Neurocomputing 562 (2023) 126879

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Computationally efficient neural hybrid automaton framework for learning complex dynamics

Tao Wang [a], Yejiang Yang [a,b], Weiming Xiang [b,*]

[a] School of Electrical Engineering, Southwest Jiaotong University, Chengdu, China
[b] School of Computer and Cyber Sciences, Augusta University, Augusta GA 30912, USA

ARTICLE INFO

Communicated by J. Cao

Keywords:
Neural networks
Hybrid automaton
Extreme learning machine
Data-driven modeling
Reachability analysis

ABSTRACT

This paper proposes a computationally efficient and effective data-driven modeling framework for dynamical systems. The proposed modeling framework employs a collection of shallow neural networks known as Extreme Learning Machines (ELMs) to model local system behaviors along with data-driven inferred transitions among local models to establish a neural hybrid automaton model. First, the sampled system inputs are mapped to the corresponding feature spaces to obtain data-driven partitions, which subsequently define the transitions and invariants of the neural hybrid automaton model through a novel data-driven mode clustering process. Then, a collection of ELMs are trained to approximate the local dynamics. The learning processes integrate a segmented data merging procedure for location identification and a local dynamics modeling process. The proposed neural hybrid automaton models can capture behaviors of complex dynamical systems with high modeling precision but significantly lower computational complexities in computationally expensive tasks such as training and verification, which are traditionally considered to be computationally expensive tasks for neural network models. A computationally efficient set-valued reachability analysis method which is commonly used in safety verification is then developed based on interval analysis and a novel Split and Combine process. Finally, applications to modeling the limit cycle and human handwritten motions are presented to show the effectiveness and efficiency of our approach.

1. Introduction

The success of neural networks are well documented, such as learning-based control for hybrid systems [1,2] and nonlinear dynamics [3,4], studying the speech enhancement in [5], predicting the cumulative COVID-19 incidence rate in [6], etc. Among these applications, Deep Neural Networks (DNN) have received particular attention for their accuracy and ability to handle complex training samples such as studying the recommender systems in [7], predicting and elucidating the optical chirality of two-dimensional diffractive chiral metamaterials in [8], etc. Compared with DNN, Shallow Neural Networks (SNN) are prone to getting overfitted when modeling the system, which leads to a bad-performance model. Based on Universal Approximation Theory (UAT), and thanks to recent works on Broad Learning System [9], the guaranteed distance between a function and its SNN approximation in [10], modeling with SNN will be competitive when the training error is tolerable.

When it comes to modeling a dynamical system in real-life applications, some requirements such as safety and stability must be satisfied in practice. Works on adding safety constrains in training and verification of the learning models provide great tools to meet the requirements, but might become an extra computational burden in training and verification. Besides, the scale of the neural network model can affect the time consumption and computational complexity of the verification and validation of the model, which makes the DNN model may be difficult to verify due to its large scale. On the other hand, modeling with the SNN will reduce the computational complexity for training and verification and improve scalability.

Neural networks are often viewed as black boxes. In most cases of data-driven modeling, a neural network model, especially a DNN model, may represent significant computation challenges in the training and verification due to their complex structures. When it comes to safety-critical scenarios, the time consumption for verification of the neural network model will be unacceptable. The computational bottleneck exists considering the complex structure of the model. Compared with DNN models for complex applications such as graphics and language, modeling the dynamic system with multiple SNNs to approximate the local information of the system makes the model

* Corresponding author.
E-mail address: wxiang@augusta.edu (W. Xiang).

https://doi.org/10.1016/j.neucom.2023.126879
Received 16 November 2022; Received in revised form 26 June 2023; Accepted 3 October 2023
Available online 6 October 2023

more competitive (e.g., reducing computational complexity) to train and verify.

A dynamical system can be modeled by a hybrid automaton with a finite number of partitions, including transitions between them. When modeling and verifying, the hybrid automaton will be parallel-trained and partially verified under various partitions, which will reduce the computational complexity. Previous works are focusing on modeling the hybrid system with the hybrid model based on the sample collected. In this paper, we propose a neural hybrid automaton modeling framework that aims to model general dynamical systems to reduce the computational complexity of training and verification of the learning-based model. Each subsystem of our proposed model is trained by an SNN called Extreme Learning Machine (ELM) [11], and the ELMs will be trained and verified under partitions that are generated from the feature space of samples. Our proposed modeling framework will reduce the time consumption and improve the scalability of the neural network model without sacrificing no accuracy compared with traditional modeling methods.

