**Reachable Set Computation and Safety Verification for Neural Networks with ReLU Activations**

<table>
<thead>
<tr>
<th>Journal:</th>
<th><em>IEEE Transactions on Cybernetics</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID</td>
<td>CYB-E-2017-09-1579</td>
</tr>
<tr>
<td>Manuscript Type:</td>
<td>Regular Paper</td>
</tr>
<tr>
<td>Date Submitted by the Author:</td>
<td>25-Sep-2017</td>
</tr>
<tr>
<td>Complete List of Authors:</td>
<td>Xiang, Weiming; Vanderbilt University, Department of Electrical Engineering and Computer Science; Tran, Hoang-Dung; Vanderbilt University, Department of Electrical Engineering and Computer Science; Johnson, Taylor; Vanderbilt University, Department of Electrical Engineering and Computer Science</td>
</tr>
<tr>
<td>Keywords:</td>
<td>ReLU neural network, reachable set computation, verification</td>
</tr>
</tbody>
</table>
Reachable Set Computation and Safety Verification for Neural Networks with ReLU Activations

Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson

Abstract—Neural networks have been widely used to solve complex real-world problems. Due to the complicate, nonlinear, non-convex nature of neural networks, formal safety guarantees for the output behaviors of neural networks will be crucial for their applications in safety-critical systems. In this paper, the output reachable set computation and safety verification problems for a class of neural networks consisting of Rectified Linear Unit (ReLU) activation functions are addressed. A layer-by-layer approach is developed to compute output reachable set. The computation is formulated in the form of a set of manipulations for a union of polyhedra, which can be efficiently applied with the aid of polyhedron computation tools. Based on the output reachable set computation results, the safety verification for a ReLU neural network can be performed by checking the intersections of unsafe regions and output reachable set. This approach for computing output reachable set of ReLU neural networks is addressed. A numerical example of a randomly generated ReLU neural network is provided to show the effectiveness of the approach developed in this paper.

Index Terms—ReLU neural network, reachable set computation, verification.

I. INTRODUCTION

Artificial neural networks have been widely used in machine learning systems. Applications include adaptive control [1]–[8], pattern recognition [9], [10], game playing [11], autonomous vehicles [12], and many others. Though neural networks have been showing effectiveness and powerful ability in resolving complex problems, they are confined to systems which comply only to the lowest safety integrity levels since, in most of time, a neural network is viewed as a black box without effective methods to assure safety specifications for its outputs. Neural networks are trained over a finite number of input and output data, and are expected to be able to generalize to produce desirable outputs for given inputs even including previously unseen inputs. However, in many practical applications, the number of inputs is essentially infinite, this means it is impossible to check all the possible inputs only by performing experiments and moreover, it has been observed that neural networks can react in unexpected and incorrect ways to even slight perturbations of their inputs [13], which could result in unsafe systems. Hence, methods that are able to provide formal guarantees are in a great demand for verifying specifications or properties of neural networks. Verifying neural networks is a hard problem, even simple properties about them have been proven NP-complete problems [14]. 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 [15], [16]. In [17] an abstraction-refinement approach is proposed for verification of specific networks known as Multi-Layer Perceptrons (MLPs). In [14], a specific kind of activation functions called Rectified Linear Unit (ReLU) is considered for verification of neural networks. A simulation-based approach is developed in [18], which turns the reachable set estimation problem into a neural network maximal sensitivity computation problem that is described in terms of a chain of convex optimization problems. Additionally, some recent reachable set/state estimation results are reported for neural networks [3], [4], [19]–[21], these results that are based on Lyapunov functions certainly have potentials to be further extended to safety verification.

A neural network is comprised of a set of layers of neurons, with a linear combination of values from nodes in the preceding layer and applying an activation function to the result. These activation functions are usually nonlinear functions. In this work, we are going to focus on ReLU activation functions [22], which is widely used in many neural networks [15], [23]–[25]. A ReLU function is basically piecewise linear. It returns zero when the node is with a negative value, implying the node is inactive. On the other hand, when the node is active with a positive value, the ReLU returns the value unchanged. This piecewise linearity allows ReLU neural networks have several advantages such as a faster training process and avoiding gradient vanishing problem. As to output reachable set computation and verification problems addressed in this paper, this piecewise linearity will also play a fundamental role in the computing procedure.

The main contribution of this work is to develop an approach for computing output reachable set of ReLU neural networks and further applying to safety verification problems. Under the assumption that the initial set is described by a union of ployhedra, the output reachable set is computed layer-by-layer via a set of manipulations of ployhedra. For a ReLU
function, there are three cases from the view of the output vectors:

- **Case 1**: All the elements in the input vector are positive, thus the output is exactly equal to the input;
- **Case 2**: All the elements are non-positive so that the ReLU function produces a zero vector according to the definition of ReLU function;
- **Case 3**: The input vector has both positive and non-positive elements. This is a much more intricate case which will be proved later that the outputs belong to a union of polyhedra, that is essentially non-convex.

These above three cases are able to fully characterize the output behaviors of a ReLU function and form the basic idea of computing the output reachable set for neural networks comprised of ReLU neurons. With the above classifications and a complete reachability analysis for ReLU functions, the output reachable set of a ReLU layer can be obtained case by case and in the expression of a union of polyhedra. Then, the approach is generalized from a single layer to a neural network consisting of multiple layers for output reachable set computation of neural networks. Finally, the safety verification can be performed by checking if there is non-empty intersection between the output reachable set and unsafe regions. Since the output reachable set computed in this work is an exact one with respect to an input set, the verification results are sound for both safe and unsafe conclusions. The main benefit of our approach is that all the computation processes are formulated with respect to an input set, the verification results are sound and a complete reachability analysis for ReLU functions, the output of 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

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

where $x_j$ is the $j$th input of the $i$th neuron, $\omega_{ij}$ is the weight from the $j$th input to the $i$th neuron, $\theta_i$ is called the bias of the $i$th neuron, $y_i$ is the output of the $i$th neuron, $f(\cdot)$ is the activation function. The activation function is generally a nonlinear function describing the reaction of $i$th neuron with inputs $x_j(t)$, $j = 1, \ldots, n$. Typical activation functions include rectified linear unit, logistic, tanh, exponential linear unit, linear functions, etc.

