in Section IV. An application to robotic arms is provided in Section V and Conclusions are presented in Section VI.

II. Preliminaries and Problem Formulation

A. Multi-Layer Neural Networks

A neural network consists of a number of interconnected neurons. Each neuron is a simple processing element that responds to the weighted inputs it received from other neurons. In this paper, we consider the most popular and general feed-forward neural networks called the Multi-Layer Perceptron (MLP). Generally, an MLP consists of three typical classes of layers: An input layer, that serves to pass the input vector to the network, hidden layers of computation neurons, and an

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract numbers FA8750-15-1-0105 and FA8650-12-3- 7255 via subcontract number WBSC 7255 SOI VU 0001, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246.

Authors are with the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37212 USA. Email: Weiming Xiang (xiangwming@gmail.com), Hoang-Dung Tran (trhoangdung@gmail.com), Taylor T. Johnson (taylor.johnson@gmail.com).

output layer composed of at least a computation neuron to produce the output vector. The action of a neuron depends on its activation function, which is described as

$$y_i = f\left(\sum\nolimits_{j=1}^{n} \omega_{ij} x_j + \theta_i\right) \tag{1}$$

where x_j is the jth input of the ith neuron, ω_{ij} is the weight from the jth input to the ith neuron, θ_i is called the bias of the ith neuron, y_i is the output of the ith neuron, $f(\cdot)$ is the activation function. The activation function is a nonlinear function describing the reaction of ith neuron with inputs $x_j(t),\ j = 1, \cdots, n$. Typical activation functions include rectified linear unit, logistic, tanh, exponential linear unit, linear functions, for instance. In this work, our approach aims at dealing with the most of activation functions regardless of their specific forms, only the following monotonic assumption needs to be satisfied.

Assumption 1: For any $x_1 \leq x_2$, the activation function satisfies $f(x_1) \leq f(x_2)$.

Remark 1: Assumption 1 is a common property that can be satisfied by a variety of activation functions. For example, it is easy to verify that the most commonly used logistic function $f(x) = 1/(1+e^{-x})$ satisfies Assumption 1.

An MLP has multiple layers, each layer ℓ, $1 \leq \ell \leq L$, has $n^{[\ell]}$ neurons. In particular, layer $\ell = 0$ is used to denote the input layer and $n^{[0]}$ stands for the number of inputs in the rest of this paper, and $n^{[L]}$ stands for the last layer, that is the output layer. For a neuron i, $1 \leq i \leq n^{[\ell]}$ in layer ℓ, the corresponding input vector is denoted by $\mathbf{x}^{[\ell]}$ and the weight matrix is $\mathbf{W}^{[\ell]} = [\boldsymbol{\omega}_1^{[\ell]}, \ldots, \boldsymbol{\omega}_{n^{[\ell]}}^{[\ell]}]^\top$, where $\boldsymbol{\omega}_i^{[\ell]}$ is the weight vector. The bias vector for layer ℓ is $\boldsymbol{\theta}^{[\ell]} = [\theta_1^{[\ell]}, \ldots, \theta_{n^{[\ell]}}^{[\ell]}]^\top$.

The output vector of layer ℓ can be expressed as

$$\mathbf{y}^{[\ell]} = f_\ell(\mathbf{W}^{[\ell]}\mathbf{x}^{[\ell]} + \boldsymbol{\theta}^{[\ell]})$$

where $f_\ell(\cdot)$ is the activation function for layer ℓ.

For an MLP, the output of $\ell - 1$ layer is the input of ℓ layer. The mapping from the input $\mathbf{x}^{[0]}$ of input layer to the output $\mathbf{y}^{[L]}$ of output layer stands for the input-output relation of the MLP, denoted by

$$\mathbf{y}^{[L]} = F(\mathbf{x}^{[0]}) \tag{2}$$

where $F(\cdot) \triangleq f_L \circ f_{L-1} \circ \cdots \circ f_1(\cdot)$.

According to the *Universal Approximation Theorem* [26], it guarantees that, in principle, such an MLP in the form of (2), namely the function $F(\cdot)$, is able to approximate any nonlinear real-valued function. Despite the impressive ability of approximating functions, much complexities represent in predicting the output behaviors of an MLP. In most of real applications, an MLP is usually viewed as a *black box* to generate a desirable output with respect to a given input. However, regarding property verifications such as safety verification, it has been observed that even a well-trained neural network can react in unexpected and incorrect ways to even slight perturbations of their inputs, which could result in unsafe systems. Thus, the output reachable set estimation of an MLP, which is able to cover all possible values of outputs, is necessary for the safety verification of an MLP and draw a safe or unsafe conclusion for an MLP.

B. Problem Formulation

Given an input set $\mathcal{X}$, the output reachable set of neural network (2) is stated by the following definition.

Definition 1: Given an MLP in the form of (2) and an input set $\mathcal{X}$, the output reachable set of (2) is defined as

$$\mathcal{Y} \triangleq \{\mathbf{y}^{[L]} \mid \mathbf{y}^{[L]} = F(\mathbf{x}^{[0]}),\ \mathbf{x}^{[0]} \in \mathcal{X}\}. \tag{3}$$

Since MLPs are often large, nonlinear, and non-convex, it is extremely difficult to compute the exact output reachable set $\mathcal{Y}$ for an MLP. Rather than directly computing the exact output reachable set for an MLP, a more practical and feasible way is to derive an over-approximation of $\mathcal{Y}$, which is called output reachable set estimation.

Definition 2: A set $\tilde{\mathcal{Y}}$ is called an output reachable set estimation of MLP (2), if $\mathcal{Y} \subseteq \tilde{\mathcal{Y}}$ holds, where $\mathcal{Y}$ is the output reachable set of MLP (2).

Based on Definition 2, the problem of output reachable set estimation for an MLP is given as below.

Problem 1: Given a bounded input set $\mathcal{X}$ and an MLP described by (2), how to find a set $\tilde{\mathcal{Y}}$ such that $\mathcal{Y} \subseteq \tilde{\mathcal{Y}}$, and make the estimation set $\tilde{\mathcal{Y}}$ as small as possible[1]?

In this work, we will focus on the safety verification for neural networks. The safety specification for outputs is expressed by a set defined in the output space, describing the safety requirement.

Definition 3: Safety specification $\mathcal{S}$ of an MLP formalizes the safety requirements for output $\mathbf{y}^{[L]}$ of MLP $\mathbf{y}^{[L]} = F(\mathbf{x}^{[0]})$, and is a predicate over output $\mathbf{y}^{[L]}$ of MLP. The MLP is safe if and only if the following condition is satisfied:

$$\mathcal{Y} \cap \neg\mathcal{S} = \emptyset \tag{4}$$

where $\neg$ is the symbol for logical negation.