1.1. Motivation and related works

This work is majorly inspired by the research on learning-based modeling for hybrid systems; safety verification for learning models; distributed neural network learning; development of ELM such as

- Various studies have been conducted on modeling the hybrid systems with learning models, such as modeling the DC-DC converter in [12], focusing on modeling the hybrid system on jump detection in [13], studying the modeling of the hybrid and Cyber–physical Systems in [14], etc. The idea of modeling with a hybrid system allows distributed training and verification and therefore improves the model's scalability.
- A variety of research has been conducted on adding constraints in training neural networks and leading to a computational burden in training and verification, such as certifying the data-driven model from a safety perspective in [15,16], training robust neural networks in [17], providing guaranteed distance between the neural network model and the system in [10], utilizing control contraction metrics in training neural networks in [18,19], adding the Lyapunov constraints in training neural networks in [20], focusing on the Lipschitz constant of neural networks in [21], developing a bound verifier for neural networks in [22], approximating the reachable set of the neural network model in [23], etc. The verification tools presented in [22,24–27] provide powerful solutions to verify the neural network model and also become a computational bottleneck.
- The effectiveness of distributed learning is well demonstrated by many works, e.g., studying distributed reduced convolution neural networks in [28], solving forward and inverse problems in nonlinear partial differential equations using distributed learning machines in [29], promoting a distributed deep reinforcement learning method for traffic light control in [30], studying the quantum distributed deep learning architectures in [31], promoting wide and deep graph neural network with distributed online learning in [32]. The recently proposed Pathways Language Model [33] developed from the Mixture of Experts Model [34] shines a new light on DNN modeling by reducing the computational complexity through multiple small-size neural networks which are activated by a sparse gating system.
- As a computationally efficient neural network, ELM has received particular attention such as investigating multilayer ELM in [35, 36], proposing variational quantum extreme learning machine in [37], and studying the residual compensation ELM for regression problems in [38]. Therefore, we employ multiple simple ELMs as the dynamical description of the proposed neural hybrid automaton in this work.

1.2. Contributions

The main contributions of this paper include:

- A novel data-driven neural hybrid automaton modeling framework is proposed to learn complex dynamical systems, which aims to advance the state-of-the-art of dynamical system modeling techniques with a focus on improving the model scalability.
- Novel data-driven modeling and verification methods are developed to enrich the neural hybrid automaton modeling framework, which aims to efficiently handle computationally expensive tasks such as training and verification while sacrificing no accuracy in modeling.

This paper is organized as follows: Preliminaries and Problem Formulation, challenges in modeling the dynamical system with a neural hybrid automaton via ELMs are given in Section 2. The main result, a detailed neural hybrid automaton modeling framework is given in Section 3. Then, the applications to modeling complex dynamics with our proposed framework is presented in Section 4. The conclusions are given in Section 5.

Notations: In the rest of the paper, $\mathbb{N}$ denotes the natural number sets, $\mathbb{R}$ is the field of real numbers, $\mathbb{N}^+$ refers to the positive integer set; $\mathbb{R}^n$ stands for the vector space of n-tuples of real number; $\underline{X}$ and $\overline{X}$ are the lower bound and upper bound of an interval X, respectively.

2. Preliminaries and problem formulation

In this section, we attempt to form formal definitions for the general framework. This is useful for future development of this framework consisting of different components such as feature, set-valued reachability analysis for the neural networks, etc. The main problems of the modeling framework with multiple ELMs will be formed based on these formal definitions.

Our goal is to build a hybrid automaton with learning capabilities using samples of states and external inputs from a dynamical system. Based on the samples, we will model the system in the form of

$$x(k+1) = f(x(k), u(k)), \tag{1}$$

where the states $x(k) \in \mathbb{R}^d$, external inputs $u(k) \in \mathbb{R}^n$, while the nonlinear function f needs to be approximated by training neural networks.

Our proposed framework will be based on feature functions and feature spaces in [39] defined as follows.

Definition 1. A feature is a function

$$\Theta : \Omega \to \hat{\mathcal{P}}, \tag{2}$$

in which $\Omega \subset \mathbb{R}^d$ is the input region where all sample states $x(k) \in \Omega$ and $\hat{\mathcal{P}} \subset \mathbb{R}^m$ is the feature space.

Remark 1. The states of the dynamical system in this study will be mapped to a feature space that reflects the intrinsic dynamics of the system with Θ. The feature is helpful when the states of the system are high-dimensional and we need to abstract the intrinsic dynamics to a low-dimensional space [40], namely, $m \leq d$. An example of a feature is when modeling a time-depended switching system, the feature maps $x(k)$ to a one-dimensional space [41], i.e., $\Theta : \mathbb{R}^3 \to \mathbb{R}$, which suggests the feature should capture the key characteristics of the dynamics from states of the system.

In this work, features are for analyzing the states from a dynamical system in a low-dimensional feature space. On the other hand, when considering a dynamical system with low-dimensional states, such as [42], the states may be low-dimensional, which implies that the feature may be a function that maps the states to themselves. Feature space can be divided into subspaces called partitions defined as follows.

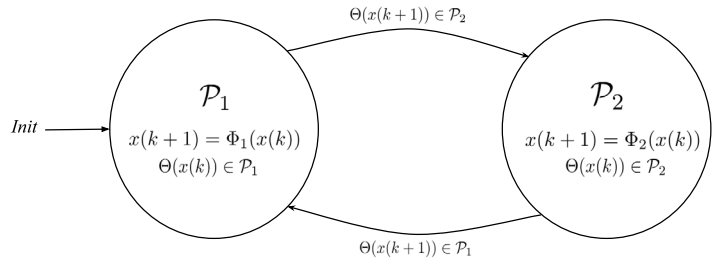


Fig. 1. An illustration of a neural hybrid automaton model with 2 partitions $\mathcal{P}_1$, $\mathcal{P}_2$, where the initial condition is denoted by $Init : (x_0, u_0)$ starting at $\mathcal{P}_1$. The guard functions that define the transitions between $\mathcal{P}_1$ and $\mathcal{P}_2$ are specified based on feature space Θ, while the invariants determine whether the next time step is within a specific partition, the set of ELMs $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2\}$ is as the dynamical descriptions for its corresponding partitions.