An MLP has multiple layers, each layer $\ell$, $1 \leq \ell \leq L$, 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 thus $n^{[L]}$ stands for the last layer, that is the output layer. For a neuron $i$, $1 \leq i \leq n^{[\ell]}$ in layer $\ell$, the corresponding input vector is denoted by $x^{[\ell]} = \left( x_{i1}, \ldots, x_{i n^{[\ell-1]}} \right)^T$ and the weight matrix is

$$ W^{[\ell]} = \begin{bmatrix} \omega_{i1}^{[\ell]}, \ldots, \omega_{i n^{[\ell-1]}}^{[\ell]} \end{bmatrix}^T $$

where $\omega_{ij}^{[\ell]} \in \mathbb{R}^{n^{[\ell-1]}}$ is the weight vector. The bias vector for layer $\ell$ is

$$ \theta^{[\ell]} = \left[ \theta_{i1}^{[\ell]}, \ldots, \theta_{i n^{[\ell-1]}}^{[\ell]} \right]^T. $$

The output vector of layer $\ell$ can be expressed as

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

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

For an MLP, the output of $\ell-1$ layer is the input of $\ell$ layer, and the mapping from the input of input layer $x^{[0]} \in \mathbb{R}^{n_{in}}$ to the output of output layer $y^{[L]} \in \mathbb{R}^{n_{out}}$ stands for the input-output relation of the MLP, denoted by

$$ y^{[L]} = F(x^{[0]}) $$

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

In this work, we aim at a class of activation functions called **Rectified Linear Unit (ReLU)**, which is expressed as below:

$$ f(x) = \text{ReLU}(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}. $$

Thus, the output of a neuron considered in (1) can be rewritten as

$$ y_i = \text{ReLU} \left( \sum_{j=1}^{n} \omega_{ij} x_j + \theta_i \right) $$

and the corresponding output vector of layer $\ell$ becomes

$$ y^{[\ell]} = \text{ReLU}(W^{[\ell]} x^{[\ell]} + \theta^{[\ell]}) $$

in which the ReLU function is applied element-wise.

In the 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...
the 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 reachable set of neural network (2) is stated by the following definition.

**Definition 1:** A neural network in the form of (2) and input \( x^{[0]} \) belonging to a set \( \mathcal{X}^{[0]} \), the output reachable set of (2) is defined by

\[
\mathcal{Y} \triangleq \left\{ y^{[L]} \mid y^{[L]} = F(x^{[0]}), x^{[0]} \in \mathcal{X}^{[0]} \right\}. \tag{6}
\]

In our work, the input set is considered to be a union of \( N \) polyhedra, that is expressed as \( \mathcal{X}^{[0]} = \bigcup_{s=1}^{N_0} \mathcal{X}_s^{[0]} \), where \( \mathcal{X}_s^{[0]} \), \( s = 1, \ldots, N_0 \), are described by

\[
\mathcal{X}_s^{[0]} \triangleq \left\{ x^{[0]} \mid A_s x^{[0]} \leq b_s^{[0]}, x^{[0]} \in \mathbb{R}^{n^{[0]}} \right\}, s = 1, \ldots, N_0. \tag{7}
\]

With respect to input set (7), the reachable set computation problem for neural network (2) with ReLU activations is given as below.

**Problem 1:** Given an input set \( \mathcal{X}^{[0]} \) and a neural network in the form of (2) with ReLU activations described by (5), how to check if condition (9) is satisfied?

Then, we will focus on the safety verification for neural networks. The safety specification for output is expressed by a set defined in the output space, describing the safety requirement. For example, in accordance to input set, the safety region can be also considered as a union of polyhedra defined in output space as \( \mathcal{S} = \bigcup_{m=1}^{M} S_m \), where \( S_m, m = 1, \ldots, M \), are given by

\[
S_m \triangleq \left\{ y^{[L]} \mid C_m y^{[L]} \leq d_m, y^{[L]} \in \mathbb{R}^{n^{[L]}} \right\}, m = 1, \ldots, M. \tag{8}
\]

The safety region in the form of (8) formalizes the safety requirements for output \( y^{[L]} \). If output \( y^{[L]} \) belongs to safety region \( \mathcal{S} \), we say the neural network is safe, otherwise, it is called unsafe.

**Definition 2:** A neural network in the form of (2) and safety region \( \mathcal{S} \), the MLP is safe if and only if the following condition is satisfied:

\[
\mathcal{Y} \cap \neg \mathcal{S} = \emptyset \tag{9}
\]

where \( \neg \) is the symbol for logical negation and \( \mathcal{Y} \) is the output reachable set of MLP defined by (6).

Therefore, the safety verification problem for MLP with ReLU activations can be stated as follows.

**Problem 2:** Given an input set \( \mathcal{X} \) by (7), a safety specification by (8), a neural network in the form of (2) with ReLU activation by (5), how to check if condition (9) is satisfied?

The two above linked problems are the main concerns to be addressed in the rest of this paper. The crucial step is to find an efficient way to compute the output reachable set for a ReLU neural network with a given input set. In the next sections, the main results will be presented for the output reachable set computation and safety verification for ReLU neural networks.

