2022 American Control Conference (ACC)
Atlanta, USA, June 8-10, 2022

Approximate Bisimulation Relations for Neural Networks and Application to Assured Neural Network Compression

Weiming Xiang and Zhongzhu Shao

***Abstract*— In this paper, we propose a concept of approximate bisimulation relation for feedforward neural networks. In the framework of approximate bisimulation relation, a novel neural network merging method is developed to compute the approximate bisimulation error between two neural networks based on reachability analysis of neural networks. The developed method is able to quantitatively measure the distance between the outputs of two neural networks with the same inputs. Then, we apply the approximate bisimulation relation results to perform neural networks model reduction and compute the compression precision, i.e., assured neural networks compression. At last, using the assured neural network compression, we accelerate the verification processes of ACAS Xu neural networks to illustrate the effectiveness and advantages of our proposed approximate bisimulation approach.**

I. Introduction

Deep neural networks (DNN) are now widely used in a variety of contemporary applications, such as image processing [1], pattern recognition [2], [3], adaptive control, [4], [5] autonomous vehicles [6], and other fields, showing the powerful capabilities solving complex and challenging problems that traditional approaches fail to deal with. To cope with complex tasks and different environments, neural network models have been developed with increasing scale and complexity, which aim to provide better performance and higher accuracy. However, the increasing scale and complexity of the neural network models also mean that neural networks require a large number of resources for real-world implementation such as higher memory, more computational power, and higher energy consumption [7]. Therefore, neural network model compression methods were developed to reduce the complexity of neural networks at the least possible price of performance deterioration. For instance, in [8], four compression methods for deep convolutional neural networks are summarized, but some problems were pointed out such as a sharp drop in the accuracy of the network when compressing. More neural network compression results can be found in the recent survey [9] and references therein. Moreover, it has been observed that well-trained neural networks on abundant data are sometimes sensitive to updates, and react in unexpected and incorrect ways to even slight changes of the parameters [10]. The neural network compression inevitably introduces changes to the neural network. Therefore, an approach is needed to formally characterize the changes between the original neural network model and its compressed version.

In this paper, we propose an approximate bisimulation relation between two neural networks, which formally characterize the maximal difference between the outputs of two neural networks generated from the same inputs. Based on the framework of the approximate bisimulation relation, we propose a neural network merging algorithm to calculate the approximate bisimulation error, measuring the distance between two neural networks. Applying this approximate bisimulation method to neural network model compression, we can obtain the precision of neural network model compression, which is able to provide assurance to perform tasks using compressed neural networks on behalf of original ones. To illustrate the feasibility of the approximate bisimulation method, we apply it to accelerate verification processes of the ACAS Xu neural networks using the compressed neural networks.

The remainder of the paper is organized as follows: Preliminaries are given in Section II. The approximate bisimulation relation and approximate bisimulation error computation are presented in Section III. Assured neural network compression and examples are given in Section IV. The conclusion is presented in Section V.

Notations: For the rest of the paper, $\mathbf{0}_{n\times m}$ denotes a matrix of n rows and m columns with all elements zero, $\mathbf{I}_n$ denotes the n-dimensional identity matrix. $\mathsf{purelin}(\cdot)$ is linear transfer function, i.e., $x = \mathsf{purelin}(x)$.

II. Preliminaries

In this paper, we consider a class of feedforward neural networks that generally consist of one input layer, multiple hidden layers and one output layer. Each layer consists of one or multiple neurons. The action of a neuron depends on its activation function, which is in the description of

$$y_i = \phi(\sum\nolimits_{j=1}^{n} w_{ij}x_j + b_i) \tag{1}$$

where y_i is the output of the ith neuron, x_j is the jth input of the ith neuron, w_{ij} is the weight from the jth input to the ith neuron, b_i is the bias of the ith input, $\phi(\cdot)$ is the activation function. Each layer ℓ ($1 \le \ell \le L$) of a feedforward neural network has $n^{\{\ell\}}$ neurons. Layer $\ell = 0$ denotes the input layer, $n^{\{0\}}$ denotes the number of the input for the input layer. For the layer ℓ, the input vector is denoted by $\mathbf{x}^{\{\ell\}}$, respectively, the weight matrix and the bias vector are

$$\mathbf{W}^{\{\ell\}} = [w_1^{\{\ell\}}, \cdots, w_{n^{\{\ell\}}}^{\{\ell\}}]^{\mathrm{T}} \tag{2}$$

$$\mathbf{b}^{\{\ell\}} = [b_1^{\{\ell\}}, \cdots, b_{n^{\{\ell\}}}^{\{\ell\}}]^{\mathrm{T}} \tag{3}$$

This research was supported by the National Science Foundation, under NSF CAREER Award 2143351.

W. Xiang is with the School of Computer and Cyber Sciences, Augusta University, Augusta GA 30912 USA. Email: wxiang@augusta.edu

Z. Shao is with Department of Electrical Engineering, Southwest Jiaotong University, Chengdu, China.

Authorized licensed use limited to: Augusta University. Downloaded on March 02,2023 at 16:50:11 UTC from IEEE Xplore. Restrictions apply.

where $w_i^{\{\ell\}}$ is the weight vector, $b_i^{\{\ell\}}$ is the bias value. The output vector of layer ℓ is $\mathbf{y}^{\{\ell\}}$ defined by

$$\mathbf{y}^{\{\ell\}} = \phi^{\{\ell\}}(\mathbf{W}^{\{\ell\}}\mathbf{x}^{\{\ell\}} + \mathbf{b}^{\{\ell\}}) \tag{4}$$

where $\phi^{\{\ell\}}(\cdot)$ is the activation function of layer ℓ.