Therefore, the safety verification problem for an MLP is stated as follows.

Problem 2: Given a bounded input set $\mathcal{X}$, an MLP in the form of (2) and a safety specification $\mathcal{S}$, how to check if condition (4) is satisfied?

Before ending this section, a lemma is presented to show that the safety verification of an MLP can be relaxed by checking with the over-approximation of the output reachable set.

Lemma 1: Consider an MLP in the form of (2), an output reachable set estimation $\mathcal{Y} \subseteq \tilde{\mathcal{Y}}$ and a safety specification $\mathcal{S}$, the MLP is safe if the following condition is satisfied

$$\tilde{\mathcal{Y}} \cap \neg\mathcal{S} = \emptyset. \tag{5}$$

Proof: Since $\mathcal{Y} \subseteq \tilde{\mathcal{Y}}$, (5) directly leads to $\mathcal{Y} \cap \neg\mathcal{S} = \emptyset$. The proof is complete. ■

Lemma 1 implies that it is sufficient to use the estimated output reachable set for the safety verification of an MLP, thus the solution of Problem 1 is also the key to solve Problem 2.

[1]For a set $\mathcal{Y}$, its over-approximation $\tilde{\mathcal{Y}}_1$ is smaller than another over-approximation $\tilde{\mathcal{Y}}_2$ if $d_H(\tilde{\mathcal{Y}}_1, \mathcal{Y}) < d_H(\tilde{\mathcal{Y}}_2, \mathcal{Y})$ holds, where d_H stands for the Hausdorff distance.

III. Maximum Sensitivity for Neural Networks

Due to the complex structure and nonlinearities in activation functions, estimating the output reachable sets of MLPs represents much difficulties if only using analytical methods. One possible way to circumvent those difficulties is to employ the information produced by a finite number of simulations. As well known, the finitely many simulations generated from input set $\mathcal{X}$ are incomplete to characterize output set $\mathcal{Y}$, a conception called maximum sensitivity is introduced to bridge the gap between simulations and output reachable set estimations of MLPs.

Definition 4: Given an MLP $\mathbf{y}^{[L]} = F(\mathbf{x}^{[0]})$, an input $\mathbf{x}^{[0]}$ and disturbances $\Delta\mathbf{x}^{[0]}$ satisfying $\left\|\Delta\mathbf{x}^{[0]}\right\| \leq \delta$, the maximum sensitivity of the MLP with input error δ at $\mathbf{x}^{[0]}$ is defined by

$$\epsilon_F(\mathbf{x}^{[0]}, \delta) \triangleq \inf\{\epsilon : \left\|\Delta\mathbf{y}^{[L]}\right\| \leq \epsilon, \text{ where } \mathbf{y}^{[L]} = F(\mathbf{x}^{[0]}) \text{ and } \left\|\Delta\mathbf{x}^{[0]}\right\| \leq \delta\} \quad (6)$$

Remark 2: In some previous articles as [27], [28], the sensitivity for neural networks is defined as the mathematical expectation of output deviations due to input and weight deviations with respect to overall input and weight values in a given continuous interval. The sensitivity in the average point of view works well for learning algorithm improvement [29], weight selection [30], architecture construction [31], for instance. However, it cannot be used for safety verification due to the concern of soundness. In this paper, the maximum sensitivity is introduced to measure the maximum deviation of outputs, which is caused by the bounded disturbances around the nominal input $\mathbf{x}^{[0]}$.

Due to the multiple layer structure, we are going to develop a layer-by-layer method to compute the maximum sensitivity defined by (6). First, we consider a single layer ℓ. According to Definition 4, the maximum sensitivity for layer ℓ, which is denoted by $\epsilon(\mathbf{x}^{[\ell]}, \delta^{[\ell]})$ at $\mathbf{x}^{[\ell]}$, can be computed by

$$\begin{aligned} &\max\ \epsilon(\mathbf{x}^{[\ell]}, \delta^{[\ell]}) \\ &\text{s.t. } \epsilon(\mathbf{x}^{[\ell]}, \delta^{[\ell]}) = \left\| f_\ell(\mathbf{W}^{[\ell]}(\mathbf{x}^{[\ell]} + \Delta\mathbf{x}^{[\ell]}) + \boldsymbol{\theta}^{[\ell]}) - \mathbf{y}^{[\ell]} \right\| \\ &\quad \mathbf{y}^{[\ell]} = f_\ell(\mathbf{W}^{[\ell]}\mathbf{x}^{[\ell]} + \boldsymbol{\theta}^{[\ell]}) \\ &\quad \left\|\Delta\mathbf{x}^{[\ell]}\right\| \leq \delta^{[\ell]}. \end{aligned} \quad (7)$$

In the rest of paper, the norm $\|\cdot\|$ is considered the infinity norm, that is $\|\cdot\|_\infty$. By the definition of $\|\cdot\|_\infty$ and monotonicity assumption in Assumption 1, the optimal solution $\Delta\mathbf{x}^{[\ell]}_{\text{opt}}$ of (7) can be found by running the following set of optimization problems.

To find the optimal solution of (7) for layer ℓ, we start from the neuron i in layer ℓ, the following two convex optimizations can be set up

$$\begin{aligned} &\max\ \beta_i^{[\ell]} \\ &\text{s.t. } \beta_i^{[\ell]} = (\boldsymbol{\omega}_i^{[\ell]})^\top(\mathbf{x}^{[\ell]} + \Delta\mathbf{x}^{[\ell]}) + \theta^{[\ell]} \\ &\quad \left\|\Delta\mathbf{x}^{[\ell]}\right\|_\infty \leq \delta^{[\ell]} \end{aligned} \quad (8)$$

and

$$\begin{aligned} &\min\ \beta_i^{[\ell]} \\ &\text{s.t. } \beta_i^{[\ell]} = (\boldsymbol{\omega}_i^{[\ell]})^\top(\mathbf{x}^{[\ell]} + \Delta\mathbf{x}^{[\ell]}) + \theta^{[\ell]} \\ &\quad \left\|\Delta\mathbf{x}^{[\ell]}\right\|_\infty \leq \delta^{[\ell]}. \end{aligned} \quad (9)$$