Definition 2. Feature space $\hat{\mathcal{P}}$ can be divided to a collection of N subspaces, which satisfies $\hat{\mathcal{P}} \subseteq \bigcup_{i=1}^{N} \mathcal{P}_i$ and $\mathcal{P}_i \bigcap \mathcal{P}_j = \emptyset$, $\forall i \neq j$, in which the collection of sets $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_N\}$, is called partitions.

Remark 2. By classifying the system dynamics into different modes in the feature space, the neural network will perform better in approximating the local dynamics. Hence the feature function that maps the states to the feature space is very important, it should be noted that for our following detailed algorithms, the feature is rough (by mapping the low-dimensional state to itself in the modeling of the limit cycle and the Human handwritten motions), there is plenty of room for future updates by applying the power function such as Lyapunov candidates, etc., as the feature function. However, the obtaining of partitions from feature space should be a data-driven process when there is no prior knowledge of the dynamical system. Partitions are useful when we classify the system dynamics from samples and they pave the way for parallel training of neural networks in approximating the dynamics, namely, neural networks can approximate the dynamics from the samples within their corresponding partitions.

2.1. Neural hybrid automata

A neural hybrid automaton model consists of variable components describing both dynamics and the transition logic. To demonstrate this concept, an illustration is given in Fig. 1. Specifically, the neural hybrid automaton can be formally defined as follows.

Definition 3. A neural hybrid automaton is defined by a tuple

$$\mathcal{H} \triangleq \langle \mathcal{P}, \mathcal{X}, init, \mathcal{E}, g, \mathcal{G}, inv, \boldsymbol{\Phi} \rangle \tag{3}$$

in which the components are defined by

- *Partitions*: $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_N\}$ is a finite set of feature space partitions where $\mathcal{P}_i \subset \hat{\mathcal{P}}$ denotes the ith partition.
- *State Variables*: $\mathcal{X} \subset \mathbb{R}^d$ is the state region which contains the state variable $x(k) \in \Omega$. For a kth time step state variable at ith partitions, it is denoted by $x(k)$ which satisfies $\Theta(x(k)) \in \mathcal{P}_i$.
- *Initial conditions*: $init = (x(0), u)$ where $x(0) \in \mathcal{X}_{(0)}$, and $u \in \mathcal{U}$ are the initial states and external inputs, in which $\mathcal{X}_{(0)}$ is the initial set of states and $\mathcal{U}$ is the input set for the external input u.
- *Transitions*: $\mathcal{E} \subset \mathcal{P} \times \mathcal{P}$ is the set of transitions where a discrete transition from ith partition to jth partition is taking place, i.e., $\mathcal{P}_i \to \mathcal{P}_j$, $e_{ij} = (\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{E}$.
- *Guard functions*: $g : \mathcal{E} \to \mathcal{G}$ is the guard function mapping each transition element e_{ij} to its guard $g(e_{ij}) \in \mathcal{G}$.
- *Guards*: $\mathcal{G} \subseteq 2^{\mathcal{X}}$ is the guard set which satisfies $\forall e_{ij} \in \mathcal{E}$, $g(e_{ij}) \in \mathcal{G}$. The guard is satisfied by the state variable when the neural hybrid automaton model takes a transition from current partition to another given partition, i.e., $x_k \vDash g(e_{ij})$ if and only if $\mathcal{P}_k = \mathcal{P}_i$ and $x_k \in g(e_{ij})$.
- *Invariants*: $inv : \mathcal{P} \to 2^{\mathcal{X}}$ is a mapping that assigns an invariant $inv(\mathcal{P}_i) \subseteq \mathcal{X}$ for each partition $\mathcal{P}_i \in \mathcal{P}$. An invariant is satisfied by all the states of a hybrid automaton model for a given partition $\mathcal{P}_i$, i.e., $(\mathcal{P}_i, x) \vDash inv(\mathcal{P}_i)$ if and only if $x \in inv(\mathcal{P}_i)$.
- *Set of ELMs*: $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \dots, \boldsymbol{\Phi}_N\}$ is the set of ELMs in which each $\boldsymbol{\Phi}_i$ describes the dynamical behaviors for each given partition $\mathcal{P}_i \in \mathcal{P}$.

