2022 IEEE 61st Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

Safety Verification of Neural Network Control Systems Using Guaranteed Neural Network Model Reduction

Weiming Xiang and Zhongzhu Shao

2022 IEEE 61st Conference on Decision and Control (CDC) | 978-1-6654-6761-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDC51059.2022.9992984

***Abstract*— This paper aims to enhance the computational efficiency of safety verification of neural network control systems by developing a guaranteed neural network model reduction method. First, a concept of model reduction precision is proposed to describe the guaranteed distance between the outputs of a neural network and its reduced-size version. A reachability-based algorithm is proposed to accurately compute the model reduction precision. Then, by substituting a reduced-size neural network controller into the closed-loop system, an algorithm to compute the reachable set of the original system is developed, which is able to support much more computationally efficient safety verification processes. Finally, the developed methods are applied to a case study of the Adaptive Cruise Control system with a neural network controller, which is shown to significantly reduce the computational time of safety verification and thus validate the effectiveness of the method.**

I. Introduction

Neural networks are currently widely used in various fields, such as image processing [1], pattern recognition [2], adaptive control [3], unmanned vehicles [4] and aircraft collision avoidance systems [5], etc., demonstrating their powerful capabilities in solving complex and challenging problems that traditional approaches fail to address. As neural networks are further investigated, the size and complexity of their models continue to increase in order to improve their performance and accuracy to cope with complex and difficult tasks and changing environments. However, more complex large-scale neural network models also imply larger computational resources, such as larger memory, higher computational power and more energy consumption in applications [6]. As a result, many neural network model reduction methods have been developed, such as parameter pruning and sharing, low-rank factorization, transfer/compact convolution filters, and knowledge distillation [7]. More results on neural network model reduction can be found in a recent survey [8].

On the other hand, due to the black-box nature of neural networks, neural networks are vulnerable in the face of resistance to interference/attacks. It has been observed that neural networks trained on large amounts of data are sometimes sensitive to updates and react to even small changes in parameters in unexpected and incorrect ways [9]. When neural networks are applied as controllers onto dynamical systems, they will inevitably suffer from safety problems due to the inevitable disturbances and uncertainties in the control process, further affecting the stability and safety of the whole closed-loop system. Therefore, when integrating neural networks into safety-critical control systems, the safety of the neural network needs to be guaranteed at all times, i.e., the safety verification of the neural network needs to be implemented. However, due to the sensitivity of neural networks to perturbations and the complex structure of neural networks, the verification of neural networks is extremely difficult. It has been demonstrated that the verification of simple properties of a small-scale neural network is an uncertainty polynomial (NP) complete problem [10]. A few results have been reported in the literature for the formal verification of systems consisting of neural networks, readers are referred to the recent survey [11]. Specifically, reachability analysis is one of the promising safety verification tools such as in [12]–[16], a simulation-based approach is proposed that transforms the difficulty of over-approximating the neural network's output set into a problem of estimating the neural network's maximal sensitivity, which is formulated as a series of convex optimization problems [13], [14]. Polytope-operation-based approaches were developed in [12], [15], [16] for dealing with a class of neural networks with activation functions of Rectified Linear Units (ReLU). However, the scalability issue is the major barrier preventing applying these methods to large-scale neural networks as well as neural network control systems active in a long period of time which means a large amount of reachable set computation is required during the time of interest.

In this paper, we propose a guaranteed model reduction method for neural network controllers based on the neural network reachability analysis and apply it to enhance the scalability of the reachability-based safety verification of closed-loop systems. Firstly, a concept of model reduction precision is proposed to accurately measure the distance between the outputs of an original neural network and its reduced-size version, and an approach to compute the model reduction precision is proposed, which ensures that the difference between the outputs obtained from two neural networks for a given input interval, chosen with any identical input, is within the model reduction precision. This algorithm is then applied to the model reduction of the neural network control system, enabling computationally efficient verification processes based on the reduced-size neural network controller. Finally, the correctness and feasibility of our approach are verified by applying it to the safety verification through the Adaptive Cruise Control (ACC) case study.

The remainder of the paper is organized as follows: Preliminaries are given in Section II. The guaranteed model

This research was supported by the National Science Foundation, under NSF CAREER Award 2143351 and NSF CNS Award 2223035.

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

Zhongzhu Shao is with School of Electrical Engineering, Southwest Jiaotong University, China.

reduction of neural networks is presented in Section III. The reachable set computation and safety verification algorithm for the neural network control system are presented in Section IV. The evaluation on the adaptive cruise control system is given in Section V. The conclusion is given in Section VI.

II. Preliminaries

In this paper, we consider a class of continuous-time nonlinear systems in the form of

$$\begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) = h(\mathbf{x}(t)) \end{cases} \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ is the state vector, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the control input and the $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ is the output vector. In general, the control input is in the form of

$$\mathbf{u}(t) = \gamma(\mathbf{y}(t), \mathbf{r}(t), t) \tag{2}$$

where $\mathbf{r}(t) \in \mathbb{R}^{n_r}$ is the reference input for the controller.