For the whole neural network, the input and output layer are $\mathbf{x}^{[0]}$ and $\mathbf{y}^{[L]}$ respectively, the input of the layer ℓ is the output of the layer $\ell-1$, the mapping relation from the input to the output is denoted by

$$\mathbf{y}^{\{L\}} = \Phi(\mathbf{x}^{\{0\}}) \tag{5}$$

where $\Phi(\cdot) \triangleq \phi^{\{L\}} \circ \phi^{\{L-1\}} \cdots \phi^{\{1\}}(\cdot)$. The mapping relation Φ includes not only the activation function of the neural network but also the weight matrix and the bias vectors, which represent the structural information of the neural network.

Given an input set $\mathcal{X}$, the output reachable set of a neural network is stated by the definition below.

Definition 1: Given a neural network in the form of (5) and input set $\mathcal{X} \in \mathbb{R}^{n^{\{0\}}}$, the following set

$$\mathcal{Y} = \left\{\mathbf{y}^{\{L\}} \in \mathbb{R}^{n^{\{L\}}} \mid \mathbf{y}^{\{L\}} = \Phi(\mathbf{x}^{\{0\}}), \mathbf{x}^{\{0\}} \in \mathcal{X}\right\} \tag{6}$$

is called the output reachable set of neural network (5).

The safety specification of a neural network is expressed by the set defined in the output space, describing the safety requirement.

Definition 2: Safety specification $\mathcal{S}$ formalizes the safety requirement for output $\mathbf{y}^{\{L\}}$ of neural network (5), and is a predicate over output $\mathbf{y}^{\{L\}}$ of neural network (5). The neural network (5) is safe if and only if the following condition is satisfied:

$$\mathcal{Y} \cap \neg\mathcal{S} = \emptyset \tag{7}$$

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

The above safety verification concept is reachability-based and will be used in Section IV for safety verification of neural networks of Airborne Collision Avoidance Systems in [11].

III. Approximation Simulation Relations of Neural Networks

A. Approximation Bisimulation Relations

In order to characterize the difference of two feedforward neural networks in terms of outputs, we defined the following metric which measures the distance between the outputs of two neural networks in the framework of the reachable set defined in Definition 1.

Definition 3: Consider two neural networks $\mathbf{y}^{\{L\}} = \Phi_j(\mathbf{x}^{\{0\}})$, $j \in \{1,2\}$, input set $\mathcal{X} \in \mathbb{R}^{n^{\{0\}}}$, and output sets $\mathcal{Y}_j \in \mathbb{R}^{n^{\{L\}}}$, $j \in \{1,2\}$, we define $\mathcal{N}_j = (\mathcal{X}, \mathcal{Y}_j, \Phi_j)$, $j \in \{1,2\}$, and

$$d(\Phi_1(\mathbf{x}_1^{\{0\}}), \Phi_2(\mathbf{x}_2^{\{0\}})) = \begin{cases} \rho(\mathbf{y}_1^{\{L\}}, \mathbf{y}_2^{\{L\}}) & \text{if } \mathbf{x}_1^{\{0\}} = \mathbf{x}_2^{\{0\}} \\ +\infty & \text{otherwise} \end{cases} \tag{8}$$

where

$$\rho(\mathbf{y}_1^{\{L\}}, \mathbf{y}_2^{\{L\}}) = \sup_{\mathbf{y}_1^{\{L\}} \in \mathcal{Y}_1, \mathbf{y}_2^{\{L\}} \in \mathcal{Y}_2} \left\| \mathbf{y}_1^{\{L\}} - \mathbf{y}_2^{\{L\}} \right\|. \tag{9}$$

It is noted that $d(\Phi_1(\mathbf{x}_1^{\{0\}}), \Phi_2(\mathbf{x}_2^{\{0\}}))$ defined in (8) characterizes the maximal difference between the outputs of two neural networks generated from the same input, which quantifies the discrepancy between two neural networks Φ_1 and Φ_2 in terms of outputs. Based on Definition 3, we will be able to establish the approximate bisimulation relation of two neural networks.

Definition 4: Consider $\mathcal{N}_j = (\mathcal{X}, \mathcal{Y}_j, \Phi_j)$, $j \in \{1,2\}$, and let $\varepsilon \geq 0$, a relation $\mathscr{R}_\varepsilon \in \mathbb{R}^{n^{\{L\}}} \times \mathbb{R}^{n^{\{L\}}}$ is called an approximate simulation relation between $\mathcal{N}_1$ and $\mathcal{N}_2$, of precision ε, if for all $(\mathbf{y}_1^{\{L\}}, \mathbf{y}_2^{\{L\}}) \in \mathcal{R}_\varepsilon$

1) $d(\Phi_1(\mathbf{x}^{\{0\}}), \Phi_2(\mathbf{x}^{\{0\}})) \leq \varepsilon$, $\forall \mathbf{x}^{\{0\}} \in \mathcal{X}$;
2) $\forall \mathbf{x}^{\{0\}} \in \mathcal{X}$, $\forall \Phi_1(\mathbf{x}^{\{0\}}) \in \mathcal{Y}_1$, $\exists \Phi_2(\mathbf{x}^{\{0\}}) \in \mathcal{Y}_2$ such that $(\Phi_1(\mathbf{x}^{\{0\}}), \Phi_2(\mathbf{x}^{\{0\}})) \in \mathscr{R}_\varepsilon$;
3) $\forall \mathbf{x}^{\{0\}} \in \mathcal{X}$, $\forall \Phi_2(\mathbf{x}^{\{0\}}) \in \mathcal{Y}_2$, $\exists \Phi_1(\mathbf{x}^{\{0\}}) \in \mathcal{Y}_1$ such that $(\Phi_1(\mathbf{x}^{\{0\}}), \Phi_2(\mathbf{x}^{\{0\}})) \in \mathscr{R}_\varepsilon$

and we say neural networks $\mathcal{N}_1$ and $\mathcal{N}_2$ are approximately bisimilar with precision ε, denoted by $\mathcal{N}_1 \sim_\varepsilon \mathcal{N}_2$.