III. OUTPUT REACH SET COMPUTATION OF ReLU FUNCTIONS

In this section, we consider the output reachable set of a single ReLU function \( y = \text{ReLU}(x) \) with an input set \( \mathcal{X} \subset \mathbb{R}^n \). Before presenting the result, an indicator vector \( p = [p_0, \ldots, p_n] \), \( p_i \in \{0, 1\} \), is introduced for the following derivation. In the indicator vector \( p \), the element \( p_i \) is valued as below:

\[
p_i = \begin{cases} 0 & x_i \leq 0, i = 1, \ldots, n. \\
1 & x_i > 0, \end{cases}
\]

Considering all the valuations of \( p_i \) in \( p \), there are \( 2^n \) possible valuations in total, which are indexed as

\[
\begin{bmatrix}
p_0 \\
p_1 \\
\vdots \\
p_{2^n-1}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\vdots & \vdots & \vdots & \vdots \\
1 & 1 & 1 & 1
\end{bmatrix}.
\]

Furthermore, each indicator vector from \( p_0 \) to \( p_{2^n-1} \) are diagonalized and denoted as

\[
\begin{bmatrix}
P_0 \\
P_1 \\
\vdots \\
P_{2^n-1}
\end{bmatrix} = \begin{bmatrix}
\text{diag}(p_0) \\
\text{diag}(p_1) \\
\vdots \\
\text{diag}(p_{2^n-1})
\end{bmatrix}.
\]

Now, we are ready to compute the output reachable set of ReLU function \( y = \text{ReLU}(x) \) with an input \( x \in \mathcal{X} \subset \mathbb{R}^n \). For the input set, we have three cases listed below:

- **Case 1:** All the elements are positive by the input in \( \mathcal{X} \), that implies

\[
x \in \mathcal{X}^+ = \{ x \mid x > 0, x \in \mathcal{X} \}.
\]

According to the definition of ReLU function, the output set should be

\[
\mathcal{Y}^+ = \{ y \mid y = x, x \in \mathcal{X}^+ \}. \tag{10}
\]

- **Case 2:** All the elements in the outputs are non-positive, which means

\[
x \in \mathcal{X}^- = \{ x \mid x \leq 0, x \in \mathcal{X} \}.
\]

By the definition of ReLU, it directly leads to

\[
\mathcal{Y}^- = \{ y \mid y = 0, x \in \mathcal{X}^- \} \tag{11}
\]

- **Case 3:** The outputs have both positive and non-positive elements, which corresponds to indicator vectors \( p_h \), \( h = 1, \ldots, 2^n - 2 \). Note that, for each \( p_h \), \( h = 1, \ldots, 2^n - 2 \), the element \( p_i = 0 \) indicates \( y_i = \text{ReLU}(x_i) = 0 \) due to \( x_i \leq 0 \). With respect to each \( p_h \), \( h = 1, \ldots, 2^n - 2 \) and noting \( x = [x_1, \ldots, x_n] \), we define the set

\[
\mathcal{X}^-_h = \{ x \mid x_i \leq 0, x \in \mathcal{X} \}, h = 1, \ldots, 2^n - 2
\]
characterize the behaviors of ReLU functions due to $\mathcal{X} = \mathcal{X}^+ \cup \mathcal{X}^- \cup \bigcup_{h=1}^{2^{n-2}} \mathcal{Y}_h$. For the case of $x \in \mathcal{X}^+$, it produces $\mathcal{Y}^+$, and $x \in \mathcal{X}^-$ leads to $\mathcal{Y}^-$. As to $x \in \mathcal{X}_h^-$, $h = 1, \ldots, 2^n - 2$, the output reachable set is $\bigcup_{h=1}^{2^{n-2}} \mathcal{Y}_h$. Thus, the output reachable set $\mathcal{Y}$ is the union of output sets of three cases, that is $\mathcal{Y} = \mathcal{Y}^+ \cup \mathcal{Y}^- \cup \bigcup_{h=1}^{2^{n-2}} \mathcal{Y}_h$.

Theorem 1 gives out a general result for output reachable set of a ReLU function, since there is no restriction imposed on input set $\mathcal{X}$. In the following, we consider the input set as a union of polyhedra described by $\mathcal{X} = \bigcup_{s=1}^N \mathcal{X}_s$, where $\mathcal{X}_s$, $s \in \{1, \ldots, N\}$ are given as

$$\mathcal{X}_s \triangleq \{x \mid A_s x \leq b_s, \ x \in \mathbb{R}^n\}, \ s = 1, \ldots, N.$$  

(14)

Based on Theorem 1, the following result can be derived for input sets described by a union of polyhedra.

Theorem 2: Given a ReLU function $y = \text{ReLU}(x)$ with an input set $\mathcal{X} = \bigcup_{s=1}^N \mathcal{X}_s$ in which $\mathcal{X}_s$, $s = 1, \ldots, N$, is defined by (14), its output reachable set $\mathcal{Y}$ is

$$\mathcal{Y} = \bigcup_{s=1}^N \left( \mathcal{Y}_s^+ \cup \mathcal{Y}_s^- \cup \bigcup_{h=1}^{2^{n-2}} \mathcal{Y}_{h,s} \right)$$  

(15)

where $\mathcal{Y}_{h,s}$, $h = 1, \ldots, 2^n - 1$, $s = 1, \ldots, N$, are as follows

$\mathcal{Y}_s^+ = \{y \mid A_s y \leq b_s \land y \geq 0, \ y \in \mathbb{R}^n\};$

$\mathcal{Y}_s^- = \{0\}, \ X_s^- = \emptyset,$

$\mathcal{X}_s^- = \{x \mid A_s x \leq b_s, \ x \in \mathbb{R}^n\};$

$\mathcal{Y}_{h,s} = \{y \mid \begin{bmatrix} A_s & I \\ P_h & 0 \end{bmatrix} y \leq \begin{bmatrix} b_s \\ 0 \end{bmatrix}, \ y \in \mathbb{R}^n\}.$