To avoid the difficulties in the controller design when system models are complex or even unavailable, one effective method is to use input-output data to train neural networks capable of generating appropriate control input signals to achieve control objectives. The neural network controller is in the form of

$$\mathbf{u}(t) = \Phi(\mathbf{y}(t), \mathbf{r}(t)) \tag{3}$$

where Φ denotes the neural network mapping output and reference signals to control input.

In actual applications, the neural network receives input and generates output in a fraction of the computation time, so the control input generated by the neural network is generally discrete, generated only at each sampling time point $t_k, k \in \mathbb{N}$, and then remains a constant value between two successive sampling time instants. Therefore, the continuous-time nonlinear dynamical system with a neural network controller with sampling actions can be expressed in the following form of

$$\begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \Phi(\boldsymbol{\tau}(t_k))) \\ \mathbf{y}(t) = h(\mathbf{x}(t)) \end{cases}, \quad t \in [t_k, t_{k+1}) \tag{4}$$

where $\boldsymbol{\tau}(t_k) = [\mathbf{y}^\top(t_k), \mathbf{r}^\top(t_k)]^\top$.

In this work, we consider feedforward neural networks for controllers in the form of $\Phi : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ defined by the following recursive equations in the form of

$$\begin{cases} \boldsymbol{\eta}_\ell = \phi_\ell(\mathbf{W}_\ell \boldsymbol{\eta}_{\ell-1} + \mathbf{b}_\ell), \ \ell = 1, \ldots, L \\ \boldsymbol{\eta}_L = \Phi(\boldsymbol{\eta}_0) \end{cases} \tag{5}$$

where $\boldsymbol{\eta}_\ell$ denotes the output of the ℓ-th layer of the neural network, and in particular $\boldsymbol{\eta}_0 \in \mathbb{R}^{n_0}$ is the input to the neural network and $\boldsymbol{\eta}_L \in \mathbb{R}^{n_L}$ is the output produced by the neural network, respectively. $\mathbf{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $\mathbf{b}_\ell \in \mathbb{R}^{n_\ell}$ are weight matrices and bias vectors for the ℓ-th layer. $\phi_\ell = [\psi_\ell, \cdots, \psi_\ell]$ is the concatenation of activation functions of the ℓ-th layer in which $\psi_\ell : \mathbb{R} \to \mathbb{R}$ is the activation function.

In this paper, we aim at reducing the computational cost of safety verification of neural network control systems in the framework of reachable set computation.

Definition 1: Given a neural network in the form of (5) and an input set $\mathcal{U}$, the following set

$$\mathcal{Y} = \{\boldsymbol{\eta}_L \in \mathbb{R}^{n_L} \mid \boldsymbol{\eta}_L = \Phi(\boldsymbol{\eta}_0), \ \boldsymbol{\eta}_0 \in \mathcal{U}\} \tag{6}$$

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

Definition 2: A set $\mathcal{Y}_e$ is called an output reachable set over-approximation of neural network (5), if $\mathcal{Y} \subseteq \mathcal{Y}_e$ holds, where $\mathcal{Y}$ is the output reachable set of neural network (5).

Definition 3: Given a neural network control system in the form of (1) and (3) with initial set $\mathcal{X}_0$ and input set $\mathcal{V}$, the reachable set at time t is

$$\mathcal{R}(t) = \{\mathbf{x}(t; \mathbf{x}_0, \mathbf{r}(\cdot)) \in \mathbb{R}^{n_x} \mid \mathbf{x}_0 \in \mathcal{X}_0, \ \mathbf{r}(t) \in \mathcal{V}\} \tag{7}$$

and the union of $\mathcal{R}(t)$ over $[t_0, t_f]$ defined by

$$\mathcal{R}([t_0, t_f]) = \bigcup\nolimits_{t \in [t_0, t_f]} \mathcal{R}(t) \tag{8}$$

is the reachable set over time interval $[t_0, t_f]$.

Definition 4: A set $\mathcal{R}_e(t)$ is an over-approximation of $\mathcal{R}(t)$ at time t if $\mathcal{R}(t) \subseteq \mathcal{R}_e(t)$ holds. Moreover, $\mathcal{R}_e([t_0, t_f]) = \bigcup_{t \in [t_0, t_f]} \mathcal{R}_e(t)$ is an over-approximation of $\mathcal{R}([t_0, t_f])$ over time interval $[t_0, t_f]$.

Definition 5: Safety specification $\mathcal{S}$ formalizes the safety requirements for state $\mathbf{x}(t)$ of neural network control system (1), and is a predicate over state $\mathbf{x}(t)$ of neural network control system (1). The neural network control system (1) is safe over time interval $[t_0, t_f]$ if the following condition is satisfied:

$$\mathcal{R}_e([t_0, t_f]) \cap \neg\mathcal{S} = \emptyset \tag{9}$$

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

As indicated in [12]–[16], the computation cost for reachable set computation heavily relies on the size of neural networks, i.e., numbers of layers and neurons. In this paper, we aim to reduce the size of the neural network controller and rigorously compute the model reduction error, i.e., guaranteed neural network model reduction, so that the reachable set computation can be efficiently performed on a significantly reduced-size neural network and then mapped back to the original neural network to reach safety verification conclusions.