Then, due to the monotonicity, the following optimization problem defined over a finite set consisting of $\beta^{[\ell]}_{i,\max}$ and $\beta^{[\ell]}_{i,\min}$ obtained in (8) and (9) is formulated to compute the maximum absolute value of output of neuron i in layer ℓ

$$\begin{aligned} &\max\ \gamma_i^{[\ell]} \\ &\text{s.t. } \gamma_i^{[\ell]} = \left| f_\ell(\beta_i^{[\ell]}) - f_\ell((\boldsymbol{\omega}_i^{[\ell]})^\top(\mathbf{x}^{[\ell]}) + \theta^{[\ell]}) \right| \\ &\quad \beta_i^{[\ell]} \in \{\beta^{[\ell]}_{i,\min},\ \beta^{[\ell]}_{i,\max}\}. \end{aligned} \quad (10)$$

Finally, based on the maximum absolute value of the output of neuron i and because of the definition of infinity norm, we are ready to compute the maximum sensitivity of layer ℓ by picking out the largest value of $\gamma_i^{[\ell]}$ in layer ℓ, that is

$$\begin{aligned} &\max\ \epsilon(\mathbf{x}^{[\ell]}, \delta^{[\ell]}) \\ &\text{s.t. } \epsilon(\mathbf{x}^{[\ell]}, \delta^{[\ell]}) \in \{\gamma_1^{[\ell]}, \ldots, \gamma_{n^{[\ell]}}^{[\ell]}\}. \end{aligned} \quad (11)$$

In summary, the maximum sensitivity of a single layer ℓ can be computed through solving optimizations (8)–(11) sequentially.

Proposition 1: Given a single layer ℓ, the maximum sensitivity $\epsilon(\mathbf{x}^{[\ell]}, \delta^{[\ell]})$ is the solution of (11) in which $\gamma_1^{[\ell]}, \ldots, \gamma_{n^{[\ell]}}^{[\ell]}$ are solutions of (10) with $\beta^{[\ell]}_{i,\min},\ \beta^{[\ell]}_{i,\max}$ being solutions of (8) and (9).

The above optimizations (8)–(11) provide a way to compute the maximum sensitivity for one layer. Then, for an MLP, we have $\mathbf{x}^{[\ell]} = \mathbf{y}^{[\ell-1]}$, $\ell = 1, \ldots, L$, so the output of each layer can be computed by iterating above optimizations with updated input $\mathbf{x}^{[\ell]} = \mathbf{y}^{[\ell-1]}$, $\delta^{[\ell]} = \epsilon(\mathbf{x}^{[\ell-1]}, \delta^{[\ell-1]})$, $\ell = 1, \ldots, L$. The maximum sensitivity of neural network $\mathbf{y}^{[L]} = F(\mathbf{x}^{[0]})$ is the outcome of optimization (11) for output layer L, namely, $\epsilon_F(\mathbf{x}^{[0]}, \delta) = \epsilon(\mathbf{x}^{[L]}, \delta^{[L]})$. The layer-by-layer idea is illustrated in Fig. 1, which shows the general idea of the computation process for multiple layer neural networks.

In conclusion, the computation for the maximal sensitivity of an MLP is converted to a chain of optimization problems. Furthermore, the optimization problems (8), (9) are convex optimization problems which can be efficiently solved by existing tools such as `cvx`, `linprog` in Matlab. To be more efficient in computation without evaluating the objective function repeatedly, we can even pre-generate the expression of optimal solutions given the weight and bias of the neural network. Optimizations (10), (11) only have finite elements to search for the optimum, which can be also computed efficiently. The algorithm for computing the maximum sensitivity of an MLP is given in Algorithm 1.

IV. Reachable Set Estimation and Verification

In previous section, the maximum sensitivity for an MLP can be computed via a chain of optimizations. The computation result actually can be viewed as a *reachtube* for the inputs

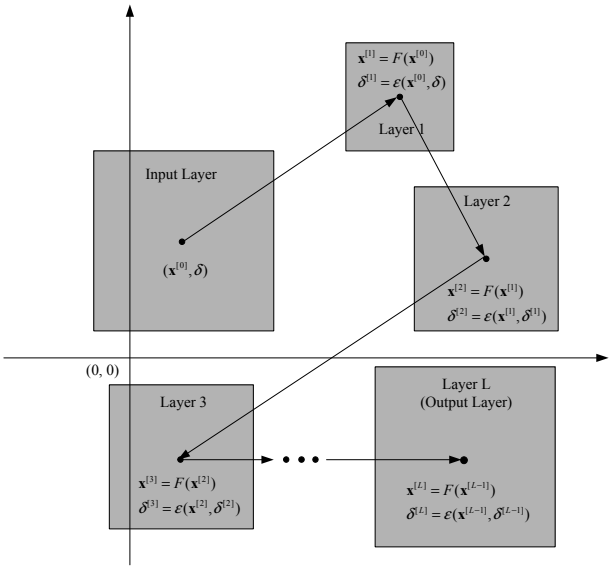

Fig. 1. Illustration for computing maximum sensitivity for an MLP.

Algorithm 1 Maximum Sensitivity Computation Function for MLP

Require: MLP F, input $\mathbf{x}^{[0]}$ and disturbance error δ.
Ensure: Maximum Sensitivity $\epsilon_F(\mathbf{x}^{[0]}, \delta)$.

function MAXSENSITIVITY(F, $\mathbf{x}^{[0]}$, δ)
  $\mathbf{x}^{[1]} \leftarrow \mathbf{x}^{[0]}$; $\delta^{[1]} \leftarrow \delta$
  for $\ell = 1:1:L-1$ **do**
    Solve (8), (9) to obtain $\beta^{[\ell]}_{i,\min}$, $\beta^{[\ell]}_{i,\max}$
    With $\beta^{[\ell]}_{i,\min}$, $\beta^{[\ell]}_{i,\max}$, solve (10) to obtain $\gamma^{[\ell]}_i$
    With $\gamma^{[\ell]}_i$, solve (11) to obtain $\epsilon(\mathbf{x}^{[\ell]}, \delta^{[\ell]})$
    $\mathbf{x}^{[\ell+1]} \leftarrow f_\ell(\mathbf{W}^{[\ell]}\mathbf{x}^{[\ell]} + \boldsymbol{\theta}^{[\ell]})$; $\delta^{[\ell+1]} \leftarrow \epsilon(\mathbf{x}^{[\ell]}, \delta^{[\ell]})$
  end for
  With $\mathbf{x}^{[L]}$, $\delta^{[L]}$, solve (8)–(11) to obtain $\epsilon(\mathbf{x}^{[L]}, \delta^{[L]})$
  $\epsilon_F(\mathbf{x}^{[0]}, \delta) \leftarrow \epsilon(\mathbf{x}^{[L]}, \delta^{[L]})$
  return $\epsilon_F(\mathbf{x}^{[0]}, \delta)$
end function

around nominal input $\mathbf{x}^{[0]}$, that are the inputs bounded in the tube $\left\|\Delta\mathbf{x}^{[0]}\right\|_\infty \leq \delta$. This allows us to relate the individual simulation outputs to the output reachable set of an MLP.