Proof: First, when $x \in \mathcal{X}_s^+$, it has $y = x$, thus the output set $\mathcal{Y}_s^+ = \mathcal{X}_s^+$, which is

$\mathcal{Y}_s^+ = \{y \mid A_s y \leq b_s \land y \geq 0, \ y \in \mathbb{R}^n\}.$

Then, if $x \in \mathcal{X}_s^-$, where $\mathcal{X}_s^-$ can be defined by

$\mathcal{X}_s^- = \{x \mid A_s x \leq b_s, \ x \in \mathbb{R}^n\}.$

According to the definition of ReLU functions, it directly shows that the output set $\mathcal{Y}_s^-$ is

$\mathcal{Y}_s^- = \{0\}, \ X_s^- \neq \emptyset.$

Finally, we consider $x \in \mathcal{X}_{h,s}^-$, $h = 1, \ldots, 2^n - 2$, which can be expressed by

$\mathcal{X}_{h,s}^- = \{x \mid \begin{bmatrix} A_s & I - P_h \end{bmatrix} x \leq \begin{bmatrix} b_s \\ 0 \end{bmatrix}, \ x \in \mathbb{R}^n\}$

where $h = 1, \ldots, 2^n - 2$.

Adding the additional constraint that $y = P_h x \geq 0$, set $\mathcal{X}_{h,s}^+, h = 1, \ldots, 2^n - 2$, is expressed as

$\mathcal{X}_{h,s}^+ = \{x \mid \begin{bmatrix} A_s \\ -P_h \end{bmatrix} x \leq \begin{bmatrix} b_s \\ 0 \end{bmatrix}, \ x \in \mathbb{R}^n\}$

with $h = 1, \ldots, 2^n - 2$. 

![Fig. 1. Visualization of ReLU function $y = \text{ReLU}(x)$, $x \in \mathcal{X} \triangleq \{x \mid A x \leq b, x \in \mathbb{R}^n\}$. (a) is for the Case 1, the red area is for $\mathcal{Y}^+$; (b) is for the Case 2 where the red dot denotes $y = 0$; (c) and (d) are for the Case 3. The red lines on the axis are $Y_h, h = 1, 2$. The resulting output set in (d) is non-convex but expressed by a union of polyhedra.](image-url)
By the fact of \( y = x, \ x \in X_{h,s}^+, \ h = 1, \ldots, 2^n - 2 \), it can be obtained that

\[
Y_{h,s} = X_{h,s}^+, \ h = 1, \ldots, 2^n - 2.
\]

Thus, based on Theorem 1, the output reachable set for input set \( X_s \) is

\[
Y_s = Y_s^+ \cup Y_s^- \cup \left( \bigcup_{h=1}^{2^n-2} Y_{h,s} \right).
\]

Moreover, for input set \( X = \bigcup_{s=1}^{N} X_s \), the output reachable set is \( Y = \bigcup_{s=1}^{N} Y_s \), which implies (15) holds.

According to the result in Theorem 2, the following useful corollary can be derived before ending this subsection.

**Corollary 1:** Given a ReLU function \( y = \text{ReLU}(x) \), if the input set \( X \) is a union of polyhedra, then the output reachable set \( Y \) is also a union of polyhedra.

**Proof:** By Theorem 2, \( Y_{s}^+, Y_{s}^-, Y_{h,s}, \ h = 1, \ldots, 2^n - 1, \ s = 1, \ldots, N, \) are all defined as polyhedra when input set \( X \) is a union of polyhedra, thus \( Y = \bigcup_{s=1}^{N} \left( Y_s^+ \cup Y_s^- \cup \left( \bigcup_{h=1}^{2^n-2} Y_{h,s} \right) \right) \) is a union of polyhedra. The proof is complete.

Theorems 1 and 2 present the output reach set of a ReLU function. Following those results for ReLU functions, the reachable set for neural networks composed of ReLU activations will be studied in the next section.

### IV. Reach Set Computation and Verification for ReLU Neural Networks

Based on the reachability analysis for ReLU activation functions in previous section, we are ready for the main problems in this paper, the reachable set computation and verification problems for ReLU neural networks. First, ReLU neural networks can be expressed recursively in a layer-by-layer form of

\[
\begin{cases}
  y^{[\ell]} = \text{ReLU}(W^{[\ell]}x^{[\ell]} + \theta^{[\ell]}), & \ell = 1, \ldots, L \\
  x^{[\ell+1]} = y^{[\ell]}, & \ell = 1, \ldots, L - 1 \\
  x^{[1]} = x^{[0]}
\end{cases}
\]  

(16)

where \( x^{[0]} \in X^{[0]} \subseteq \mathbb{R}^{n^{[0]}} \) is the input set defined by a union of polyhedra as in (7). The input set and output set of layer \( \ell \) are denoted as \( X^{[\ell]} \) and \( Y^{[\ell]} \), respectively.

**Lemma 1:** Consider neural network (16) with input set \( X^{[0]} \) defined by a union of polyhedra (7), the output sets of each layer \( Y^{[\ell]}, \ \ell = 1, \ldots, L \), are all defined by a union of polyhedra.

**Proof:** Consider input set of layer \( \ell = 1, X^{[1]} = X^{[0]} \) which is a union of polyhedra, the following set which is an affine map of \( X^{[0]} \) should be a union of polyhedra

\[
X^{[1]} = \left\{ x^{[1]} \mid \hat{x}^{[1]} = W^{[1]}x^{[1]} + \theta^{[1]}, \ x \in X^{[0]} \right\}.
\]

Then, using Corollary 1 and \( Y^{[1]} = \text{ReLU}(X^{[1]}) \), the output reachable set \( Y^{[1]} \) is a union of polyhedra.