III. Guaranteed Neural Network Model Reduction

Given a large-scale neural network Φ, there exist a large number of neural network model reduction methods as in survey paper [8] to obtain its reduced-size version $\hat{\Phi}$ as below:

$$\begin{cases} \hat{\boldsymbol{\eta}}_\ell = \hat{\phi}_\ell(\hat{\mathbf{W}}_\ell \hat{\boldsymbol{\eta}}_{\ell-1} + \hat{\mathbf{b}}_\ell), \ \ell = 1, \ldots, \hat{L} \\ \hat{\boldsymbol{\eta}}_L = \hat{\Phi}(\hat{\boldsymbol{\eta}}_0) \end{cases} \tag{10}$$

To enable guaranteed neural network model reduction, the key is how to rigorously compute the output difference between the original neural network Φ and its reduced-size version $\hat{\Phi}$. Without loss of generality, the following

assumption is given for neural network Φ and its reduced-size version $\hat{\Phi}$.

Assumption 1: The following assumptions hold for neural network Φ and its reduced-size version $\hat{\Phi}$:

1) The number of inputs of two neural networks are the same, i.e., $n_0 = \hat{n}_0$;
2) The number of outputs of two neural networks are the same, i.e., $n_L = n_{\hat{L}}$;
3) The number of hidden layers of neural network Φ is greater than or equal to the number of hidden layers of neural network $\hat{\Phi}$, i.e., $L \geq \hat{L}$.

To characterize the output difference between Φ and $\hat{\Phi}$, we define the following metric for model reduction precision.

Definition 6: Consider neural network Φ and its reduced-size version $\hat{\Phi}$ with one same input set $\mathcal{U}$ and their corresponding output sets $\mathcal{Y}$ and $\hat{\mathcal{Y}}$, we define the distance between the outputs of Φ and $\hat{\Phi}$ with respect to the input set $\mathcal{U}$ by

$$\rho(\Phi, \hat{\Phi}, \mathcal{U}) = \sup_{\boldsymbol{\eta}_0 = \hat{\boldsymbol{\eta}}_0, \boldsymbol{\eta}_0, \hat{\boldsymbol{\eta}}_0 \in \mathcal{U}} \left\| \Phi(\boldsymbol{\eta}_0) - \hat{\Phi}(\hat{\boldsymbol{\eta}}_0) \right\| \tag{11}$$

where $\rho(\Phi, \hat{\Phi}, \mathcal{U})$ is called model reduction precision.

In the framework of reachability analysis of neural networks, the following theorem presents a numerically tractable method to compute model reduction precision $\rho(\Phi, \hat{\Phi}, \mathcal{U})$.

Theorem 1: Given neural network Φ and its reduced-size version $\hat{\Phi}$ with input set $\mathcal{U}$, the model reduction precision $\rho(\Phi, \hat{\Phi}, \mathcal{U})$ can be computed by

$$\rho(\Phi, \hat{\Phi}, \mathcal{U}) = \sup_{\tilde{\boldsymbol{\eta}}_0 \in \mathcal{U}} \left\| \tilde{\Phi}(\tilde{\boldsymbol{\eta}}_0) \right\| \tag{12}$$

where neural network $\tilde{\Phi}$ is an augmented neural network of Φ and $\hat{\Phi}$ defined as follows:

$$\begin{cases} \tilde{\boldsymbol{\eta}}_\ell = \tilde{\phi}_\ell(\tilde{\mathbf{W}}_\ell \tilde{\boldsymbol{\eta}}_{\ell-1} + \tilde{\mathbf{b}}_\ell), \ \ell = 1, \ldots, L+1 \\ \tilde{\boldsymbol{\eta}}_{L+1} = \tilde{\Phi}(\tilde{\boldsymbol{\eta}}_0) \end{cases} \tag{13}$$

in which input $\tilde{\boldsymbol{\eta}}_0 = \boldsymbol{\eta}_0 = \hat{\boldsymbol{\eta}}_0$ and