Remark 1: The meaning of approximate bisimulation between two neural networks $\mathcal{N}_1$ and $\mathcal{N}_2$ with precision ε, which denoted by $\mathcal{N}_1 \sim_\varepsilon \mathcal{N}_2$, is as follows: Considering two neural networks $\mathcal{N}_1$ and $\mathcal{N}_2$ and any output of neural network $\mathcal{N}_1$, we can find one output generated by the same corresponding input out of neural network $\mathcal{N}_2$, and vice versa. The two outputs of two neural networks always satisfy that the distance between them is bounded by ε. In the case of $\varepsilon = 0$, we can define that the two neural networks have an exact simulation relation.

Then, we define metrics measuring the distance between the observed behaviors of neural networks $\mathcal{N}_1$ and $\mathcal{N}_2$. Based on the defined notion of approximate bisimulation, we can define the approximate bisimulation error to represent the distance between two neural networks.

Definition 5: Given two neural networks $\mathcal{N}_1$ and $\mathcal{N}_2$, the approximate bisimulation error of them is defined by

$$d(\mathcal{N}_1, \mathcal{N}_2) = \sup\{\varepsilon \mid \mathcal{N}_1 \sim_\varepsilon \mathcal{N}_2\} \tag{10}$$

where $\varepsilon \geq 0$.

The key to establish the approximation bisimulation relation between two neural networks is how to efficiently compute the approximation bisimulation error defined by (10). In the next subsection, a reachability-based method is proposed to compute the approximate bisimulation error.

B. Approximate Bisimulation Error Computation

In order to compute the approximate bisimulation error ε between two neural network outputs, the set-valued reachability methods can be used. First, consider two neural networks with the same input set $\mathcal{X}$, a feedforward neural network $\mathcal{N}_L$ with L hidden layers and $n^{\{l\}}$, $l = 1, \ldots, L$ neurons in each layer, and its bisimilar feedforward neural network $\mathcal{N}_S$ with S hidden layers and $n^{\{s\}}$, $s = 1, \ldots, S$ neurons in each hidden layer.

Authorized licensed use limited to: Augusta University. Downloaded on March 02,2023 at 16:50:11 UTC from IEEE Xplore. Restrictions apply.

Without loss of generality, the following assumption is given for neural networks $\mathcal{N}_L$ and $\mathcal{N}_S$.

Assumption 1: The following assumptions hold for two neural networks $\mathcal{N}_L$ and $\mathcal{N}_s$:

1) The number of inputs of two neural networks is same, i.e., $n_L^{\{0\}} = n_S^{\{0\}}$;
2) The number of outputs of two neural networks is same, i.e., $n_L^{\{L\}} = n_S^{\{S\}}$;
3) The number of hidden layers of neural network $\mathcal{N}_L$ is greater than or equal the number of hidden layers of neural network $\mathcal{N}_S$, i.e., $L \geq S$.

According to (9), (10), the approximate bisimulation error between $\mathcal{N}_L$ and $\mathcal{N}_S$ can be expressed by

$$d(\mathcal{N}_L, \mathcal{N}_S) = \sup_{\mathbf{x}^{\{0\}} \in \mathcal{X}} \left\| \Phi_L(\mathbf{x}^{\{0\}}) - \Phi_S(\mathbf{x}^{\{0\}}) \right\|. \quad (11)$$

To obtain the approximate bisimulation error of the two neural networks, i.e., $d(\mathcal{N}_L, \mathcal{N}_S)$, we propose to merge the two neural networks in a non-fully connected structure $\mathcal{N}_M$, which is able to generate the output $\mathbf{y}_M^{\{M\}}$ exactly characterizing the difference of the outputs of $\mathcal{N}_L$ and $\mathcal{N}_S$, i.e., $\mathbf{y}_M^{\{M\}} = \mathbf{y}_L^{\{L\}} - \mathbf{y}_S^{\{S\}}$.

Merged Neural Network $\mathcal{N}_M$: To begin with, we consider two neural networks $\mathcal{N}_L$ and $\mathcal{N}_S$ with same input $\mathbf{x}^{\{0\}}$. We use $\mathbf{W}_M^{\{m\}}$ and $\mathbf{b}_M^{\{m\}}$ to denote the weight matrix and bias vector of the mth layer of the merged neural network $\mathcal{N}_M$, $\mathbf{x}_M^{\{m\}}$ and $\mathbf{y}_M^{\{m\}}$ are input and output vectors of mth layer of $\mathcal{N}_M$. The structure of the merged neural network $\mathcal{N}_M$ with $L+1$ layers is recursively defined as below:

$$\begin{cases} \mathbf{y}_M^{\{m\}} = \phi_M^{\{m\}}(\mathbf{W}_M^{\{m\}} \mathbf{x}_M^{\{m-1\}} + \mathbf{b}_M^{\{m\}}) \\ \mathbf{x}_M^{\{m\}} = \mathbf{y}_M^{\{m\}} \end{cases} \quad (12)$$

where $m = 1, 2, \ldots, L+1$. The input is $\mathbf{x}_M^{\{0\}} = \mathbf{x}^{\{0\}}$, output is $\mathbf{y}_M^{\{L+1\}}$, weight matrices $\mathbf{W}_M^{\{m\}}$ and bias vectors $\mathbf{b}_M^{\{m\}}$, and activation functions $\phi_M^{\{m\}}(\cdot)$ are categorized as the following five cases:

1) When $m = 1$, $\mathbf{W}_M^{\{1\}}$, $\mathbf{b}_M^{\{1\}}$, and $\phi_M^{\{1\}}(\cdot)$ are

$$\mathbf{W}_M^{\{1\}} = \begin{bmatrix} \mathbf{W}_L^{\{1\}} \\ \mathbf{W}_S^{\{1\}} \end{bmatrix} \quad (13)$$

$$\mathbf{b}_M^{\{1\}} = \begin{bmatrix} \mathbf{b}_L^{\{1\}} \\ \mathbf{b}_S^{\{1\}} \end{bmatrix} \quad (14)$$

$$\phi_M^{\{1\}}(\cdot) = \begin{bmatrix} \phi_L^{\{1\}}(\cdot) \\ \phi_S^{\{1\}}(\cdot) \end{bmatrix}. \quad (15)$$

2) When $1 < m \leq S-1$, $\mathbf{W}_M^{\{m\}}$, $\mathbf{b}_M^{\{m\}}$, and $\phi_M^{\{m\}}(\cdot)$ are

$$\mathbf{W}_M^{\{m\}} = \begin{bmatrix} \mathbf{W}_L^{\{m\}} & \mathbf{0}_{n_L^{\{m\}} \times n_S^{\{m-1\}}} \\ \mathbf{0}_{n_S^{\{m\}} \times n_L^{\{m-1\}}} & \mathbf{W}_S^{\{m\}} \end{bmatrix} \quad (16)$$

$$\mathbf{b}_M^{\{m\}} = \begin{bmatrix} \mathbf{b}_L^{\{m\}} \\ \mathbf{b}_S^{\{m\}} \end{bmatrix} \quad (17)$$

$$\phi_M^{\{m\}}(\cdot) = \begin{bmatrix} \phi_L^{\{m\}}(\cdot) \\ \phi_S^{\{m\}}(\cdot) \end{bmatrix}. \quad (18)$$

3) When $S-1 < m \leq L-1$, $\mathbf{W}_M^{\{m\}}$, $\mathbf{b}_M^{\{m\}}$, and $\phi_M^{\{m\}}(\cdot)$ are

$$\mathbf{W}_M^{\{m\}} = \begin{bmatrix} \mathbf{W}_L^{\{m\}} & \mathbf{0}_{n_L^{\{m\}} \times n_S^{\{S-1\}}} \\ \mathbf{0}_{n_S^{\{S-1\}} \times n_L^{\{m\}}} & \mathbf{I}_{n_S^{\{S-1\}}} \end{bmatrix} \quad (19)$$

$$\mathbf{b}_M^{\{m\}} = \begin{bmatrix} \mathbf{b}_L^{\{m\}} \\ \mathbf{0}_{n_S^{\{S-1\}} \times 1} \end{bmatrix} \quad (20)$$

$$\phi_M^{\{m\}}(\cdot) = \begin{bmatrix} \phi_L^{\{m\}}(\cdot) \\ \mathsf{purelin}(\cdot) \end{bmatrix}. \quad (21)$$

4) When $m = L$, $\mathbf{W}_M^{\{L\}}$, $\mathbf{b}_M^{\{L\}}$, and $\phi_M^{\{L\}}(\cdot)$ are

$$\mathbf{W}_M^{\{L\}} = \begin{bmatrix} \mathbf{W}_L^{\{L\}} & \mathbf{0}_{n_L^{\{L\}} \times n_S^{\{S-1\}}} \\ \mathbf{0}_{n_S^{\{S\}} \times n_L^{\{L-1\}}} & \mathbf{W}_S^{\{S\}} \end{bmatrix} \quad (22)$$

$$\mathbf{b}_M^{\{L\}} = \begin{bmatrix} \mathbf{b}_L^{\{L\}} \\ \mathbf{b}_S^{\{S\}} \end{bmatrix} \quad (23)$$

$$\phi_M^{\{L\}}(\cdot) = \begin{bmatrix} \phi_L^{\{L\}}(\cdot) \\ \phi_S^{\{S\}}(\cdot) \end{bmatrix}. \quad (24)$$

5) When $m = L+1$, $\mathbf{W}_M^{\{L+1\}}$, $\mathbf{b}_M^{\{L+1\}}$, and $\phi_M^{\{L+1\}}(\cdot)$ are

$$\mathbf{W}_M^{\{L+1\}} = \begin{bmatrix} \mathbf{I}_{n_L^{\{L\}}} & -\mathbf{I}_{n_L^{\{L\}}} \end{bmatrix} \quad (25)$$

$$\mathbf{b}_M^{\{L+1\}} = \begin{bmatrix} \mathbf{0}_{2n_L^{\{L\}} \times 1} \end{bmatrix} \quad (26)$$

$$\phi_M^{\{L+1\}}(\cdot) = \mathsf{purelin}(\cdot). \quad (27)$$

Remark 2: In the merging process of neural networks $\mathcal{N}_L$ and $\mathcal{N}_S$, (13)–(15) ensures that merged neural network $\mathcal{N}_M$ takes the one input $\mathbf{x}^{\{0\}}$ for the subsequent calls involving both processes of $\mathcal{N}_L$ and $\mathcal{N}_S$. Then, for $1 < m \leq S-1$, $\mathcal{N}_M$ conducts the computation of $\mathcal{N}_L$ and $\mathcal{N}_S$ parallelly for the hidden layers of $1 < m \leq S-1$. When $S-1 < m \leq L-1$, the hidden layers of neural network $\mathcal{N}_S$ which have less hidden layers are expanded to match the number of layers of neural network $\mathcal{N}_L$ with a larger number of hidden layers, but the expanded layers are forced to pass the information to subsequent layers without any changes, i.e., the weight matrices of the expanded hidden layers are identity matrices, and the bias vector is the zero vectors. This expansion is formalized as (19)–(21). Moreover, as $m = L$, this layer is a combination of output layers of both $\mathcal{N}_L$ and $\mathcal{N}_S$ to generate the same outputs of $\mathcal{N}_L$ and $\mathcal{N}_S$. At last, a comparison layer $L+1$ is added to compute the exact difference between two bisimular neural networks.