Also from (16), because we have

\[
X^{[\ell+1]} = Y^{[\ell]}, \ \ell = 1, \ldots, L - 1
\]

The above procedure can be iterated from \( \ell = 1 \) to \( \ell = L \) to claim \( Y^{[\ell]}, \ \ell = 1, \ldots, L \), are all defined by a union of polyhedra.

By Lemma 1, the output sets of each layer are defined as a union of polyhedra, and due to \( X^{[\ell+1]} = Y^{[\ell]} \), the input set of each layer can be represented as \( X^{[\ell]} = \bigcup_{s=1}^{N^{[\ell]}} X^{[\ell]}_s \), in which \( X^{[\ell]}_s \) is

\[
X^{[\ell]}_s \triangleq \left\{ x^{[\ell]} \mid A^{[\ell]}_s x^{[\ell]} \leq b^{[\ell]}_s, \ x \in \mathbb{R}^{n^{[\ell]}} \right\}.
\]

(17)

With regard to the input set of layer \( \ell \) described by (17), the output reachable set can be obtained by the following theorem, which is the main result in this paper.

**Theorem 3:** Consider layer \( \ell \) of ReLU neural network (16) with input set \( X^{[\ell]} \) defined by (17), the output reachable set \( Y^{[\ell]} \) of layer \( \ell \) is

\[
Y^{[\ell]} = \bigcup_{s=1}^{N^{[\ell]}} \left( Y^{[\ell]}_s^+ \cup Y^{[\ell]}_s^- \cup \left( \bigcup_{h=1}^{2^n-2} Y^{[\ell]}_{h,s} \right) \right)
\]

(18)

where

\[
\begin{align*}
Y^{[\ell]}_s^+ &= \left\{ y \mid y = W^{[\ell]}x^{[\ell]} + \theta^{[\ell]}, \ x \in X^{[\ell]}_s \right\}, \\
X^{[\ell]}_s &= \left\{ x^{[\ell]} \mid A^{[\ell]}_s x^{[\ell]} \leq b^{[\ell]}_s \land W^{[\ell]}x^{[\ell]} > -\theta^{[\ell]}, \ x^{[\ell]} \in \mathbb{R}^{n^{[\ell]}} \right\}; \\
Y^{[\ell]}_s^- &= \left\{ \left\{ 0 \right\}, \ X^{[\ell]}_s^- = \emptyset, \ x^{[\ell]} \in \mathbb{R}^{n^{[\ell]}} \right\}; \\
X^{[\ell]}_h,s &= \left\{ \left[ \begin{array}{c}
A^{[\ell]}_s \\
W^{[\ell]}
\end{array} \right] \left[ \begin{array}{c}
x^{[\ell]} \\
\theta^{[\ell]}
\end{array} \right], x^{[\ell]} \in \mathbb{R}^{n^{[\ell]}} \right\}; \\
Y^{[\ell]}_{h,s} &= \left\{ \left[ \begin{array}{c}
b^{[\ell]}_s \\
-P_h W^{[\ell]}
\end{array} \right], b^{[\ell]}_s \in \mathbb{R}^{n^{[\ell]}}, (P_h - 1)\theta^{[\ell]} \right\}.
\end{align*}
\]

**Proof:** The proof is briefly presented below since it is similar to the proof line for Theorem 2.

Like the proof line in Theorem 2, when \( y > 0 \) which means \( W^{[\ell]}x^{[\ell]} + \theta^{[\ell]} < 0 \), combining \( A^{[\ell]}_s x^{[\ell]} \leq b^{[\ell]}_s \), it defines a set \( X^{[\ell]}_s^+ \) as

\[
X^{[\ell]}_s^+ = \left\{ x^{[\ell]} \mid A^{[\ell]}_s x^{[\ell]} \leq b^{[\ell]}_s \land W^{[\ell]}x^{[\ell]} > -\theta^{[\ell]}, x^{[\ell]} \in \mathbb{R}^{n^{[\ell]}} \right\}.
\]

Moreover, due to \( y^{[\ell]} = W^{[\ell]}x^{[\ell]} + \theta^{[\ell]} \), the output reachable set for \( Y^{[\ell]} \) is

\[
Y^{[\ell]}_s^+ = \left\{ y \mid y = W^{[\ell]}x^{[\ell]} + \theta^{[\ell]}, \ x \in X^{[\ell]}_s^+ \right\}.
\]

Similarly, if \( y \leq 0 \), we have set \( X^{[\ell]}_s^- \) defined by

\[
X^{[\ell]}_s^- = \left\{ x^{[\ell]} \mid A^{[\ell]}_s x^{[\ell]} \leq b^{[\ell]}_s, x^{[\ell]} \in \mathbb{R}^{n^{[\ell]}} \right\}.
\]

Using ReLU function to get

\[
Y^{[\ell]}_s^- = \left\{ \left\{ 0 \right\}, \ X^{[\ell]}_s^- = \emptyset \right\}.
\]
Lastly, we consider the case that \( y \) has both positive and non-positive elements, that is \( y \in \mathcal{Y}_{h,s}^{\ell} \), \( h = 1, \ldots , 2^n - 2 \), where \( \mathcal{Y}_{h,s}^{\ell} \) is expressed as follows

\[
\mathcal{Y}_{h,s}^{\ell} = \{ y \mid y = \mathbf{W}^{\ell}x^{\ell} + \mathbf{\theta}^{\ell}, \ x \in \mathcal{X}_{h,s}^{\ell} \}
\]

\[
\mathcal{A}_{h,s}^{\ell} = \{ x^{\ell} \mid \mathbf{A}_{h,s}^{\ell} x^{\ell} \leq b_{h,s}^{\ell}, \ x^{\ell} \in \mathbb{R}^n \}
\]

where

\[
\mathbf{A}_{h,s}^{\ell} = \left[ \begin{array}{c} \mathbf{A}_{h,s}^{\ell} \\ \mathbf{I} - \mathbf{P}_h \mathbf{W}^{\ell} \end{array} \right], \quad \mathbf{b}_{h,s}^{\ell} = \left[ \begin{array}{c} \mathbf{b}_{h,s}^{\ell} \\ \mathbf{P}_h \mathbf{\theta}^{\ell} \end{array} \right].
\]