$$\tilde{\mathbf{W}}_\ell = \begin{cases} \begin{bmatrix} \mathbf{W}_1 \\ \hat{\mathbf{W}}_1 \end{bmatrix}, & \ell = 1 \\ \begin{bmatrix} \mathbf{W}_\ell & \mathbf{0}_{n_\ell \times \hat{n}_{\ell-1}} \\ \mathbf{0}_{\hat{n}_\ell \times n_{\ell-1}} & \hat{\mathbf{W}}_\ell \end{bmatrix}, & 1 < \ell \leq \hat{L} - 1 \\ \begin{bmatrix} \mathbf{W}_\ell & \mathbf{0}_{n_\ell \times \hat{n}_{\hat{L}-1}} \\ \mathbf{0}_{\hat{n}_{\hat{L}-1} \times n_\ell} & \mathbf{I}_{\hat{n}_{\hat{L}-1}} \end{bmatrix}, & \hat{L} \leq \ell \leq L - 1 \\ \begin{bmatrix} \mathbf{W}_L & \mathbf{0}_{n_L \times n_{\hat{L}-1}} \\ \mathbf{0}_{n_{\hat{L}} \times n_{L-1}} & \hat{\mathbf{W}}_{\hat{L}} \end{bmatrix}, & \ell = L \\ \begin{bmatrix} \mathbf{I}_{n_L} & -\mathbf{I}_{n_{\hat{L}}} \end{bmatrix}, & \ell = L+1 \end{cases} \tag{14}$$

$$\tilde{\mathbf{b}}_\ell = \begin{cases} \begin{bmatrix} \mathbf{b}_\ell \\ \hat{\mathbf{b}}_\ell \end{bmatrix}, & 1 \leq \ell \leq \hat{L} - 1 \\ \begin{bmatrix} \mathbf{b}_\ell \\ \mathbf{0}_{\hat{n}_{\hat{L}-1} \times 1} \end{bmatrix}, & \hat{L} \leq \ell \leq L - 1 \\ \begin{bmatrix} \mathbf{b}_L \\ \hat{\mathbf{b}}_{\hat{L}} \end{bmatrix}, & \ell = L \\ \begin{bmatrix} \mathbf{0}_{(n_L + n_{\hat{L}}) \times 1} \end{bmatrix}, & \ell = L+1 \end{cases} \tag{15}$$

$$\tilde{\phi}_\ell(\cdot) = \begin{cases} \begin{bmatrix} \phi_\ell(\cdot) \\ \hat{\phi}_\ell(\cdot) \end{bmatrix}, & 1 \leq \ell \leq \hat{L} - 1 \\ \begin{bmatrix} \phi_\ell(\cdot) \\ \mathsf{purelin}(\cdot) \end{bmatrix}, & \hat{L} \leq \ell \leq L - 1 \\ \begin{bmatrix} \phi_L(\cdot) \\ \hat{\phi}_{\hat{L}}(\cdot) \end{bmatrix}, & \ell = L \\ \mathsf{purelin}(\cdot), & \ell = L+1 \end{cases} \tag{16}$$

where $\mathsf{purelin}(\cdot)$ is linear transfer function, i.e., $x = \mathsf{purelin}(x)$.

Proof: Given an input $\boldsymbol{\eta}_0 \in \mathcal{U}$ and $\tilde{\boldsymbol{\eta}}_0 = \hat{\boldsymbol{\eta}}_0 = \boldsymbol{\eta}_0$ and considering layers $1 \leq \ell \leq \hat{L} - 1$ of $\tilde{\Phi}$, we have

$$\tilde{\boldsymbol{\eta}}_{\hat{L}-1} = \begin{bmatrix} \phi_{L-1} \circ \cdots \circ \phi_1(\mathbf{W}_1 \boldsymbol{\eta}_0 + \mathbf{b}_1) \\ \hat{\phi}_{\hat{L}-1} \circ \cdots \circ \hat{\phi}_1(\hat{\mathbf{W}}_1 \hat{\boldsymbol{\eta}}_0 + \hat{\mathbf{b}}_1) \end{bmatrix} \tag{17}$$

Specifically, we consider $\ell = 1$ such that

$$\tilde{\boldsymbol{\eta}}_1 = \begin{bmatrix} \phi_1(\mathbf{W}_1 \boldsymbol{\eta}_0 + \mathbf{b}_1) \\ \hat{\phi}_1(\hat{\mathbf{W}}_1 \hat{\boldsymbol{\eta}}_0 + \hat{\mathbf{b}}_1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\eta}_1 \\ \hat{\boldsymbol{\eta}}_1 \end{bmatrix} \tag{18}$$

Moreover, when $1 < \ell \leq \hat{L} - 1$, it leads to

$$\tilde{\mathbf{W}}_\ell \tilde{\boldsymbol{\eta}}_{\ell-1} + \tilde{\mathbf{b}}_\ell = \begin{bmatrix} \mathbf{W}_\ell \boldsymbol{\eta}_{\ell-1} + \mathbf{b}_\ell \\ \hat{\mathbf{W}}_\ell \hat{\boldsymbol{\eta}}_{\ell-1} + \hat{\mathbf{b}}_\ell \end{bmatrix}, \ 1 < \ell \leq \hat{L} - 1 \tag{19}$$

Staring from $\ell = 1$ and recursively using (19) into (17), one can obtain

$$\tilde{\boldsymbol{\eta}}_{\hat{L}-1} = \begin{bmatrix} \boldsymbol{\eta}_{\hat{L}-1} \\ \hat{\boldsymbol{\eta}}_{\hat{L}-1} \end{bmatrix} \tag{20}$$