With the merged neural network $\mathcal{N}_M$ in the description of (12)–(27), we are ready to propose the main contribution of this work in Proposition 1.

Proposition 1: Given two neural networks $\mathcal{N}_L$ with L layers and $\mathcal{N}_S$ with S layers under Assumption 1, the output $\mathbf{y}_M^{\{L+1\}}$ of their merged neural network $\mathcal{N}_M$ defined by (12)–(27) equals the difference of the output $\mathbf{y}_L^{\{L\}}$ of $\mathcal{N}_L$ and the output $\mathbf{y}_S^{\{S\}}$ of $\mathcal{N}_S$, i.e.,

$$\mathbf{y}_M^{\{L+1\}} = \mathbf{y}_L^{\{L\}} - \mathbf{y}_S^{\{S\}} \quad (28)$$

Authorized licensed use limited to: Augusta University. Downloaded on March 02,2023 at 16:50:11 UTC from IEEE Xplore. Restrictions apply.

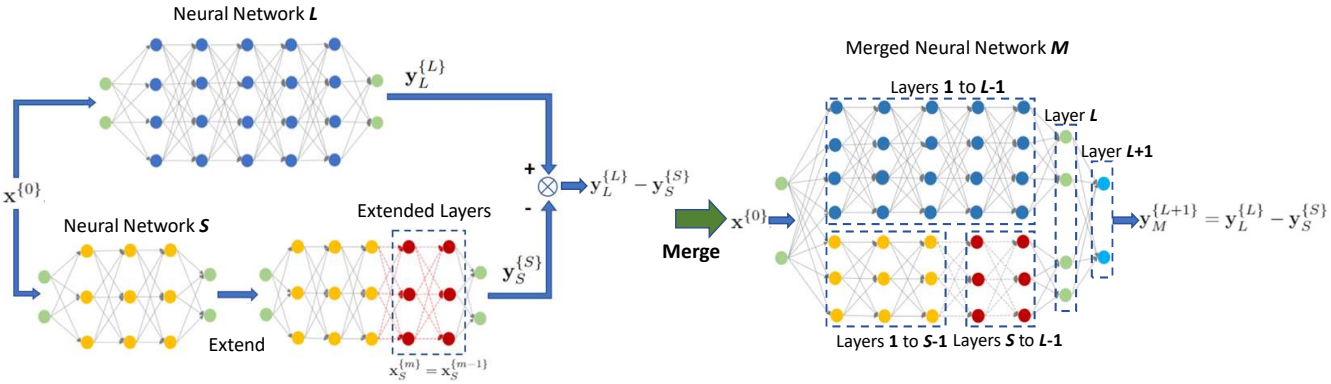

Fig. 1. Neural network merging process for approximate bisimulation error computation

holds for any input $\mathbf{x}^{\{0\}} \in \mathcal{X}$, where $\mathbf{y}_L^{\{L\}} = \Phi_L(\mathbf{x}^{\{0\}})$ and $\mathbf{y}_S^{\{S\}} = \Phi_S(\mathbf{x}^{\{0\}})$.

Proof: Considering an input $\mathbf{x}^{\{0\}} \in \mathcal{X}$ and according to (13)–(15), the following results for output of layer $m = 1$ of merged neural network $\mathcal{N}_M$ can be obtained

$$\mathbf{y}_M^{\{1\}} = \begin{bmatrix} \phi_L^{\{1\}}(\mathbf{W}_L^{\{1\}}\mathbf{x}^{\{0\}} + \mathbf{b}_L^{\{1\}}) \\ \phi_S^{\{1\}}(\mathbf{W}_S^{\{1\}}\mathbf{x}^{\{0\}} + \mathbf{b}_S^{\{1\}}) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_L^{\{1\}} \\ \mathbf{x}_S^{\{1\}}. \end{bmatrix} \tag{29}$$

Further considering layers $1 < m \leq S-1$ of $\mathcal{N}_M$, and using (16) and (17), it leads to

$$\mathbf{W}_M^{\{m\}}\mathbf{x}_M^{\{m-1\}} + \mathbf{b}_M^{\{m\}} = \begin{bmatrix} \mathbf{W}_L^{\{m\}}\mathbf{x}_L^{\{m-1\}} + \mathbf{b}_L^{\{m\}} \\ \mathbf{W}_S^{\{m\}}\mathbf{x}_S^{\{m-1\}} + \mathbf{b}_S^{\{m\}} \end{bmatrix} \tag{30}$$

where $1 < m \leq S-1$. Then based on (18), recursively we can obtain

$$\mathbf{y}_M^{\{S-1\}} = \begin{bmatrix} \phi_L^{\{S-1\}} \circ \cdots \circ \phi_L^{\{1\}}(\mathbf{W}_L^{\{1\}}\mathbf{x}^{\{0\}} + \mathbf{b}_L^{\{1\}}) \\ \phi_S^{\{S-1\}} \circ \cdots \circ \phi_S^{\{1\}}(\mathbf{W}_S^{\{1\}}\mathbf{x}^{\{0\}} + \mathbf{b}_S^{\{1\}}) \end{bmatrix} \tag{31}$$

$$= \begin{bmatrix} \mathbf{x}_L^{\{S-1\}} \\ \mathbf{x}_S^{\{S-1\}} \end{bmatrix}. \tag{32}$$