Furthermore, due to \( y \geq 0 \), an additional constraint \( \mathbf{P}_h (\mathbf{W}^{\ell}x^{\ell} + \mathbf{\theta}^{\ell}) \geq 0 \) should be added to \( \mathcal{X}_{h,s}^{\ell} \) to obtain \( \mathcal{A}_{h,s}^{\ell} \) as

\[
\mathcal{X}_{h,s}^{\ell} = \{ x^{\ell} \mid \mathbf{A}_{h,s}^{\ell} x^{\ell} \leq \mathbf{b}_{h,s}^{\ell}, \ x^{\ell} \in \mathbb{R}^n \}
\]

where

\[
\mathbf{A}_{h,s}^{\ell} = \left[ \begin{array}{c} \mathbf{A}_{h,s}^{\ell} \\ \mathbf{P}_h \mathbf{W}^{\ell} \end{array} \right], \quad \mathbf{b}_{h,s}^{\ell} = \left[ \begin{array}{c} \mathbf{b}_{h,s}^{\ell} \\ \mathbf{P}_h \mathbf{\theta}^{\ell} \end{array} \right].
\]

Then, following the guidelines of Theorem 2, the output reachable set of the form (18) can be established.

**Algorithm 1:** Output Reach Set Computation for ReLU Layers

**Input:** ReLU neural network weight matrix and bias \((\mathbf{W}^{\ell}, \mathbf{\theta}^{\ell})\), input set \(\mathcal{X}_i^{\ell} = \bigcup_{s=1}^{N_s} \mathcal{A}_s^{\ell} \) with \(\mathcal{A}_s^{\ell} \triangleq \{ x^{\ell} \mid \mathbf{A}_s^{\ell} x^{\ell} \leq \mathbf{b}_s^{\ell}, \ x \in \mathbb{R}^n \}\).

**Output:** Output reachable set \(\mathcal{Y}_i^{\ell}\),

1. function \( \text{OUTPUTRELU}((\mathbf{W}^{\ell}, \mathbf{\theta}^{\ell}), \mathcal{X}_i^{\ell}) \)
2. for \( s = 1 : 1 : N_s \)
3. \( \mathcal{X}_s^{\ell+} \leftarrow \{ x^{\ell} \mid \mathbf{A}_s^{\ell} x^{\ell} \leq \mathbf{b}_s^{\ell} \land \mathbf{W}^{\ell}x^{\ell} > -\mathbf{\theta}^{\ell} \}\)
4. \( \mathcal{Y}_s^{\ell+} \leftarrow \{ y \mid y = \mathbf{W}^{\ell}x^{\ell} + \mathbf{\theta}^{\ell}, \ x \in \mathcal{X}_s^{\ell+} \}\)
5. \( \mathcal{X}_s^{\ell-} \leftarrow \{ x^{\ell} \mid \mathbf{A}_s^{\ell} \mathbf{W}^{\ell} x^{\ell} \leq \mathbf{b}_s^{\ell} \}\)
6. if \( \mathcal{X}_s^{\ell-} \neq \emptyset \) then
7. \( \mathcal{Y}_s^{\ell-} \leftarrow \{ 0 \}\)
8. else
9. \( \mathcal{Y}_s^{\ell-} \leftarrow \emptyset \)
10. end if
11. for \( h = 1 : 1 : 2^n - 2 \)
12. \( \mathbf{A}_{h,s}^{\ell} \leftarrow \left( \mathbf{I} - \mathbf{P}_h \mathbf{W}^{\ell} \right) \)
13. \( \mathbf{b}_{h,s}^{\ell} \leftarrow \left( \mathbf{P}_h \mathbf{\theta}^{\ell} \right) \)
14. \( \mathcal{X}_{h,s}^{\ell} \leftarrow \{ x^{\ell} \mid \mathbf{A}_{h,s}^{\ell} x^{\ell} \leq \mathbf{b}_{h,s}^{\ell}, \ x^{\ell} \in \mathbb{R}^n \}\)
15. \( \mathcal{Y}_{h,s}^{\ell} \leftarrow \{ y \mid y = \mathbf{W}^{\ell}x^{\ell} + \mathbf{\theta}^{\ell}, \ x \in \mathcal{X}_{h,s}^{\ell} \}\)
16. end for
17. \( \mathcal{Y}_s^{\ell} \leftarrow \mathcal{Y}_s^{\ell+} \cup \mathcal{Y}_s^{\ell-} \cup \left( \bigcup_{h=1}^{2^n-2} \mathcal{Y}_{h,s}^{\ell} \right) \)
18. end for
19. return \( \mathcal{Y}_i^{\ell} \leftarrow \bigcup_{s=1}^{N_s} \mathcal{Y}_s^{\ell} \)
20. end function

**Algorithm 2:** Output Reach Set Computation for ReLU Networks

**Input:** ReLU neural network weight matrices and biases \((\mathbf{W}^{\ell}, \mathbf{\theta}^{\ell}), \ell = 1, \ldots , L\), input set \(\mathcal{X}_0 = \bigcup_{s=1}^{N_s} \mathcal{A}_s^{\ell} \) with \(\mathcal{A}_s^{\ell} \triangleq \{ x^{\ell} \mid \mathbf{A}_s^{\ell} x^{\ell} \leq \mathbf{b}_s^{\ell}, \ x \in \mathbb{R}^n \}\).