Furthermore, when $\hat{L} \leq \ell \leq L - 1$, one can derive

$$\tilde{\mathbf{W}}_\ell \tilde{\boldsymbol{\eta}}_{\ell-1} + \tilde{\mathbf{b}}_\ell = \begin{bmatrix} \mathbf{W}_\ell \boldsymbol{\eta}_{\ell-1} + \mathbf{b}_\ell \\ \hat{\boldsymbol{\eta}}_{\ell-1} \end{bmatrix}, \ \hat{L} \leq \ell \leq L - 1 \tag{21}$$

and

$$\boldsymbol{\eta}_{L-1} = \phi_{L-1} \circ \cdots \circ \phi_{\hat{L}}(\mathbf{W}_{\hat{L}} \boldsymbol{\eta}_{\hat{L}-1} + \mathbf{b}_{\hat{L}-1})$$

which leads to

$$\tilde{\boldsymbol{\eta}}_{L-1} = \begin{bmatrix} \boldsymbol{\eta}_{L-1} \\ \hat{\boldsymbol{\eta}}_{\hat{L}-1} \end{bmatrix} \tag{22}$$

Then, when $\ell = L$, it yields that

$$\tilde{\mathbf{W}}_L \tilde{\boldsymbol{\eta}}_{L-1} + \tilde{\mathbf{b}}_L = \begin{bmatrix} \mathbf{W}_L \boldsymbol{\eta}_{L-1} + \mathbf{b}_L \\ \hat{\mathbf{W}}_{\hat{L}} \hat{\boldsymbol{\eta}}_{\hat{L}-1} + \hat{\mathbf{b}}_{\hat{L}} \end{bmatrix} \tag{23}$$

Thus, one can obtain

$$\tilde{\boldsymbol{\eta}}_L = \begin{bmatrix} \boldsymbol{\eta}_L \\ \hat{\boldsymbol{\eta}}_{\hat{L}} \end{bmatrix} \tag{24}$$

At last, when $\ell = L+1$, the following result can be obtained

$$\tilde{\boldsymbol{\eta}}_{L+1} = \tilde{\mathbf{W}}_{L+1}\tilde{\boldsymbol{\eta}}_L + \tilde{\mathbf{b}}_{L+1} = \begin{bmatrix}\mathbf{I}_{n_L} & -\mathbf{I}_{n_{\hat{L}}}\end{bmatrix}\begin{bmatrix}\boldsymbol{\eta}_L \\ \hat{\boldsymbol{\eta}}_{\hat{L}}\end{bmatrix}$$

which implies that $\tilde{\boldsymbol{\eta}}_{L+1} = \boldsymbol{\eta}_L - \hat{\boldsymbol{\eta}}_{\hat{L}}$. Therefore, we can conclude that $\tilde{\Phi}(\tilde{\boldsymbol{\eta}}_0) = \Phi(\boldsymbol{\eta}_0) - \hat{\Phi}(\hat{\boldsymbol{\eta}}_0)$ as long as $\tilde{\boldsymbol{\eta}}_0 = \hat{\boldsymbol{\eta}}_0 = \hat{\eta}_0$ and

$$\rho(\Phi, \hat{\Phi}, \mathcal{U}) = \sup_{\tilde{\eta}_0 \in \mathcal{U}} \left\| \tilde{\Phi}(\tilde{\boldsymbol{\eta}}_0) \right\| \tag{25}$$

The proof is complete. ■

Remark 1: In the process of augmenting neural networks Φ and $\hat{\Phi}$ into $\tilde{\Phi}$, the case of $\ell = 1$ in (14)–(16) ensures that augmented neural network $\tilde{\Phi}$ takes the one same input $\boldsymbol{\eta}_0$ for the subsequent calls involving both processes of Φ and its reduced-size version $\hat{\Phi}$. Then, for $1 < \ell \leq \hat{L} - 1$, augmented neural network $\tilde{\Phi}$ conducts the computation of Φ and and its reduced-size version $\hat{\Phi}$ parallelly for the hidden layers of $1 < \ell \leq \hat{L} - 1$. When $\hat{L} \leq \ell \leq L - 1$, the hidden layers of reduced-size neural network $\hat{\Phi}$ which has fewer hidden layers are expanded to match the number of layers of the original neural network Φ 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 vectors are zero vectors. This expansion is formalized as in the case of $\hat{L} \leq \ell \leq L - 1$ in (14)–(16). Moreover, as $\ell = L$, this layer is a combination of output layers of both Φ and $\hat{\Phi}$ to generate the same outputs of Φ and $\hat{\Phi}$. At last, a comparison layer $L + 1$ is added to compute the exact difference between the original neural network Φ and its reduced-size version $\hat{\Phi}$.