Moreover, considering $S-1 < m \leq L-1$ and using (19) and (20), one can obtain

$$\mathbf{W}_M^{\{m\}}\mathbf{x}_M^{\{m-1\}} + \mathbf{b}_M^{\{m\}} = \begin{bmatrix} \mathbf{W}_L^{\{m\}}\mathbf{x}_L^{\{m-1\}} + \mathbf{b}_L^{\{m\}} \\ \mathbf{x}_S^{\{m-1\}} \end{bmatrix} \tag{33}$$

where $S-1 < m \leq L-1$. From (21), it yields

$$\mathbf{y}_M^{\{L-1\}} = \begin{bmatrix} \mathbf{x}_L^{\{L-1\}} \\ \mathbf{x}_S^{\{S-1\}} \end{bmatrix}$$

in which $\mathbf{x}_L^{\{L-1\}}$ is defined as

$$\begin{aligned} \mathbf{x}_L^{\{L-1\}} &= \phi_L^{\{L-1\}} \circ \cdots \circ \phi_L^{\{S\}}(\mathbf{W}_L^{\{S\}}\mathbf{x}_L^{\{S-1\}} + \mathbf{b}_L^{\{S\}}) \\ &= \phi_L^{\{L-1\}} \circ \cdots \circ \phi_L^{\{1\}}(\mathbf{W}_L^{\{1\}}\mathbf{x}_L^{\{0\}} + \mathbf{b}_L^{\{1\}}). \end{aligned} \tag{34}$$

Then, as $m = L$ with (22) and (23) as well as $\mathbf{x}_M^{\{L-1\}} = \mathbf{y}_M^{\{L-1\}}$, it leads to

$$\mathbf{W}_M^{\{L\}}\mathbf{x}_M^{\{L-1\}} + \mathbf{b}_M^{\{L\}} = \begin{bmatrix} \mathbf{W}_L^{\{L\}}\mathbf{x}_L^{\{L-1\}} + \mathbf{b}_L^{\{L\}} \\ \mathbf{W}_S^{\{S\}}\mathbf{x}_S^{\{S-1\}} + \mathbf{b}_S^{\{S\}} \end{bmatrix} \tag{35}$$

Also due to (24), we can have

$$\mathbf{y}_M^{\{L\}} = \begin{bmatrix} \phi_L^{\{L\}}(\mathbf{W}_L^{\{L\}}\mathbf{x}_L^{\{L-1\}} + \mathbf{b}_L^{\{L\}}) \\ \phi_S^{\{S\}}(\mathbf{W}_S^{\{S\}}\mathbf{x}_S^{\{S-1\}} + \mathbf{b}_S^{\{S\}}) \end{bmatrix} = \begin{bmatrix} \mathbf{y}_L^{\{L\}} \\ \mathbf{y}_S^{\{S\}} \end{bmatrix}. \tag{36}$$

At last, when $m = L+1$ with (25)–(27), the following result can be obtained

$$\begin{aligned} \mathbf{y}_M^{\{L+1\}} &= \mathbf{W}_M^{\{L+1\}}\mathbf{x}_L^{\{M\}} + \mathbf{b}_M^{\{L+1\}} \\ &= \begin{bmatrix} \mathbf{I}_{n_L^{\{L\}}} & -\mathbf{I}_{n_L^{\{L\}}} \end{bmatrix} \begin{bmatrix} \mathbf{y}_L^{\{L\}} \\ \mathbf{y}_S^{\{S\}} \end{bmatrix} \\ &= \mathbf{y}_L^{\{L\}} - \mathbf{y}_S^{\{S\}}. \end{aligned} \tag{37}$$

where $\mathbf{y}_L^{\{L\}} = \Phi_L(\mathbf{x}^{\{0\}})$ and $\mathbf{y}_S^{\{S\}} = \Phi_S(\mathbf{x}^{\{0\}})$. The proof is complete. ■

Proposition 1 implies that, for any individual input $\mathbf{x}^{\{0\}}$, we can compute the difference of the outputs between two bisimilar neural networks via generating the output of their merged neural network of $\mathbf{x}^{\{0\}}$. This lays the foundation of computing the approximate bisimulation error in the description of (11), i.e., the computation of the maximum discrepancy between two bisimilar neural networks subject to an input set $\mathcal{X}$ can be converted to the output reachable set $\mathcal{Y}_M$ computation of merged neural network $\mathcal{N}_M$.

Proposition 2: Given an input set $\mathcal{X}$, two neural networks $\mathcal{N}_L$ with L layers and $\mathcal{N}_S$ with S layers under Assumption 1, their merged neural network $\mathcal{N}_M$ can be defined by (12)–(27). Then, the approximate bisimulation error between $\mathcal{N}_L$ and $\mathcal{N}_S$ can be computed by

$$d(\mathcal{N}_L, \mathcal{N}_S) = \sup_{\mathbf{y}_M^{\{L+1\}} \in \mathcal{Y}_M} \left\| \mathbf{y}_M^{\{L+1\}} \right\| \tag{38}$$

where $\mathbf{y}_M^{\{L+1\}} = \Phi_M(\mathbf{x}^{\{0\}})$ is the output of $\mathcal{N}_M$ and $\mathcal{Y}_M$ is the output reachable set of $\mathcal{N}_M$.

Proof: The result can be obtained straightforwardly from the result in Proposition 1, i.e., $\mathbf{y}_M^{\{L+1\}} = \mathbf{y}_L^{\{L\}} - \mathbf{y}_S^{\{S\}}$. The proof is complete. ■

As shown in Proposition 2, the key of computing $d(\mathcal{N}_L, \mathcal{N}_S)$ is to compute the output reachable set $\mathcal{Y}_M$. For instance, as in NNV neural network reachability analysis tool, the reachable sets are in the form of a family of polyhedral sets [12], and in IGNNV tool, the output reachable

Authorized licensed use limited to: Augusta University. Downloaded on March 02,2023 at 16:50:11 UTC from IEEE Xplore. Restrictions apply.

set is a family of interval sets [13], [14]. With the reachable set $\mathcal{Y}_M$, the approximate bisimulation error $d(\mathcal{N}_L, \mathcal{N}_S)$ can be easily obtained by searching for the maximal value of $\left\|\mathbf{y}_M^{\{L+1\}}\right\|$ in $\mathcal{Y}_M$, e.g., testing throughout a finite number of vertices in polyhedral sets.