**Output:** Output reachable set \(\mathcal{Y}_i^{\ell}\),

1. function \( \text{OUTPUTRELUNETWORK}((\mathbf{W}^{\ell}, \mathbf{\theta}^{\ell}), \ell = 1, \ldots , L, \mathcal{X}_0) \)
2. \( \mathcal{X}_i^{\ell} \leftarrow \mathcal{X}_0 \)
3. for \( \ell = 1 : 1 : L \)
4. if Layer \( \ell \) is a linear layer then
5. \( \mathcal{Y}_i^{\ell} \leftarrow \{ y \mid y = \mathbf{W}^{\ell}x^{\ell} + \mathbf{\theta}^{\ell}, \ x \in \mathcal{X}_i^{\ell} \}\)
6. \( \mathcal{Y}_i^{\ell} \leftarrow \bigcup_{s=1}^{N_s} \mathcal{Y}_s^{\ell} \)
7. else if Layer \( \ell \) is a ReLU Layer then
8. \( \mathcal{Y}_i^{\ell} \leftarrow \text{OUTPUTRELU}((\mathbf{W}^{\ell}, \mathbf{\theta}^{\ell}), \mathcal{X}_i^{\ell}) \)
9. end if
10. \( \mathcal{X}_i^{\ell+1} \leftarrow \mathcal{Y}_i^{\ell} \)
11. end for
12. return \( \mathcal{Y}_i^{L} \)
13. end function
Algorithm 3 Safety Verification for ReLU Networks

Input: ReLU neural network weight matrices and biases \(\langle W[\ell], \theta[\ell]\rangle\), \(\ell = 1, \ldots, L\), input set \(X[0] = \bigcup_{s=0}^{N_0} X_s\) with \(X_s \triangleq \{x[0] | A[0][s][0]x[0] \leq b[0][s], x \in \mathbb{R}^{n[0]}\}\), safety specification \(S = \bigcup_{m=1}^{M} S_m\) with \(S_m \triangleq \{y[L] | C_m y[L] \leq d_m, y \in \mathbb{R}^{n[L]}\}\).

Output: SAFE or UNSAFE.

1: function \text{VERIFYReLUNetwork}(W[\ell], \theta[\ell], \ell = 1, \ldots, L, X[0], S)
2: \(Y[L] = \text{OutputReLUNetwork}(W[\ell], \theta[\ell], \ell = 1, \ldots, L, X[0])\)
3: if \(Y[L] \cap \neg S = \emptyset\) then
4: \quad return SAFE
5: else
6: \quad return UNSAFE
7: end if
8: end function

Output is the output reachable set \(Y[L]\) of the ReLU neural network. It is used for computing the output reachable set of a ReLU neural network. It should be noted that the final layer is also allowed to exist in the neural network. Moreover, it is worth mentioning that the output reachable set computed by function \text{OutputReLUNetwork} is the exact output reachable set, not an over-approximation.

Using the output reachable set computed in function \text{OutputReLUNetwork}, the safety verification for ReLU neural networks can be converted to check if there exists non-empty intersection between output reachable set and unsafe regions, which is summarized as function \text{VERIFYReLUNetwork}.

The input of function \text{VERIFYReLUNetwork} is weight matrices and biases \(\langle W[\ell], \theta[\ell]\rangle\), \(\ell = 1, \ldots, L\), input set \(X[0]\) and safety specification \(S\). The output is to claim the neural network safe or unsafe. 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.

The computation involved in the developed algorithms is operations of polyhedra, thus the computational complexity closely relates to the number of polyhedra generated and involved in the operations. For Algorithm 1, the number of polyhedra contained in the output reachable set \(Y[\ell]\) for the single layer \(\ell\) is \(N_\ell 2^{n[\ell]}\). Furthermore, considering a ReLU neural network with \(L\) layers, the number of polyhedra for the \(i\)th layer is \(N_0 \prod_{\ell=1}^{L} 2^{n[\ell]}\). It can be observed that the number of polyhedra increases exponentially as the number of layer grows, that also indicates that the neural networks with multiple layers have great ability to approximate complex functions and resolve complicate problems. The output of a ReLU network consisting of \(L\) layers contains \(N_0 \prod_{\ell=1}^{L} 2^{n[\ell]}\) polyhedra. The numbers for a single layer, the layer in a network and the whole neural network are listed in Table 1.

The numbers are used to determine the number of polyhedra generated for each discretizing point. As shown in Figure 4, it can be observed that all the generated outputs are located in the computed reachable set to validate our approach.

When run Algorithm 2, the number of polyhedra grows from 20 to 1250 as the layer goes from 1 to 8, which is shown in Figure 5. It is noted that the output layer is a linear layer so that the number of polyhedra remains unchanged for the output layer 8. Then, in order to reduce the computational

---

### Table I

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Single Layer (\ell)</th>
<th>(i)th Layer</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_\ell 2^{n[\ell]})</td>
<td>(N_0 \prod_{\ell=1}^{L} 2^{n[\ell]})</td>
<td>(N_0 \prod_{\ell=1}^{L} 2^{n[\ell]})</td>
<td></td>
</tr>
</tbody>
</table>

---

In practice, the number is usually smaller than them since empty sets are produced for the output of the layers during the computation procedure.

In addition, it can be observed that the operations for computing \(Y[\ell]^+\), \(Y[\ell]^−\) and \(Y[\ell]_{h,s}\) are all independent, thus those computations in Algorithm 1 can be executed in a parallel manner to enhance the scalability of the proposed approach. In the following example, it will be shown how the computational time will be reduced by employing parallel computing techniques.