Remark 2: As shown in Theorem 1, the key of computing model reduction precision $\rho(\Phi, \hat{\Phi}, \mathcal{U})$ is to compute the maximal output value of augmented neural network $\tilde{\Phi}$ with respect to input set $\mathcal{U}$. This can be efficiently done by neural network reachability analysis. For instance, as in NNV neural network reachability analysis tool, the reachable sets are in the form of a family of polyhedral sets [16], and in the IGNNV tool, the output reachable set is a family of interval sets [13], [14]. With the reachable set $\tilde{\mathcal{Y}}$, the model reduction precision $\rho(\Phi, \hat{\Phi}, \mathcal{U})$ can be easily obtained by searching for the maximal value of $\|\tilde{\boldsymbol{\eta}}_{L+1}\|$ in $\tilde{\mathcal{Y}}$, e.g., testing throughout a finite number of vertices in polyhedral sets.

IV. Safety Verification of Neural Network Control Systems

In this section, we apply neural network model reduction and model reduction precision to a neural network control system. By replacing the original neural network controller with a reduced-size neural network, the computational cost of safety verification can be significantly reduced. Moreover, the model reduction precision allows an over-estimation of the difference in behavior between the original neural network and the reduced-size one. For the reachability analysis of neural networks, the following result can be obtained.

Proposition 1: Given neural network Φ, its reduced-size version $\hat{\Phi}$ with output set $\hat{\mathcal{Y}}$, and model reduction precision $\rho(\Phi, \hat{\Phi}, \mathcal{U})$, the output reachable set of original neural network Φ satisfies

$$\mathcal{Y} \subseteq \hat{\mathcal{Y}} \oplus \mathcal{B}(\mathbf{0}_{n_L \times 1}, \frac{\rho(\Phi, \hat{\Phi}, \mathcal{U})}{2}) \tag{26}$$

where $\mathcal{B}(\mathbf{0}_{n\times 1}, r)$ denotes a ball centered at $\mathbf{0}_{n\times 1}$ with a radius of r, and $\oplus$ denotes the Minkowski sum.

Proof: This can be obtained straightforwardly by the definition of model reduction precision $\rho(\Phi, \hat{\Phi}, \mathcal{U})$ which characterizes the maximal difference between the outputs of Φ and $\hat{\Phi}$. The proof is complete. ■

The reachable set estimation for a sampled-data neural network control system in the form of (4) generally involves two parts: 1) Output set computation for neural network controllers denoted by

$$\mathcal{Y} = \texttt{reachNN}(\Phi, \mathcal{U}) \tag{27}$$

which can be efficiently obtained by neural network reachability tools such as [13], [16] and (26), and 2) Reachable set computation of system (1). For the reachable set computation of systems described by ODEs, there exist a variety of approaches and tools such as those well-developed in [17]–[20]. The following functions are given to denote the reachable set estimation for sampled data ODE models during $[t_k, t_{k+1}]$,

$$\mathcal{R}_e([t_k, t_{k+1}]) = \texttt{reachODEx}(f, \mathcal{U}(t_k), \mathcal{R}_e(t_k)) \tag{28}$$

$$\mathcal{Y}_e(t_k) = \texttt{reachODEy}(h, \mathcal{R}_e(t_k)) \tag{29}$$

where $\mathcal{U}(t_k)$ is the input set for sampling interval $[t_k, t_{k+1}]$. $\mathcal{R}_e(t_k)$ and $\mathcal{R}_e([t_k, t_{k+1}])$ are the estimated reachable sets

Algorithm 1: Reachable Set computation for Neural Network Control Systems (4)

Input : System dynamics f, h; Reduced-size neural network $\hat{\Phi}$; Model reduction precision $\rho(\Phi, \hat{\Phi}, \mathcal{U})$; Initial set $\mathcal{X}_0$; Input set $\mathcal{V}$

Output: Reachable set estimation $\mathcal{R}_e([t_0, t_f])$.

Function *reachNNCS*

/* Initialization */

$k \leftarrow 0$

$t_{K+1} \leftarrow t_f$

$\mathcal{R}_e(t_0) \leftarrow \mathcal{X}_0$

/* Iteration for all sampling intervals */

while $k \leq K$ **do**

$\mathcal{Y}_e(t_k) \leftarrow \texttt{reachODEy}(h, \mathcal{R}_e(t_k))$

$\mathcal{H} \leftarrow \mathcal{Y}_e(t_k) \times \mathcal{V}$

$\hat{\mathcal{U}}_e(t_k) \leftarrow \texttt{reachNN}(\hat{\Phi}, \mathcal{H})$

$\mathcal{U}_e \leftarrow \hat{\mathcal{U}}(t_k) \oplus \mathcal{B}(\mathbf{0}_{n_L \times 1}, \frac{\rho(\Phi, \hat{\Phi}, \mathcal{U})}{2})$

$\mathcal{R}_e([t_k, t_{k+1}]) \leftarrow \texttt{reachODEx}(f, \mathcal{U}_e, \mathcal{R}_e(t_k))$