Remark 3. The feature partitions in $\mathcal{P}$ play an important role in implementing the idea of using multiple neural networks to model complex dynamics. With a finite number of partitions in $\mathcal{P}$, i.e., $N = |\mathcal{P}|$ where $|\mathcal{P}|$ is the cardinal of $\mathcal{P}$, implies the number of subsystems in neural hybrid automaton model $\mathcal{H}$, and the subsystem $i \in \mathcal{I} = \{1, \dots, N\}$ is associated to partition $\mathcal{P}_i$. Based on feature partition $\mathcal{P}$, the partitions for state space $\mathcal{X}$, i.e., invariants, can be also defined as $\mathcal{X}_i$ in the form of

$$\mathcal{X}_i = \{x \mid x = \Theta^{-1}(z),\ \forall z \in \mathcal{P}_i\}. \tag{4}$$

Remark 4. It is noted that, in the description for the system dynamics, the inputs of ELMs consider both the state variables $x(k)$ as well as external inputs $u(k)$, e.g., control input, or external disturbances, while the outputs of ELMs are the state variables at next time step, i.e.,

$$x(k+1) = \boldsymbol{\Phi}_i(x(k), u(k)). \tag{5}$$

The transitions, guards, and invariants can be subsequently defined as long as the partitions are obtained and as a result, a neural hybrid automaton model can be established. For example, the guard function, denoted as $g(e_{ij}) : \mathcal{E} \to \mathcal{G}$, is responsible for determining the transition between feature space partitions in the neural hybrid automaton model. It takes the current state or event e_{ij} as input and maps it to the corresponding guard $\mathcal{G}$. The guard function is defined based on the specific problem requirements and the characteristics of the system under study. It encapsulates the decision boundaries or conditions that trigger a transition between partitions. Investigating the guard function of the neural hybrid automaton will give us a chance to discover the transition relationship within the neural hybrid automaton.

As a class of SNN, an ELM is a single-hidden-layer neural network that offers a strong generalization performance as well as a quick training speed. Due to its simple structure and easy training mechanism, we employ ELMs in our modeling framework as dynamical system behavior approximators. For an ELM $\boldsymbol{\Phi}$ with L hidden nodes, it is in the form of

$$\boldsymbol{\Phi}(x) = \sum_{i=1}^{L} \beta_i h_i(x) = h(x)\beta, \tag{6}$$

where the output nodes are linear and $\beta = [\beta_1, \dots, \beta_L]^T$ is the output weight vector. $h(x) = [h_1(x), \dots, h_L(x)]$ is the activation functions for hidden layer neurons. In particular, $h_i(x)$ is defined by

$$h_i(x) = G(x, a_i, b_i) = g(a_i^{\mathrm{T}} x + b_i),\ a_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \tag{7}$$

where $g(\cdot)$ is the activation function for hidden layer neurons, and a_i and b_i are the input weight and the bias of the ith neuron in the hidden layer.

Given N arbitrary distinct input–output samples $\{x_i, t_i\}$ with $x_i \in \mathbb{R}^n$ and $t_i \in \mathbb{R}^d$, the objective of training ELM, i.e, by approximating its output matrix with the target matrix $\mathbf{T}$ in locating the minimal norm Least-Squares (LS) solution for a set of N input–output sample pairs written in

$$\min_{\beta \in \mathbb{R}^{L\times d}} \|\mathbf{H}\beta - \mathbf{T}\|, \tag{8}$$

in which $\mathbf{H}$ is the randomized hidden layer output matrix as

$$\mathbf{H} = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \dots & h_L(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \dots & h_L(x_N) \end{bmatrix},$$

and $\mathbf{T}$ is the target output matrix, in this case, is the sample output in the form of

$$\mathbf{T} = \begin{bmatrix} t_1^{\mathrm{T}} \\ \vdots \\ t_N^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} t_{1,1} & \cdots & t_{1,d} \\ \vdots & \ddots & \vdots \\ t_{N,1} & \cdots & t_{N,d} \end{bmatrix}.$$

The following Mean Square Error (MSE) is used as a metric for performance assessment, using N sets of input and output samples as

$$MSE = \frac{1}{N}\left\| \sum_{i=1}^{N} (\Phi(x_i) - t_i) \right\|, \tag{9}$$

in which x_i, t_i are the input and output of ith sample pair in the N sample data set, respectively.

Different from using one single neural network for modeling dynamical systems, a neural hybrid automaton model $\mathcal{H}$ resorts to approximating the local dynamics in each individual. The complexity of learning will be reduced compared with globalized methods with neural networks of smaller sizes. Additionally, the distributed modeling framework allows for parallel training, which reduces training time.

2.2. Set-valued reachability analysis

In this paper, we aim at reducing the computational cost of reachability analysis of data-driven neural network models of complex dynamical systems. A family of necessary definitions regarding reachability analysis is introduced.

Definition 4. Given a neural network Φ and an input set $\mathcal{U}$, the following set

$$\mathcal{Y} = \left\{ y \in \mathbb{R}^d \mid y = \Phi(u),\ u \in \mathcal{U} \right\}, \tag{10}$$

is called the output set of neural network Φ. Furthermore, we use $[\Phi]$ to denote the set-valued reachability computation procedure for neural network Φ, i.e.,

$$\mathcal{Y} = [\Phi](\mathcal{U}). \tag{11}$$

Remark 5. There are a number of tools available to implement set-valued reachability computation $[\Phi]$ for neural networks such as those tools mentioned in [16,22], and [25].