First, the input space is discretized into lattices, which are described by

$$\mathcal{L}_i \triangleq \{\mathbf{x}^{[0]} \mid \left\|\mathbf{x}^{[0]} - \mathbf{x}_i^{[0]}\right\|_\infty \leq \delta\} \tag{12}$$

where $\mathbf{x}_i^{[0]}$ and δ are called the center and the radius of $\mathcal{L}_i$, respectively. The sets $\mathcal{L}_i$ satisfy $\mathcal{L}_i \cap \mathcal{L}_j = \{\mathbf{x}^{[0]} \mid \left\|\mathbf{x}^{[0]} - \mathbf{x}_i^{[0]}\right\|_\infty = \delta \wedge \left\|\mathbf{x}^{[0]} - \mathbf{x}_j^{[0]}\right\|_\infty = \delta\}$ and $\bigcup_{i=1}^{\infty} \mathcal{L}_i = \mathbb{R}^{n\times n}$. Obviously, for any bounded set $\mathcal{X}$, we can find a finite number of $\mathcal{L}_i$ such that $\mathcal{L}_i \bigcap \mathcal{X} \neq \emptyset$. The index set for all $\mathcal{L}_i$ satisfying $\mathcal{L}_i \cap \mathcal{X} \neq \emptyset$ is denoted by $\mathcal{I}$, so it can be obtained that $\mathcal{X} \subseteq \bigcup_{i\in\mathcal{I}} \mathcal{L}_i$. Explicitly, the lattices with a smaller radius δ are able to achieve a preciser approximation of bounded set $\mathcal{X}$ and, moreover, $\bigcup_{i\in\mathcal{I}} \mathcal{L}_i$ will exactly be $\mathcal{X}$ if radius $\delta \to 0$. The number of lattices is closely related to the dimension of the input space and radius chosen for discretization. Taking a unit box $\{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| \leq 1\}$ for example, the number of lattices with radius δ is $\lceil 1/2\delta \rceil^n$.

The first step is to derive all the lattices $\mathcal{L}_i$, $i \in \mathcal{I}$ for the input set $\mathcal{X}$ such that $\mathcal{L}_i \cap \mathcal{X} \neq \emptyset$, $\forall i \in \mathcal{I}$. Then, based on the maximum sensitivity computation result, the output reachtube for each lattice $\mathcal{L}_i$ can be obtained by using Algorithm 1. Since $\mathcal{X} \subseteq \bigcup_{i\in\mathcal{I}} \mathcal{L}_i$, the union of output reachtubes of $\mathcal{L}_i$, $i \in \mathcal{I}$ includes all the possible outputs generated by the neural network with input set $\mathcal{X}$. The following proposition is the main result in this work.

Proposition 2: Given an MLP $\mathbf{y}^{[L]} = F(\mathbf{x}^{[0]})$, input set $\mathcal{X}$ and lattices $\mathcal{L}_i$, $i \in \mathcal{I}$ with centers $\mathbf{x}_i^{[0]}$ and radius δ, and all the lattices satisfy $\mathcal{L}_i \cap \mathcal{X} \neq \emptyset$, $\forall i \in \mathcal{I}$, the output reachable set $\mathcal{Y}$ satisfies $\mathcal{Y} \subseteq \tilde{\mathcal{Y}} \triangleq \bigcup_{i\in\mathcal{I}} \tilde{\mathcal{Y}}_i$, where

$$\tilde{\mathcal{Y}}_i \triangleq \{\mathbf{y}^{[L]} \mid \left\|\mathbf{y}^{[L]} - \mathbf{y}_i^{[L]}\right\|_\infty \leq \epsilon_F(\mathbf{x}_i^{[0]}, \delta),\ \mathbf{y}_i^{[L]} = F(\mathbf{x}_i^{[0]})\} \tag{13}$$

where $\epsilon_F(\mathbf{x}_i^{[0]}, \delta)$ is computed by Algorithm 1.

Proof: Using Algorithm 1 for inputs within lattice $\mathcal{L}_i$, $\tilde{\mathcal{Y}}_i$ is the reachtube for $\mathcal{L}_i$ via the given MLP. Thus, the union of $\tilde{\mathcal{Y}}_i$, that is $\bigcup_{i\in\mathcal{I}} \tilde{\mathcal{Y}}_i$, is the output reachable set of $\bigcup_{i\in\mathcal{I}} \mathcal{L}_i$. Moreover, due to $\mathcal{X} \subseteq \bigcup_{i\in\mathcal{I}} \mathcal{L}_i$, it directly implies that the output reachable set of $\mathcal{X}$ is a subset of $\bigcup_{i\in\mathcal{I}} \tilde{\mathcal{Y}}_i$, that is $\mathcal{Y} \subseteq \tilde{\mathcal{Y}} \triangleq \bigcup_{i\in\mathcal{I}} \tilde{\mathcal{Y}}_i$. The proof is complete. ■

Based on Proposition 2, the output reachable set estimation involves the following two key steps:

(1) Execute a finite number of simulations for MLP to get individual outputs $\mathbf{y}_i^{[L]}$ with respect to individual inputs $\mathbf{x}_i^{[0]}$. This can be done by simply generating the outputs with a finite number of inputs through the MLP as $\mathbf{y}_i^{[L]} = F(\mathbf{x}_i^{[0]})$. That is the main reason that our approach is called simulation-based.
(2) Compute the maximum sensitivity for a finite number of lattices centered at $\mathbf{x}_i^{[0]}$, which can be solved by the `MaxSensitivity` function proposed in Algorithm 1. This step is to produce the reachtubes based on the simulation results, and combine them for the reachable set estimation of outputs.

The complete algorithm to perform the output reachable set estimation for an MLP is summarized in Algorithm 2, and Example 1 is provided to validate our approach.