V. Numerical Example

Consider a neural network with 3 inputs, 2 outputs and 7 hidden layers, see Figure 2. Each layer has 7 neurons, the weights and biases are all randomly generated. Hidden layers are chosen as ReLUs and the output layer is with linear functions. The example is computed by using Matlab and Multi-parametric toolbox 3.0 [26] on a personal computer with Windows 7, Intel Core i5-4200U, 1.6GHz, 4 GB RAM.

The first objective in this example is to compute the output reachable set of this ReLU neural network with respect to polytopic input sets.

![Image](image-url)
Given an input set $X^0 \triangleq \{ x \mid \|x\|_\infty \leq 1, x \in \mathbb{R}^3 \}$, the output reachable set of the proposed neural network is computed out and described by a union of 1250 polyhedra.

Discretizing the input set $X^0$ by step 0.05, we generate 8000 outputs from those discretized points. The outputs are denoted by marker $\ast$, it can be seen that all the outputs are located in the computed reachable set.

Number of polyhedra produced by each layer is increasing as the layer goes deeper. However, due to the existence of empty sets during the computation process, the number of polyhedra produced in each layer is actually much smaller than the maximal number as shown in Table I.

With the output reachable set depicted in Figure 3, the safety property can be easily checked via function VeriReLUNetwork presented in Algorithm 3. Consider the unsafe region for the output $y^{[L]}$ is described by

$$S_{\text{unsafe}} = \{ y^{[L]} \mid \| y^{[L]} - y_0^{[L]} \|_\infty \leq 1 \}$$

which can be expressed in the hyperplane representation of

$$S_{\text{unsafe}} = \{ y^{[L]} \mid Cy^{[L]} \leq d, y \in \mathbb{R}^{n^{[L]}} \}$$

with $C$ and $d$ defined by

$$C = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad d = \begin{bmatrix} 1 + y_0^{[L]}(1) \\ 1 - y_0^{[L]}(2) \\ 1 + y_0^{[L]}(3) \\ 1 - y_0^{[L]}(4) \end{bmatrix}.$$ 

We can check the feasibility of linear programming problem formulated by the hyperplane representations of $y^{[L]}$ and $S_{\text{unsafe}}$ to determine if there exists nonempty intersection. If we choose $y_0 = [0, 0]^T$, the ReLU neural network is safe since there is no intersection between $S_{\text{unsafe}}$ and $y^{[L]}$. However,
if we select \( y_0 = [0, 5]^T \), the ReLU neural network is unsafe since \( S_{\text{unsafe}} \cap \{ y \neq 0 \} \neq \emptyset \). Figure 6 shows the safety verification results regarding the two unsafe regions.

VI. CONCLUSIONS AND FUTURE REMARK

This paper presents a novel output reachable set computation method for neural networks equipped with ReLU activations. With the aid of reachability analysis for ReLU functions, the output reachable set computation for ReLU layer can be made by a set of manipulations for polyhedra, and the output reachable set can be described by a union of a number of polyhedra if the input of the layer is given as a union of polyhedra. Since the involved computation steps are all in the form of operation for polyhedra, they can be efficiently implemented with the help of existing tools for computational geometry. Lastly, on the basis of computed output reachable set, the safety verification for ReLU neural networks can be done by checking for the nonempty intersection of output reachable set and unsafe region.

With the reachable set computed in this paper, it absolutely has potentials to be further applied into many other problems involving ReLU neural networks such as safety-oriented neural network control synthesis problems, which will be our future study based on this paper. Beside the application of the results in this paper, another meaningful research direction is to further improve the scalability of the approach developed in this paper to deal with deeper neural networks. The key step will be finding out computation-efficient method to use as few polyhedra as possible to describe the output sets produced by hidden layers during the computation procedure. This will also be our future study for the reachable set computation problem of neural networks.

REFERENCES

Visualization of ReLU function $\mathbf{y} = \mathrm{ReLU}(\mathbf{x})$, $x \in \mathcal{X} \triangleq \{\mathbf{x} \mid \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^2\}$. (a) is for the Case 1, the red area is for $\mathcal{Y}^+$. (b) is for the Case 2 where the red spot denotes $\mathbf{y} = \mathbf{0}$. (c) and (d) are for the Case 3, the red lines on the axis are $\mathcal{Y}_h$, $h=1,2$. The resulting output set in (d) is non-convex but expressed by a union of polyhedra.
Neural network with 3 inputs, 2 outputs and 7 hidden layers of 7 neurons. The hidden layers are with ReLU activation functions and output layer is with linear functions. The weight matrices and bias vectors for the neural network in the example are randomly generated.

49x19mm (300 x 300 DPI)
Given an input set $\mathcal{X}^{[0]} \triangleq \{\mathbf{x} \mid \|\mathbf{x}\|_{\infty} \leq 1, \mathbf{x} \in \mathbb{R}^{3}\}$, the output reachable set of the proposed neural network is computed out and described by a union of 1250 polyhedra.
Discretizing the input set $\mathcal{X}^{[0]}$ by step 0.05, we generate 8000 outputs from those discretized points. The outputs are denoted by marker $*$, it can be seen that all the outputs are located in the computed reachable set.
Number of polyhedra produced by each layer is increasing as the layer goes deeper. However, due to the existence of empty sets during the computation process, the number of polyhedra produced in each layer is actually much smaller than the maximal number as shown in Table I.
Safety verification regarding unsafe regions. The green area is the computed reachable set, and we consider two unsafe regions depicted in blue and red. (1) The neural network is safe for the blue Unsafe Region 1 since there is no intersection for reachable set and unsafe region; (2) For the red Unsafe Region 2, the neural network is unsafe due to the existence of nonempty intersection.