For a dynamical system (1), the reachable set is defined by the following definition.

Definition 5. Given dynamical system (1) with initial set $\mathcal{X}_{(0)}$ and input set $\mathcal{U}$, the reachable set at time k is

$$\mathcal{X}_{(k)} = \left\{ x(k; x_0, u(\cdot)) \in \mathbb{R}^d \mid x_0 \in \mathcal{X}_{(0)},\ u(k) \in \mathcal{U} \right\}, \tag{12}$$

and the union of $\mathcal{X}_{(k)}$ over $[0, K]$ defined by

$$\mathcal{R}_{(K)} = \bigcup_{k=0}^{K} \mathcal{X}_{(k)}, \tag{13}$$

is the reachable set over time interval $[0, K]$.

When modeling with a single neural-network model Φ to approximate f in (1), the reachable set for model Φ during time interval $[0, K]$ can be expressed by

$$\mathcal{X}_{(k+1)} = [\Phi](\mathcal{X}_{(k)}, \mathcal{U}), \tag{14}$$

$$\mathcal{R}_{(K)} = \bigcup_{k=0}^{K} \mathcal{X}_{(k)}. \tag{15}$$

However, this reachable computation process usually represents high computational complexity due to the large size of the single neural network model Φ. In this paper, we intend to use a collection of small-size neural networks Φ_i, namely ELMs, in neural hybrid automaton model $\mathcal{H}$ to circumvent high computational cost in reachable set computation while maintaining modeling accuracy.

It is worth noting that this study makes use of reachable set computation to demonstrate the computational efficiency of neural hybrid automata. This benefit can be extended to other computationally expensive tasks.

2.3. Problem formulation

Our proposed modeling framework aims to model complex dynamical systems with neural hybrid automaton and provide set-valued reachability analysis. Particularly, by reducing the computational complexity in computationally expensive tasks such as training and verification, this framework will be competitive with conventional neural network modeling methods. However, applying the framework of neural hybrid automaton in modeling is challenging for the following problems.

Problem 1. Given samples of system trajectories and feature Θ of the system, how does one model the dynamical system with a neural hybrid automaton under Definition 3 through a data-driven process?

Since the multiple ELMs are trained to approximate the system dynamics, when it comes to the analysis of set-valued reachability, we need to handle the relationships between the input set and different partitions.

Problem 2. Given an initial set $\mathcal{X}_{(0)}$ and input $\mathcal{U}$ and neural hybrid automaton model $\mathcal{H}$, how does one develop a scalable reachable set computation method for neural hybrid automaton, especially for situations when the input set intersect multiple partitions in the feature space?

The above two problems are the main concerns of this paper. The remainder of this paper will aim to solve the above two problems in detail.

3. Neural hybrid automaton modeling

The detailed neural hybrid automaton modeling framework focusing on solving Problems 1 and 2 will be presented in this section. The details include mode clustering and dynamics learning, as well as the Split and Combine processes for reachability analysis, as shown in Fig. 2.

To begin with, the training data is segmented according to partition $\mathcal{P}$ for the subsequent learning procedures.

Definition 6. Given a training set $\hat{\mathcal{W}} = \{\mathbf{X}, \mathbf{T}\}$ in which $\mathbf{X} \in \mathbb{R}^{(d+n)\times q}$, $\mathbf{T} \in \mathbb{R}^{d\times q}$ are the input, output matrices for q samples, and pre-specified partitions $\tilde{\mathcal{P}} = \{\mathcal{P}_1, \dots, \mathcal{P}_M\}$ under feature function Θ, a collection of M segmented input–output data pair set $\mathcal{W}$ can be defined as

$$\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \dots, \mathcal{W}_M\}, \tag{16}$$

where any input–output pair $\{x_i, t_i\} \in \mathcal{W}_i$ satisfies

$$\Theta(x_i) \in \mathcal{P}_i,\ \forall \{x_i, t_i\} \in \mathcal{W}_i. \tag{17}$$

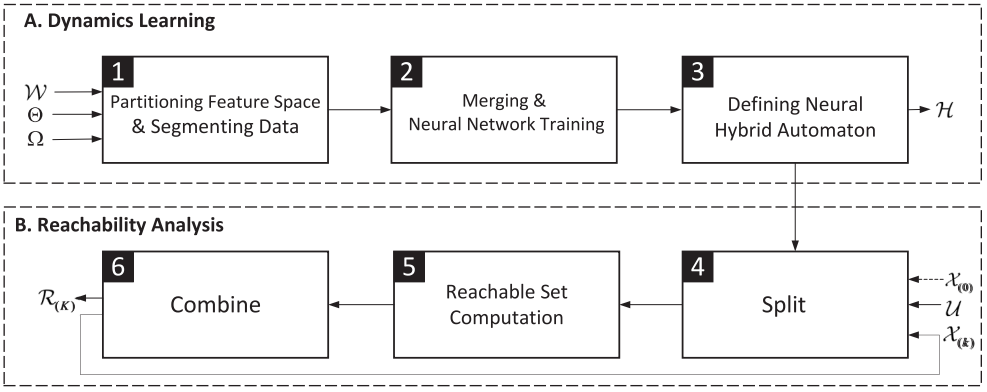


Fig. 2. Proposed framework for neural hybrid automaton modeling via ELM, including dynamics learning and reachability analysis. Specifically, the automaton approximates the dynamics through samples, while we analyze the reachable set with given initial set input and external set input via Split and Combine.