Algorithm 2 Output Reachable Set Estimation for MLP

Require: MLF F, input set $\mathcal{X}$.
Ensure: Estimated output set $\tilde{\mathcal{Y}}$.

function OUTPUTREACH(F, $\mathcal{X}$)
  Initialize $\mathcal{L}_i$, $i \in \mathcal{I}$, $\mathbf{x}_i^{[0]}$, δ; $\tilde{\mathcal{Y}} \leftarrow \emptyset$
  for $\ell = 1:1:|\mathcal{I}|$ **do**
    $\mathbf{y}_i^{[L]} \leftarrow F(\mathbf{x}_i^{[0]})$
    $\epsilon_F(\mathbf{x}_i^{[0]}, \delta) \leftarrow$ MAXSENSITIVITY$(F, \mathbf{x}_i^{[0]}, \delta)$
    $\tilde{\mathcal{Y}}_i \leftarrow \{\mathbf{y}^{[L]} \mid \left\|\mathbf{y}^{[L]} - \mathbf{y}_i^{[L]}\right\|_\infty \leq \epsilon_F(\mathbf{x}_i^{[0]}, \delta)\}$
    $\tilde{\mathcal{Y}} \leftarrow \tilde{\mathcal{Y}} \cup \tilde{\mathcal{Y}}_i$
  end for
  return $\tilde{\mathcal{Y}}$
end function

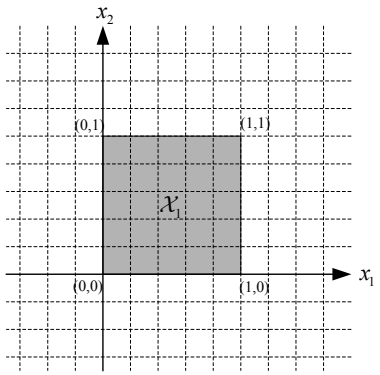

Fig. 2. Input sets $\mathcal{X}_1$ and 25 lattices with radius of $\delta = 0.1$.

Example 1: A neural network with 2 inputs, 2 outputs and 1 hidden layer consisting of 5 neurons is considered. The activation function for the hidden layer is choosen as `tanh` function and `purelin` function is for the output layer. The weight matrices and bias vectors are randomly generated as below:

$$\mathbf{W}^{[1]} = \begin{bmatrix} -0.9507 & -0.7680 \\ 0.9707 & 0.0270 \\ -0.6876 & -0.0626 \\ 0.4301 & 0.1724 \\ 0.7408 & -0.7948 \end{bmatrix}, \ \boldsymbol{\theta}^{[1]} = \begin{bmatrix} 1.1836 \\ -0.9087 \\ -0.3463 \\ 0.2626 \\ -0.6768 \end{bmatrix}$$

$$\mathbf{W}^{[2]} = \begin{bmatrix} 0.8280 & 0.6839 & 1.0645 & -0.0302 & 1.7372 \\ 1.4436 & 0.0824 & 0.8721 & 0.1490 & -1.9154 \end{bmatrix}$$

$$\boldsymbol{\theta}^{[2]} = \begin{bmatrix} -1.4048 \\ -0.4827 \end{bmatrix}.$$

The input set is considered as $\mathcal{X}_1 = \{[x_1 \ x_2]^\top \mid |x_1 - 0.5| \leq 0.5 \wedge |x_2 - 0.5| \leq 0.5\}$. In order to execute function `OutputReach` described in Algorithm 2, the first step is to initialize the lattices $\mathcal{L}_i$ with centers $\mathbf{x}_i^{[0]}$ and radius δ. In this example, the radius is chosen to be 0.1 and 25 lattices are generated as in Fig. 2 shown in gray, which means there are in total 25 simulations to be executed for the output reachable set estimation.

Executing function `OutputReach` for $\mathcal{X}_1$, the estimated output reachable set is given in Fig. 3, in which 25 reachtubes are obtained and the union of them is an over-approximation of reachable set $\mathcal{Y}$. To validate the result, 10000 random outputs are generated, it is clear to see that all the outputs are included in the estimated reachable set, showing the effectiveness of the proposed approach.

Moreover, we choose different radius for discretizing state space to show how the choice of radius affects the estimation outcome. As mentioned before, a smaller radius implies a tighter approximation of input sets and is supposed to achieve a preciser estimation. Here, we select the radius as $\delta \in \{0.1, 0.05, 0.025, 0.0125\}$. With finer discretizations, more simulations are required for running function `OutputReach`, but tighter estimations for the output reachable set can be obtained. The output reachable set estimations are shown in Fig. 4. Comparing those results, it can be observed that a smaller radius can lead to a better estimation result at the expense of more simulations and computation time, as shown in Table I.

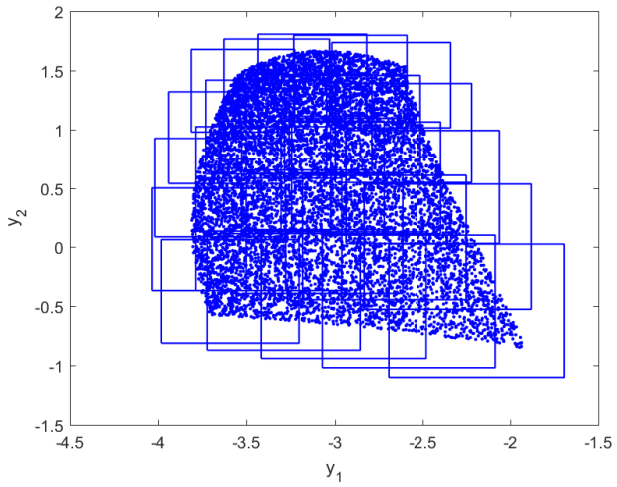

Fig. 3. Output reachable set estimation with input set $\mathcal{X}_1$ and $\delta = 0.1$. 25 reachtubes are computed and 10000 random outputs are all included in the estimated reachable set.

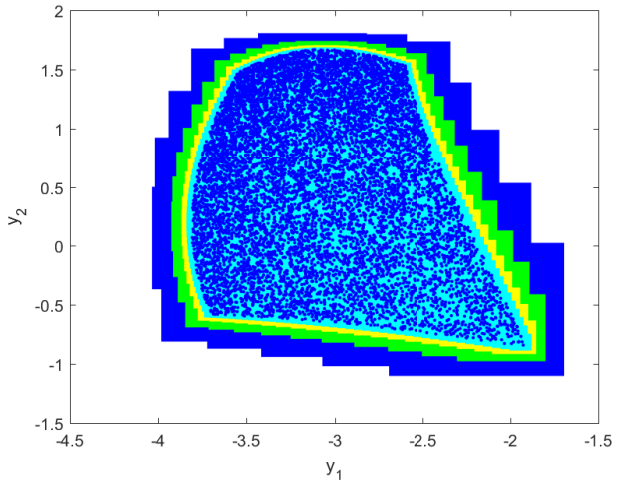

Fig. 4. Output reachable set estimations with input set $\mathcal{X}_1$ and $\delta = 0.1(\text{blue}), 0.05(\text{green}), 0.025(\text{yellow}), 0.0125(\text{cyan})$. Tighter estimations are obtained with smaller radius.