$k \leftarrow k + 1$

end

return $\mathcal{R}_e([t_0, t_f]) \leftarrow \bigcup_{k=0,1\ldots,K} \mathcal{R}_e([t_k, t_{k+1}])$

for state $\mathbf{x}(t)$ at sampling instant t_k and interval $[t_k, t_{k+1}]$, respectively. $\mathcal{Y}_e(t_k)$ is the estimated reachable set for output $\mathbf{y}(t_k)$. With the results in Proposition 1, we can use the reduced-size neural network to compute output set $\hat{\mathcal{U}}(t_k)$ which is much more computationally efficient due to its smaller size, and replace the output set of $\mathcal{U}(t_k)$ by $\hat{\mathcal{U}}(t_k) \oplus \mathcal{B}(\mathbf{0}_{n_L \times 1}, \frac{\rho(\Phi, \hat{\Phi}, \mathcal{U})}{2})$ where $\rho(\Phi, \hat{\Phi}, \mathcal{U})$ is normally obtained through a one-time offline computation. The reachable set computation process is shown in Algorithm 1.

Based on the estimated reachable set obtained by Algorithm 1, the safety property can be examined with the existence of intersections between the estimated reachable set and unsafe region $\neg\mathcal{S}$.

Proposition 2: Consider a neural network control system in the form of (4) with a safety specification $\mathcal{S}$, the system is safe in $[t_0, t_f]$, if $\mathcal{R}_e([t_0, t_f]) \cap \neg\mathcal{S} = \emptyset$, where $\mathcal{R}_e([t_0, t_f])$ is an estimated reachable set obtained by Algorithm 1.

V. Evaluation on Adaptive Cruise Control Systems

In this section, our approach will be evaluated by the safety verification of an Adaptive Cruise Control (ACC) system equipped with a neural network controller as depicted in Fig. 1. The system dynamics is in the form of

$$\begin{cases} \dot{x}_l(t) = v_l(t) \\ \dot{v}_l(t) = \gamma_l(t) \\ \dot{\gamma}_l(t) = -2\gamma_l(t) + 2\alpha_l(t) - \mu v_l^2(t) \\ \dot{x}_e(t) = v_e(t) \\ \dot{v}_e(t) = \gamma_e(t) \\ \dot{\gamma}_e(t) = -2\gamma_e(t) + 2\alpha_e(t) - \mu v_e^2(t) \end{cases} \tag{30}$$

where $x_l(x_e)$, $v_l(v_e)$ and $\gamma_l(\gamma_e)$ are the position, velocity and actual acceleration of the lead (ego) car, respectively. $\alpha_l(\alpha_e)$ is the acceleration control input applied to the lead (ego) car, and $\mu = 0.001$ is the friction parameter. The ACC controller we considered here is a 5×20 feed-forward neural network with ReLU as its activation functions. The sampling scheme is considered as a periodic sampling every 0.01 seconds, i.e., $t_{k+1} - t_k = 0.01$ seconds.

The sampled-data neural network controller for the acceleration control of the ego car is in the form of

$$\alpha_e(t) = \Phi(v_{set}(t_k), t_{gap}, v_e(t_k), d_{rel}(t_k), v_{rel}(t_k)) \tag{31}$$

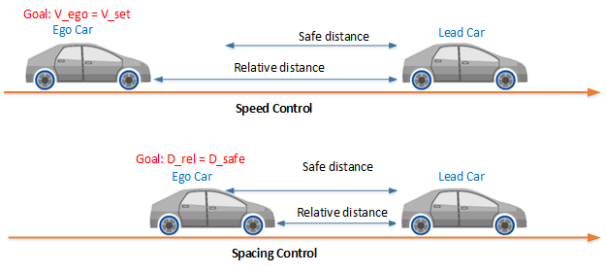

Fig. 1. Adaptive cruise control system model [13]

in which $k \in [t_k, t_{k+1}]$. The threshold of the safe distance between the two cars satisfies a function as defined below in the form of

$$d_{safe} > d_{thold} = d_{def} + t_{gap} \cdot v_e \tag{32}$$

where d_{safe} is the safe distance between the ego car and lead car, d_{thold} is the threshold of the safe distance, d_{def} is the standstill default spacing, and t_{gap} is the time gap between the vehicles. The safety verification scenario we consider is that the lead car decelerates with $\alpha_l = 2$ to reduce its speed as an emergency braking occurs. We expect that the ego car guided by a neural network control system is able to maintain a safe relative distance to the lead car to avoid the collision.

The safety specification parameter we consider in the simulation is $t_{gap} = 1.4$ seconds and $d_{def} = 10$. The time horizon that we want to verify is 3 seconds, i.e., 300 sampling intervals, after the emergency braking comes into play. The initial sets are $x_l(0) \in [94, 96]$, $v_l(0) \in [30, 30.2]$, $\gamma_l(0) = 0$, $x_e(0) \in [10, 11]$, $v_e(0) \in [30, 30.2]$, and $\gamma_e(0) = 0$.