Remark 6. It is noted that the pre-specified partitions $\tilde{\mathcal{P}} = \{\mathcal{P}_1, \dots, \mathcal{P}_M\}$ such as the initial lattices of feature space are not the partitions in $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_N\}$ in a neural hybrid automaton model. There is a mode clustering process developed to generate an optimized partition $\mathcal{P}$ out of $\tilde{\mathcal{P}}$ in the modeling framework, and normally $N \ll M$.

By the definition of segmented training data sets $\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \dots, \mathcal{W}_M\}$, we are ready to use the proposed neural hybrid automaton model to learn complex dynamics.

3.1. Mode clustering and dynamics learning

First, a Pre-Processing procedure is then developed focusing on obtaining an initial partition $\tilde{\mathcal{P}} = \{\mathcal{P}_1, \dots, \mathcal{P}_M\}$ that are in the form of lattices.

Definition 7. Given system state set $\mathcal{X}$, a feature function Θ, and a training set $\hat{\mathcal{W}}$, a Pre-Processing procedure is to initialize the feature space $\hat{\mathcal{P}} = [\Theta](\mathcal{X})$ to a collection of lattices $\tilde{\mathcal{P}} = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_M\}$ that satisfies: (1) $\hat{\mathcal{P}} \subseteq \bigcup_{i=1}^{M} \mathcal{P}_i$; (2) $\mathcal{P}_i \bigcap \mathcal{P}_j = \emptyset,\ i \neq j,\ \forall i, j = 1, 2, \dots, M$; (3) The data set $\hat{\mathcal{W}}$ is segmented into $\mathcal{W} = \{\mathcal{W}_1, \dots, \mathcal{W}_M\}$.

Remark 7. There are a variety of ways to implement the Pre-Processing procedure. In this paper, we use a bisection-based method [43], where the input space is in the form of interval set $\mathcal{X} = [\underline{\mathcal{X}}, \overline{\mathcal{X}}] \subseteq \mathbb{R}^d$, we set a tolerance coefficient $\lambda \geq 1$ to ensure at least q_i samples for $\mathcal{W}_i$ are selected in segmented data. Pre-processing can be written in pseudo code given in Algorithm 1 in [44], with respect to $d(\mathcal{P}_{i,k})$ denotes the length of kth dimension for ith lattice (interval), namely, $d(\mathcal{P}_{i,k}) = \overline{\mathcal{P}}_{i,k} - \underline{\mathcal{P}}_{i,k}$. This process is influenced by the dimensionality of the feature space, namely, the higher the dimension of the feature space, the more lattices will be divided, and the computation time will be longer.

In neural hybrid automaton $\mathcal{H}$, we aim to train the ELMs while mitigating over-fitting. To mitigate over-fitting [45] of trained neural network in approximating complex dynamics, there is a tradeoff between the complexity of the neural network model namely, layers and neurons, and the amount of training data. In this paper, since a simple neural network may lead to worse modeling accuracy, we prefer to assume that sufficient data is available for training ELM in each $\mathcal{W}_i$ for ELMs with proper structure complexities.

Under Assumption 1, the segmented data will provide sufficient training samples for ELMs learning dynamics.

Assumption 1. Given a collection of segmented data sets $\mathcal{W} = \{\mathcal{W}_1, \dots, \mathcal{W}_M\}$, we assume that each $\mathcal{W}_i$ contains at least q_i samples which are sufficient to avoid over-fitting for the ELM-based learning.

After Pre-Processing and under Assumption 1, the next step is to train multiple ELMs to approximate the local system dynamical behaviors with segmented data set via the following optimization problem

$$\min_{\beta_i \in \mathbb{R}^{L \times d}} \left\| \mathbf{H}_i \beta_i - \mathbf{T}_i \right\| \tag{18}$$

in which $\mathbf{H}_i$ denotes the randomized hidden layer output matrix of Φ_i with the input as $\mathbf{X}_i$, while $\mathbf{T}_i$ is the target output matrix from $\mathcal{W}_i$, $i = 1, 2, \dots, M$. The training error can be obtained under (9).

The above training process will result in M ELMs trained as dynamical descriptions for M lattices, which is unacceptable for the majority of $\mathcal{H}$ modeling cases since the number of lattices is big. Therefore, merging or clustering obtained modes is the key to modeling a more realistic $\mathcal{H}$. A mode clustering procedure will be developed to obtain the appropriate partitions by merging the lattices.