TABLE I
COMPARISON OF OUTPUT REACHABLE SET ESTIMATIONS WITH DIFFERENT RADIUS

Radius	0.1	0.05	0.025	0.0125
Computation time	0.044 s	0.053 s	0.086 s	0.251 s
Simulations	25	100	400	1600

Algorithm 2 is sufficient to solve the output reachable set estimation problem for an MLP, that is Problem 1. Then, we can move forward to Problem 2, the safety verification problem for an MLP with a given safety specification $\mathcal{S}$.

Proposition 3: Consider an MLP in the form of (2), an output reachable set estimation and a safety specification $\mathcal{S}$, the MLP is safe if $\tilde{\mathcal{Y}} \cap \neg\mathcal{S} = \emptyset$, where $\tilde{\mathcal{Y}}$ is the estimated output reachable set obtained by Algorithm 2.

Proof: By Algorithm 2, we have $\mathcal{Y} \subseteq \tilde{\mathcal{Y}}$, where $\mathcal{Y}$ is the

actual output reachable set of the MLP. Using Lemma 1, the safety can be guaranteed. The proof is complete. ■

The simulation-based safety verification algorithm is presented in Algorithm 3.

Algorithm 3 Safety Verification for MLP

Require: MLP F, input set $\mathcal{X}$, safety requirement $\mathcal{S}$.
Ensure: Safe or unsafe property.

function SAFETYVERI(F, $\mathcal{X}$, $\mathcal{S}$)
  Initialize $\mathcal{L}_i$, $i \in \mathcal{I}$, $\mathbf{x}_i^{[0]}$, δ
  for $\ell = 1 : 1 : |\mathcal{I}|$ **do**
    $\mathbf{y}_i^{[L]} \leftarrow F(\mathbf{x}_i^{[0]})$
    if $\mathbf{y}_i^{[L]} \cap \neg\mathcal{S} \neq \emptyset$ and $\mathbf{x}_i^{[0]} \in \mathcal{X}$ **then**
      return UNSAFE
    end if
  end for
  $\tilde{\mathcal{Y}} \leftarrow$ OUTPUTREACH(F, $\mathcal{X}$)
  if $\tilde{\mathcal{Y}} \cap \neg\mathcal{S} = \emptyset$ **then**
    return SAFE
  else
    return UNCERTAIN
  end if
end function

Remark 3: The Algorithm 3 is sound for the cases of SAFE and UNSAFE, that is, if it returns SAFE then the system is safe; when it returns UNSAFE there exists at least one output from input set is unsafe since the existence of one simulation that is unsafe is sufficient to claim unsafeness. Additionally, if it returns UNCERTAIN, caused by the fact $\tilde{\mathcal{Y}} \cap \neg\mathcal{S} \neq \emptyset$, that means the safety property is unclear for this case.

Example 2: The same MLP as in Example 1 is considered, and the input set is considered to be $\mathcal{X}_2 = \{[x_1\ x_2]^\top \mid |x_1 - 0.5| \leq 1.5 \wedge |x_2 - 0.5| \leq 0.1\}$. Furthermore, the safe specification $\mathcal{S}$ is assumed as $\mathcal{S} = \{[x_1\ x_2]^\top \mid -3.7 \leq x_1 \leq -1.5\}$. To do safety verification, the first step of using function `SafetyVeri` in Algorithm 3 is to initialize the lattices $\mathcal{L}_i$ with two radius $\delta_1 = 0.1$ and $\delta_2 = 0.05$. Since two radius are used, the verification results could be different due to different precisions selected. The verification results are shown in Figs. 5 and 6, and compared in Table II.

TABLE II
COMPARISON OF SAFETY VERIFICATIONS WITH DIFFERENT RADIUS

Radius	$\delta_1 = 0.1$	$\delta_2 = 0.05$
Safety	UNCERTAIN	SAFE
Simulations	15	60

The safety property is uncertain when the radius is chosen as 0.1. However, we can conclude the safeness of the MLP when a smaller radius $\delta = 0.05$ is chosen at the expense of increasing the number of simulations from 15 to 60.

V. APPLICATION IN SAFETY VERIFICATION FOR ROBOTIC ARM MODELS

In this section, our study focuses on *learning forward kinematics* of a robotic arm model with two joints, see Fig. 7. The learning task of the MLP is to predict the position

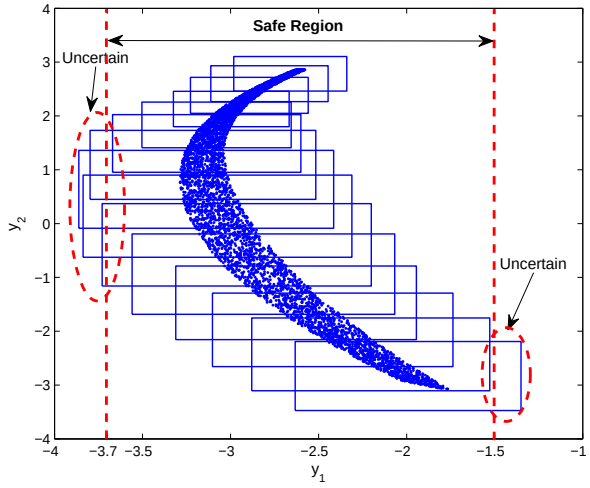

Fig. 5. Safety verification for input belonging to $\mathcal{X}_2$. With radius $\delta = 0.1$, the MLP cannot be concluded to be safe or not, since there exist intersections between the estimated reachable set and the unsafe region.

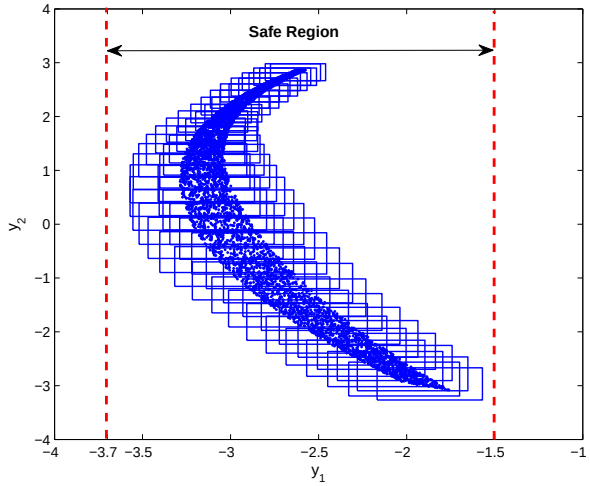

Fig. 6. Safety verification for input belonging to $\mathcal{X}_2$. With $\delta = 0.05$, the safety can be confirmed, since the estimated reachable set is in the safe region.