As mentioned above, the size of the hidden layer of the original neural network controller is 5×20, i.e., 5 layers with 20 neurons in each layer, Through neural network model reduction, we replace the original neural network controller with a reduced-size neural network of hidden layer size 2×5, i.e., 2 layers with 5 neurons in each layer, combined with a model reduction precision $\rho = 1.0234$. For the continuous-time nonlinear dynamics, we use CORA [18] to do the reachability analysis for the time interval between two sampling instants, and IGNNV in [13] is used for neural network reachability analysis.

The output reachable set of the ACC system for the relative distance between the lead car and the ego car over time can be shown in Figs. 2 and 3. Notably, the computation time has been significantly reduced from 12.355 seconds to 1.099 seconds when using a reduced-size neural network as shown

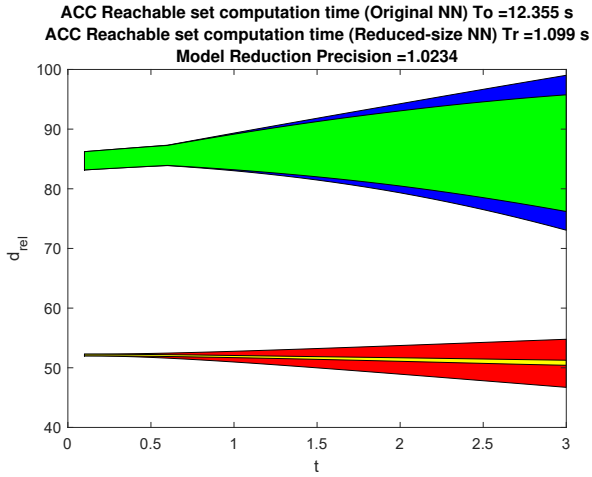

Fig. 2. Reachable set of ACC systems. The green (blue) pipe indicates the output reachable set of the dynamic system with the original (reduced-size) neural network. It is obvious that the blue pipe completely wraps around the green pipe. The yellow (red) pipe denotes the safe distance threshold region for the dynamical system with the original (reduced-size) neural network.

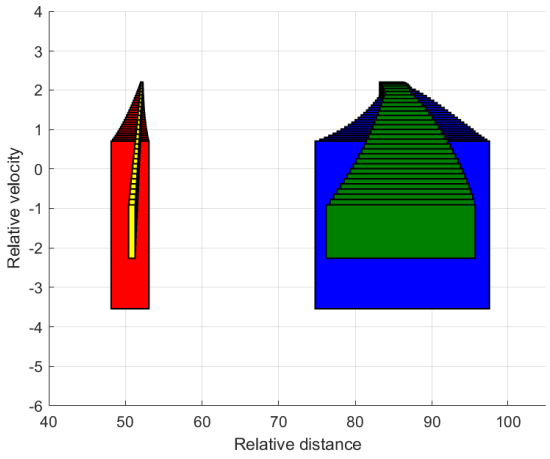

Fig. 3. Reachable set estimation for relative distance and the relative velocity between lead and ego cars. The green (blue) pipe indicates the reachable set of the dynamical system with the original (reduced-size) neural network. The yellow (red) pipe denotes the safe distance threshold and the relative velocity between lead and ego cars for the dynamical system with the original (reduced-size) neural network.

TABLE I

COMPARISON OF ACC REACHABLE SET CALCULATION TIMES

Dynamical Systems	Computational Time
ACC with original neural network	12.355 s
ACC with reduced-size neural network	1.099 s

in Fig. 2 and Table I. The system is safe when the output reachable set of relative distances does not intersect with the safe distance threshold region.

In summary, the simulations show that the closed-loop system with the reduced-size neural network can be used for safety verification of the original system as long as the model reduction precision can be provided. The reduced-size neural network can significantly reduce the computational time of the entire process of solving the neural network control system for the output reachable set.

VI. CONCLUSIONS

This paper investigates the problem of simplifying the safety verification of neural network control systems, proposes a concept of model reduction precision that characterizes the minimum upper bound on the outputs between a neural network and its reduced-size one, and proposes an algorithm to calculate the model reduction precision. By using a reduced-size neural network as the neural network controller and introducing the model reduction precision in the computation of the output reachable. Combined with the calculation of reachable sets for dynamical systems, we give the reachable set computation algorithm based on model reduction of neural network control systems. In this way, we can obtain the over-approximated output reachable set of the original neural network control system with less computation time and enable a simplification of safety verification processes. The developed results are applied to the ACC system to verify its effectiveness and feasibility.

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] 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.

[4] 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.

[5] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016, pp. 1–10.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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*. Springer, 2017, pp. 97–117.

[11] H.-D. Tran, W. Xiang, and T. T. Johnson, "Verification approaches for learning-enabled autonomous cyber–physical systems," *IEEE Design & Test*, vol. 39, no. 1, pp. 24–34, 2022.

[12] 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.

[13] 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.

[14] 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.

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

[16] H.-D. Tran, X. Yang, D. Manzanas 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*. Springer, 2020, pp. 3–17.

[17] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2011, pp. 379–395.

[18] M. Althoff, "An introduction to CORA 2015," in *Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[19] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 258–263.

[20] 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*. ACM, 2017, pp. 173–178.