IV. Application to Assured Neural Network Compression

A. Assured Neural Network Compression

In practical applications, neural networks are usually large in size, and it could be computationally expensive and time-consuming to perform those tasks requiring a large amount of computation resources. A promising method to mitigate the computation burden is to compress large-scale neural networks into small-scale ones and provide the approximate bisimulation error between two neural networks. With the approximate bisimulation error, we can infer the outputs of the original large-scale neural network via running its corresponding small-scale compressed one plus the approximate bisimulation error. The assured neural network compression is stated as below.

Definition 6: Given a large-scale neural network $\mathcal{N}_L$ with input set $\mathcal{X}$, a small-scale neural network $\mathcal{N}_S$ is called its assured compressed version with precision ε if the approximate bisimulation error of two neural networks are not greater than ε, i.e.,

$$d(\mathcal{N}_L, \mathcal{N}_S) \leq \varepsilon \tag{39}$$

where $\varepsilon \geq 0$.

Remark 3: There exist a number of neural network compression methods [9] to obtain small-scale neural network $\mathcal{N}_S$. In this paper, our focus is on how to compute the assured neural network compression precision ε using the framework of approximate bisimulation relations proposed in the previous sections.

Example 1: We verify the effectiveness of the approximate bisimulation approach in neural network compression by a numerical case. In the numerical case, we aim to soundly simulate a neural network $\mathcal{N}_L$ (large-scale) with 5 hidden layers and 50 neurons in each hidden layer using a neural network $\mathcal{N}_S$ (small-scale) with 2 hidden layers and 10 neurons in each hidden layer. To facilitate the visualization of the simulation results, the output of both neural networks is selected one-dimensional.

First, a neural network $\mathcal{N}_L$ is randomly generated, and then a neural network $\mathcal{N}_S$ is trained out of the input-output data of $\mathcal{N}_L$. All activation functions are ReLU functions. Using the merged neural network method and computing reachable set with NNV tool, the approximate bisimulation error $\varepsilon = 26.1227$ of the two neural networks can be obtained. With the help of $\varepsilon = 26.1227$, the upper and lower bounds of output $\mathbf{y}_L^{\{L\}}$ of $\mathcal{N}_L$ can be obtained via the outputs $\mathbf{y}_S^{\{S\}}$ of $\mathcal{N}_S$ with a smaller size, i.e., upper bound $\overline{\mathbf{y}}_L^{\{L\}} = \mathbf{y}_S^{\{S\}} + \varepsilon$ and lower bound $\underline{\mathbf{y}}_L^{\{L\}} = \mathbf{y}_S^{\{S\}} - \varepsilon$.

Output data of the original neural network and the compressed neural network, as well as the upper and lower bounds, are represented in Fig. 2. It can be observed that all the outputs $\mathbf{y}_L^{\{L\}}$ are within the upper bound $\overline{\mathbf{y}}_L^{\{L\}}$ and lower bound $\underline{\mathbf{y}}_L^{\{L\}}$, i.e., $\underline{\mathbf{y}}_L^{\{L\}} \leq \mathbf{y}_L^{\{L\}} \leq \overline{\mathbf{y}}_L^{\{L\}}$.

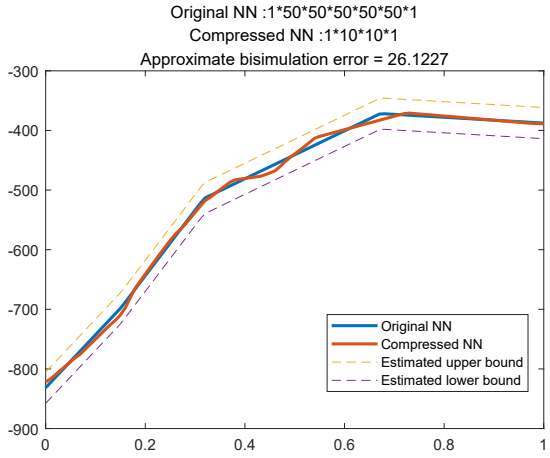

Fig. 2. Assured compression for a random neural network (from $50 \times 50 \times 50 \times 50 \times 50$ to 10×10) by approximate bisimulation approach.

B. Application of ACAS Xu Network Verification

In this subsection, we apply the neural network model compression method to ACAS Xu network in [11] to accelerate the verification processes. ACAS Xu system has been developed using a large lookup table that maps sensor measurements to warning signals, see Fig. 3. It has been shown that DNNs can significantly reduce memory (replacing a 2GB lookup table with an efficient DNNs of less than 3MB). The DNN method of ACAS Xu system consists of 45 DNNs, and each neural network contains 5 inputs and 5 outputs, with 6 hidden layers and 50 neurons with ReLU activation functions in each layer.

In practical applications, calculating the exact output reachable set of a neural network with 6 hidden layers and 50 neurons per layer requires huge computational effort and computational time [15]. Therefore, we compress the original neural networks into smaller neural networks and compute the assured precision by the approximate bisimulation method. Then, we can perform verification of properties based on those reduced-scale neural networks and approximate bisimulation error ε, i.e., expand the unsafe region $\neg\mathcal{S}$ in Definition 3 by the approximate bisimulation error ε.