(x, y) of the end with knowing the joint angles (θ_1, θ_2). For the robotic arm, the input space $[0, 2\pi] \times [0, 2\pi]$ for (θ_1, θ_2) is classified into three zones for its operations:

(1) *Normal working zone:* The normal working zone is the working region that the robotic arm works most of the time in, and the input-output training data is all collected from this region to train the MLP model. This zone is assumed to be $\theta_1, \theta_2 \in [\frac{5\pi}{12}, \frac{7\pi}{12}]$.
(2) *Forbidden zone:* The forbidden zone specifies the region that the robotic arm will never operate in due to physical constraints or safety considerations in design. This zone is assumed as $\theta_1, \theta_2 \in [0, \frac{\pi}{3}] \cup [\frac{2\pi}{3}, 2\pi]$.
(3) *Buffering zone:* The buffering zone is between the normal working zone and the forbidden zone. Some occasional operations may occur out of normal working zone, but it remains in the buffering zone, not reaching the

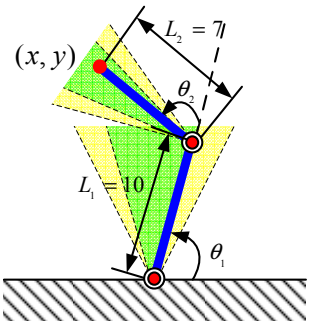

Fig. 7. Robotic arm with two joints. The normal working zone of (θ_1, θ_2) is colored in green and the buffering zone is in yellow.

forbidden zone. This zone is $\theta_1, \theta_2 \in [\frac{\pi}{3}, \frac{5\pi}{12}] \cup [\frac{7\pi}{12}, \frac{2\pi}{3}]$.

The safety specification for the position (x, y) is considered as $\mathcal{S} = \{(x, y) \mid -14 \leq x \leq 3 \wedge 1 \leq y \leq 17\}$. In the safety point of view, the MLP needs to be verified that all the outputs produced by the inputs in the normal working zone and buffering zone will satisfy safety specification $\mathcal{S}$. One point needs to emphasize is that the MLP is trained by the data in normal working zone, but the safety specification is defined on both normal working zone and buffering zone.

Using the data from normal working zone, the learning process is standard by using `train` function in the neural network toolbox in Matlab. The MLP considered for this example is with 2 inputs, 2 outputs and 1 hidden layer consisting of 5 neurons. The activation functions `tanh` and `purelin` are for hidden layer and output layer, respectively. However, for the trained MLP, there is no safety assurance for any manipulations, especially for the ones in the buffering zone where no input-output data is used to train the MLP. To verify the safety specification of the trained MLP, our function `SafetyVeri` presented in Algorithm 3 is used for this example.

First, we train the MLP with inputs $(\theta_1, \theta_2) \in [\frac{5\pi}{12}, \frac{7\pi}{12}] \times [\frac{5\pi}{12}, \frac{7\pi}{12}]$ along with their corresponding outputs. Then, to use function `SafetyVeri` for the inputs in both normal working zone and buffering zone, we discretize input space $[\frac{\pi}{3}, \frac{2\pi}{3}] \times [\frac{\pi}{3}, \frac{2\pi}{3}]$ with radius $\delta = 0.05$. The safety verification result is shown in Fig. 8. It can be observed that the safety property of the MLP is uncertain since the estimated reachable set reaches out of the safe region $\mathcal{S}$. Then, 5000 random simulations are executed, and it shows that no output is unsafe. However, 5000 simulations or any finite number of simulations are not sufficient to say the MLP is safe. Therefore, to soundly claim that the MLP trained with the data collected in normal working zone is safe with regard to both normal working and buffering zones, a smaller radius $\delta = 0.02$ has to be adopted. The verification result with $\delta = 0.02$ is shown in Fig. 9. It can be seen that the reachable set of the MLP is contained in the safe region, which is sufficient to claim the safety of the robotic arm MLP model.

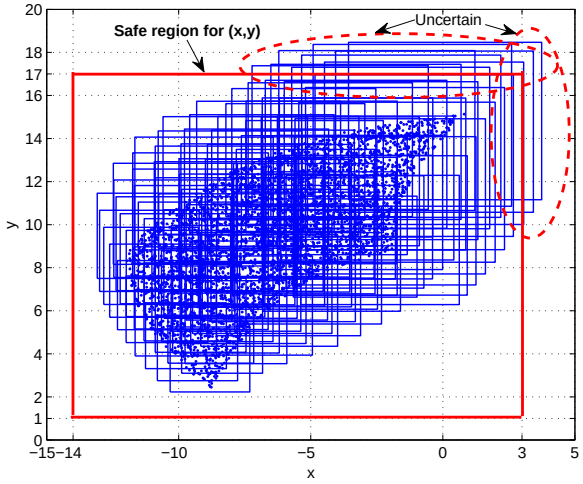

Fig. 8. Safety verification for MLP model of robotic arm with two joints. With radius $\delta = 0.05$, the safety cannot be determined.

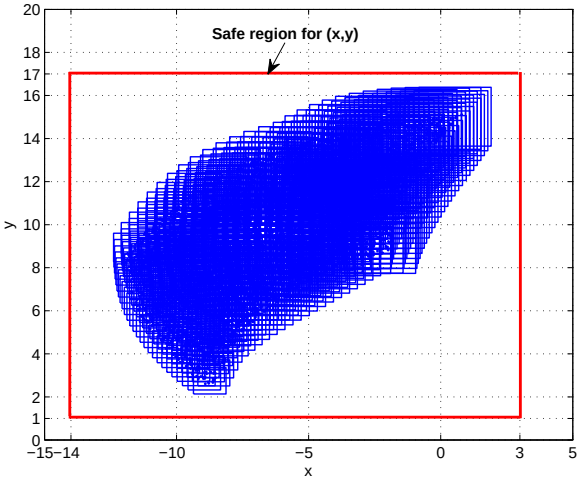

Fig. 9. Safety verification for MLP model of robotic arm with two joints. With radius $\delta = 0.02$, the MLP model can be claimed to be safe.