Considering two segmented training set $\mathcal{W}_i$, $\mathcal{W}_j$, $i \neq j$ for ith and jth lattices, a given ELM structure Φ, namely the input weight matrix, activation function, and bias are given, the training process is formulated as

$$\min_{\beta_{i,j} \in \mathbb{R}^{L \times d}} \left\| \mathbf{H}_{i,j} \beta_{i,j} - \mathbf{T}_{i,j} \right\|, \tag{19}$$

in which $\mathbf{H}_{i,j}$, $\mathbf{T}_{i,j}$ are the combined hidden layer output matrix and target output matrix, which are defined in the following form of

$$\mathbf{H}_{i,j} = \begin{bmatrix} \mathbf{H}_i \\ \mathbf{H}_j \end{bmatrix},$$

$$\mathbf{T}_{i,j} = \begin{bmatrix} \mathbf{T}_i \\ \mathbf{T}_j \end{bmatrix}.$$

After we obtain the $\beta_{i,j}$, since the input weight matrix and the bias vectors of the ELM are given, $\Phi_{i,j}$ will be obtained. Given a training error tolerance γ as the mode merge criterion such that if

$$MSE(\Phi_{i,j}(\mathbf{X}_{i,j}) - \mathbf{T}_{i,j}) \leq \gamma \tag{20}$$

holds, then the two lattices will be considered to have similar dynamical behaviors that can be described by a common neural network $\Phi_{i,j}$, hence their corresponding lattices $\mathcal{P}_i$ and $\mathcal{P}_j$ can be merged, and expected to be governed by the same ELM. The process of merging different lattices is called Merging. After Merging, the number of partitions will be reduced and the set of lattices will be the set of partitions of our neural hybrid automaton model. The Merging and Learning process is given in Algorithm 1.

Remark 8. We merge the redundant lattices that are in a similar mode under training performances for their corresponding ELMs and obtain the partitions $\mathcal{P}$ that subsequently define transitions, invariants, and guards for the neural hybrid automaton model after Merging, i.e., the transition $\mathcal{P}_i \to \mathcal{P}_j$ can be abstracted if there exist system state $x(k) \in \mathcal{P}_i$ and successive state $x(k+1) \in \mathcal{P}_j$, as aforementioned in Remarks 3

Algorithm 1: Mode Merging and Dynamics Learning

Input : Set of Lattices $\{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_M\}$; Set of Samples $\{\mathcal{W}_1, \mathcal{W}_2, \dots, \mathcal{W}_M\}$; Training Error Tolerance γ.
Output: Merged Partitions $\mathcal{P} = \{\mathcal{P}_1 \dots, \mathcal{P}_N\}$; ELMs $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_1, \dots, \boldsymbol{\Phi}_N\}$.

$\ell \leftarrow M$, $N \leftarrow 1$;
/* Segmented partitions merge */
while $N < \ell$ **do**
  $n \leftarrow 1$;
  while $n \leq \ell$ **do**
    $n \leftarrow n + 1$;
    Solve $\min_{\beta_{N,n} \in \mathbb{R}^{L \times d}} \|\mathbf{H}_{N,n}\beta_{N,n} - \mathbf{T}_{N,n}\|$;
    if $MSE(\boldsymbol{\Phi}_{N,n}(\mathbf{X}_{N,n}) - \mathbf{T}_{N,n}) \leq \gamma$ **then**
      $\mathcal{P}_N \leftarrow \{\mathcal{P}_N, \mathcal{P}_n\}$;
      $\ell \leftarrow \ell - 1$;
    end
  end
  $N \leftarrow N + 1$;
end
/* Generate ELM models */
$i \leftarrow 1$;
while $i \leq N$ **do**
  Solve $\min_{\beta \in \mathbb{R}^{L \times d}} \|\mathbf{H}_i\beta_i - \mathbf{T}_i\|$ to obtain $\boldsymbol{\Phi}_i$
end
return $\mathcal{P} = \{\mathcal{P}_1 \dots, \mathcal{P}_N\}$; $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_1, \dots \boldsymbol{\Phi}_N\}$.

and 4. After mode clustering and dynamics learning, we will be able to obtain the explicit $\mathcal{H}$ for complex dynamics.

3.2. Reachability analysis via Split and Combine

In our neural hybrid automaton framework, a collection of ELMs $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_1, \dots, \boldsymbol{\Phi}_N\}$ are employed, and the state space $\mathcal{X}$ are partitioned into a family of subsets $\Gamma = \{\mathcal{X}_1, \dots, \mathcal{X}_N\}$ in which $\mathcal{X}_i$ is determined by partition $\mathcal{P}$ and defined by (4). The reachable set computation of neural hybrid automaton $\mathcal{H}$ can be reduced to the output set computation of each ELM, which is summarized by the following theorem.