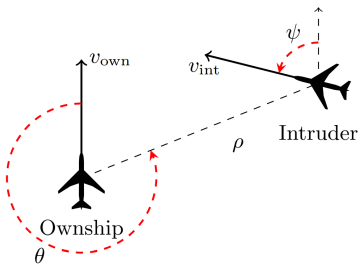

Fig. 3. ACAS Xu horizontal logic table illustration [15]

Authorized licensed use limited to: Augusta University. Downloaded on March 02,2023 at 16:50:11 UTC from IEEE Xplore. Restrictions apply.

TABLE I

PROPERTY ϕ_3 VERIFICATION FOR ACAS XU SYSTEM

ID	ε	$T_L(s)$	$T_S(s)$	V_L	V_S
N_{11}	0.0927	463.24804	0.19383	Safe	Uncertain
N_{12}	0.089	504.08039	0.26257	Safe	Uncertain
N_{13}	0.0369	185.89549	0.66355	Safe	Uncertain
N_{14}	0.0041	29.31453	0.34665	Safe	Safe
N_{15}	0.0026	45.7813	0.41446	Safe	Safe
N_{16}	0.0013	12.17051	0.2766	Safe	Safe
N_{17}	0.0018	3.22305	0.74309	Unsafe	Uncertain
N_{18}	0.0067	2.53016	0.50254	Unsafe	Uncertain
N_{19}	0.0056	3.33068	0.50024	Unsafe	Uncertain
N_{21}	0.1838	151.38468	1.2967	Safe	Uncertain
N_{22}	0.1143	56.81178	0.87974	Safe	Uncertain
N_{23}	0.018	92.08281	0.66704	Safe	Safe
N_{24}	0.0035	3.14713	0.30876	Safe	Safe
N_{25}	0.0031	19.24327	0.42653	Safe	Safe
N_{26}	0.0161	2.77801	0.2835	Safe	Safe
N_{27}	0.0047	9.83793	0.35039	Safe	Safe
N_{28}	0.0063	2.87251	0.39635	Safe	Safe
N_{29}	0.0022	1.51099	0.23274	Safe	Safe
N_{31}	0.0244	63.11602	1.19615	Safe	Safe
N_{32}	0.0907	421.81782	0.86584	Safe	Uncertain
N_{33}	0.0254	94.0685	0.19859	Safe	Safe
N_{34}	0.0055	24.4508	0.38036	Safe	Safe
N_{35}	0.002	8.88554	0.20696	Safe	Safe
N_{36}	0.0135	18.18405	0.26895	Safe	Safe
N_{37}	0.0136	1.25423	0.39768	Safe	Safe
N_{38}	0.0061	5.36596	0.15807	Safe	Safe
N_{39}	0.0055	11.92655	0.68403	Safe	Safe

In this example, we use neural networks with two hidden layers and 10 neurons in each layer as the compressed version for the compression of the DNNs of the ACAS Xu system. Then, we verify Property ϕ_3 on 27 neural networks in the ACAS Xu system using their assured compressed versions. The verification results and computational time are listed in Table I. In Table I, ε is the approximate bisimulation error. T_L is the verification time (seconds) using original neural networks and T_S is the verification time (seconds) using compressed neural networks. V_L is the verification results on original neural networks, and V_S is the verification results on compressed neural networks.

As explicitly shown in Table I, the verification time can be significantly reduced using compressed neural networks. It is worth mentioning that since the approximate bisimulation error is an over-approximation of the exact difference between the outputs of two neural networks, the safety conclusions based on compressed networks are only able to derive safe conclusions for original networks in safe cases. As to uncertain cases, we have to perform verification on original neural networks to ascertain the safety property. It can be found that the safety of 18 of the compressed neural networks can be used to conclude the safety of original neural networks. The remaining 9 unsafe verification results based on compressed neural networks are insufficient to derive safe or unsafe conclusions of original neural networks. This is mainly because the approximate bisimulation error is too large to meet the accuracy of the safety verification. Despite the 9 uncertain cases that need to be verified through original neural networks, the total verification time has been significantly reduced for these 27 neural networks.

V. CONCLUSION

This work proposed approximate bisimulation relations for feedforward neural networks. The approximate bisimulation relation formally defines the maximal difference between the outputs of two bisimular neural networks from the same inputs. A reachability-based computation procedure is developed to compute the approximation error via a novel neural network merging approach. Then, the approximation bismulation approach is applied to assured neural network compression. With the approximate bisimulation error, the perform tasks using the compressed network on behalf of the original one such as verification of neural networks, which has been demonstrated by an ACAS Xu example.

REFERENCES

[1] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017.

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

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

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

[5] 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, 2017.

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

[7] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban, T. Marinč, D. Neumann, T. Nguyen, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek, "DeepCABAC: A universal compression algorithm for deep neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 700–714, 2020.

[8] Y. Zhang, W. Ding, and C. Liu, "Summary of convolutional neural network compression technology," in *2019 IEEE International Conference on Unmanned Systems (ICUS)*, pp. 480–483, 2019.

[9] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[11] M. P. Owen, A. Panken, R. Moss, L. Alvarez, and C. Leeper, "ACAS Xu: Integrated collision avoidance and detect and avoid capability for UAS," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pp. 1–10, IEEE, 2019.

[12] H.-D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *International Conference on Computer Aided Verification*, pp. 3–17, Springer, 2020.

[13] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5777–5783, 2018.

[14] W. Xiang, H.-D. Tran, X. Yang, and T. T. Johnson, "Reachable set estimation for neural network control systems: A simulation-guided approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 5, pp. 1821–1830, 2021.

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

Authorized licensed use limited to: Augusta University. Downloaded on March 02,2023 at 16:50:11 UTC from IEEE Xplore. Restrictions apply.