VI. Related Work

In [9], an SMT solver named Reluplex is proposed for a special class of neural networks with ReLU activation functions. The Reluplex extends the well-known Simplex algorithm from linear functions to ReLU functions by making use of the piecewise linear feature of ReLU functions. In contrast to Reluplex solver in the context of SMT, the approach developed in our paper aims at the reachability problem of neural networks. In [13], A layer-by-layer approach is developed for the output reachable set computation of ReLU neural networks. The computation is formulated in the form of a set of manipulations for a union of polytopes. It should be noted that our approach is general in the sense that it is not tailored for a specific activation function.

The authors of [11] and [12] propose an approach for verifying properties of neural networks with sigmoid activation functions. They replace the activation functions with piecewise linear approximations thereof, and then invoke black-box SMT solvers. Our approach does not use any approximations of

activation functions. Instead, the approximation of our approach comes from the over-approximation of output set of each neuron, by lower and upper bounds.

In a recent paper [10], the authors propose a method for proving the local adversarial robustness of neural networks. The purpose of paper [10] is to check the robustness around one fixed point. Instead, our approach is focusing on a set defined on a continuous domain, rather than one single point.

Finally, Lyapunov function approach plays a key role in stability and reachability analysis for dynamical systems such as uncertain systems [18], positive systems [19], hybrid systems [20]–[22]. The neural networks involved in papers [14]–[16] are recurrent neural networks modeled by a family of differential equations so that Lyapunov function approach works. For the MLP considered in this paper which is described by a set of nonlinear algebraic equations, Lyapunov function approach is not a suitable tool. Thus, we introduce another conception called maximal sensitivity to characterize the reachability property of neural networks.

VII. Conclusions

This paper represents a simulation-based method to compute the output reachable sets for MLPs by solving a chain of optimization problems. Based on the monotonic assumption which can be satisfied by a variety of activation functions of neural networks, the computation for the so-called maximum sensitivity is formulated as a set of optimization problems essentially described as convex optimizations. Then, utilizing the results for maximum sensitivity, the reachable set estimation of an MLP can be performed by checking the maximum sensitivity property of finite number of sampled inputs to the MLP. Finally, the safety property of an MLP can be verified based on the estimation of output reachable set. Numerical examples are provided to validate the proposed algorithms. Finally, an application of safety verification for a robotic arm model with two joints is presented.

References

[1] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop, "Neural networks for control systems: A survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[2] S. S. Ge, C. C. Hang, and T. Zhang, "Adaptive neural network control of nonlinear systems by state and output feedback," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 6, pp. 818–828, 1999.

[3] T. Wang, H. Gao, and J. Qiu, "A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 416–425, 2016.

[4] Z.-G. Wu, P. Shi, H. Su, and J. Chu, "Exponential stabilization for sampled-data neural-network-based control systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2180–2190, 2014.

[5] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[6] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[7] D. Silver, A. Huang, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[8] M. Bojarski, D. Del Testa, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[9] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*, pp. 97–117, Springer, 2017.

[10] 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.

[11] L. Pulina and A. Tacchella, "Challenging SMT solvers to verify neural networks," *AI Communications*, vol. 25, no. 2, pp. 117–135, 2012.

[12] L. Pulina and A. Tacchella, "An abstraction-refinement approach to verification of artificial neural networks," in *International Conference on Computer Aided Verification*, pp. 243–257, Springer, 2010.

[13] W. Xiang, H.-D. Tran, and T. T. Johnson, "Reachable set computation and safety verification for neural networks with ReLU activations," *arXiv preprint arXiv: 1712.08163*, 2017.

[14] Z. Xu, H. Su, P. Shi, R. Lu, and Z.-G. Wu, "Reachable set estimation for Markovian jump neural networks with time-varying delays," *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3208–3217, 2017.

[15] Z. Zuo, Z. Wang, Y. Chen, and Y. Wang, "A non-ellipsoidal reachable set estimation for uncertain neural networks with time-varying delay," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 4, pp. 1097–1106, 2014.

[16] M. V. Thuan, H. M. Tran, and H. Trinh, "Reachable sets bounding for generalized neural networks with interval time-varying delay and bounded disturbances," *Neural Computing and Applications*, pp. 1–12, 2016.

[17] W. Xiang, H.-D. Tran, and T. T. Johnson, "Robust exponential stability and disturbance attenuation for discrete-time switched systems under arbitrary switching," *IEEE Transactions on Automatic Control*, 2017, doi: 10.1109/TAC.2017.2748918.

[18] W. Xiang, "Parameter-memorized Lyapunov functions for discrete-time systems with time-varying parametric uncertainties," *Automatica*, vol. 87, pp. 450–454, 2018.

[19] W. Xiang, J. Lam, and J. Shen, "Stability analysis and $\mathcal{L}_1$-gain characterization for switched positive systems under dwell-time constraint," *Automatica*, vol. 85, pp. 1–8, 2017.

[20] W. Xiang, "Necessary and sufficient condition for stability of switched uncertain linear systems under dwell-time constraint," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3619–3624, 2016.

[21] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation for switched linear systems and its application in safety verification," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5380–5387, 2017.

[22] W. Xiang, H.-D. Tran, and T. T. Johnson, "On reachable set estimation for discrete-time switched linear systems under arbitrary switching," in *American Control Conference (ACC), 2017*, pp. 4534–4539, IEEE, 2017.

[23] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2E2: a verification tool for stateflow models," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 68–82, Springer, 2015.

[24] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala, "Automatic reachability analysis for nonlinear hybrid models with c2e2," in *International Conference on Computer Aided Verification*, pp. 531–538, Springer, 2016.

[25] S. Bak and P. S. Duggirala, "HyLAA: A tool for computing simulation-equivalent reachability for linear systems," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pp. 173–178, ACM, 2017.

[26] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[27] X. Zeng and D. S. Yeung, "Sensitivity analysis of multilayer perceptron to input and weight perturbations," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1358–1366, 2001.

[28] X. Zeng and D. S. Yeung, "A quantified sensitivity measure for multilayer perceptron to input perturbation," *Neural Computation*, vol. 15, no. 1, pp. 183–212, 2003.

[29] X.-Z. Wang, Q.-Y. Shao, Q. Miao, and J.-H. Zhai, "Architecture selection for networks trained with extreme learning machine using localized generalization error model," *Neurocomputing*, vol. 102, pp. 3–9, 2013.

[30] S. W. Piche, "The selection of weight accuracies for madalines," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 432–445, 1995.

[31] D. Shi, D. S. Yeung, and J. Gao, "Sensitivity analysis applied to the construction of radial basis function networks," *Neural Networks*, vol. 18, no. 7, pp. 951–957, 2005.