Theorem 1. *Given a neural hybrid automaton $\mathcal{H}$ with an initial set $\mathcal{X}_{(0)} \in \mathcal{X}$ and an input set $\mathcal{U}$, the reachable set $\mathcal{X}_{(k)}$ can be computed recursively as*

$$\mathcal{X}_{(k+1)} = \bigcup_{i=0}^{N} [\boldsymbol{\Phi}_i](\hat{\mathcal{X}}_{(k),i}, \mathcal{U}), \tag{21}$$

where $\hat{\mathcal{X}}_{(k),i}$ is the intersection of $\mathcal{X}_{(k)}$ and $\mathcal{X}_i$, i.e.,

$$\hat{\mathcal{X}}_{(k),i} = \mathcal{X}_{(k)} \cap \mathcal{X}_i, \ k = 0, 1, \dots, K, \tag{22}$$

and the reachable set of $\mathcal{H}$ over $[0, K]$ is

$$\mathcal{R}_{(K)} = \bigcup_{k=0}^{K} \mathcal{X}_{(k)}. \tag{23}$$

Proof. Let us consider set $\mathcal{X}_{(k)}$ at time instant k, the following result can be obtained

$$\begin{aligned} \bigcup_{i=0}^{N} \hat{\mathcal{X}}_{(k),i} &= \bigcup_{i=0}^{N} (\mathcal{X}_{(k)} \cap \mathcal{X}_i) \\ &= \left(\bigcup_{i=0}^{N} \mathcal{X}_i \right) \cap \mathcal{X}_{(k)} \\ &= \mathcal{X} \cap \mathcal{X}_{(k)}, \end{aligned}$$

which, due to $\mathcal{X}_{(k)} \in \mathcal{X}$, leads to

$$\bigcup_{i=0}^{N} \hat{\mathcal{X}}_{(k),i} = \mathcal{X}_{(k)}. \tag{24}$$

For each $\hat{\mathcal{X}}_i$, the ELM $\boldsymbol{\Phi}_i$ is associated with it for state evolution to $k+1$, therefore one has

$$\mathcal{X}_{(k+1)} = \bigcup_{i=0}^{N} [\boldsymbol{\Phi}_i](\hat{\mathcal{X}}_{(k),i}, \mathcal{U}). \tag{25}$$

As a result, reachable set $\mathcal{R}_{(K)}$ can be obtained by (23) according to Definition 5. The proof is completed. □

Remark 9. Two points concerned with computational complexity and efficiency are addressed:

- From (21) in Theorem 1, it implies that it requires to compute the output sets of a collection of ELMs $\boldsymbol{\Phi}_i$, $i = 1, \dots, N$ because the input sets of ELMs may intersect multiple partitions in the state space. However, in practice, we only need to consider a very small number of $\hat{\mathcal{X}}_{(k),i}$ since the reachable set $\mathcal{X}_{(k)}$ only intersects with very few of state partitions $\mathcal{X}_i$ and the majority of $\hat{\mathcal{X}}_{(k),i}$ satisfy $\hat{\mathcal{X}}_{(k),i} = \emptyset$ in (22). Therefore, the employment of multiple local models in proposed neural hybrid automata will not cause a significant increase in computation complexity. On the other hand, the employment of multiple small neural networks will significantly enhance computational efficiency in reachable set computation.
- Considering that the computational complexity of reachable set computation heavily relies on the complexity of the neural network, i.e., with a larger number of layers and neurons, the computation for reachable sets will be more complex hence more time will be consumed. Compared with modeling dynamical systems using one single large-scale neural network to approximate the global system dynamics, a small number of neurons for each ELM are used in our neural hybrid automaton framework, since they are sufficient to approximate local system behaviors instead of global dynamics. This core feature is the key to enable computational efficiency in set-valued reachability analysis for neural hybrid automaton.

To implement Theorem 1 in reachable set computation for a neural hybrid automaton $\mathcal{H}$, we need to split the current reachable set $\mathcal{X}_{(k)}$ as shown in (22) and combine the output reachable set of ELMs as in (21) for solving Problem 2.

First, as discussed in Theorem 1, state space $\mathcal{X}$ are partitioned into a family of subsets $\Gamma = \{\mathcal{X}_1, \dots, \mathcal{X}_N\}$ in which $\mathcal{X}_i$ is determined by partition $\mathcal{P}$ and defined by (4). Then, the reachable set $\mathcal{X}_{(k)}$ may intersect multiple partitions in Γ, which means there will be a splitting computation of the reachable set for each intersection.

Definition 8. For a reachable set $\mathcal{X}_{(k)}$ of a neural hybrid automaton $\mathcal{H}$ with a partition $\mathcal{P}$, a family of subsets $\hat{\mathcal{X}}_{(k)} = \{\hat{\mathcal{X}}_{(k),1}, \dots, \hat{\mathcal{X}}_{(k),N}\}$, where $\hat{\mathcal{X}}_{(k),i} = \mathcal{X}_{(k)} \cap \mathcal{X}_i$ and $\bigcup_{i=1}^{N} \hat{\mathcal{X}}_{(k),i} = \mathcal{X}_{(k)}$. Given an input set $\mathcal{U}$, the reachable set computation based on $\hat{\mathcal{X}}_{(k)}$ can be performed as

$$\mathcal{X}_{(k+1),i} = [\boldsymbol{\Phi}_i](\hat{\mathcal{X}}_{(k),i}, \mathcal{U}). \tag{26}$$

The process of computing reachable set $\mathcal{X}_{(k+1),i}$, $i = 1, \dots, N$, is called Split.

After the Split, the Combine process is needed to obtain a complete reachable set for the next step.

Definition 9. Given $\mathcal{X}_{(k+1),i}$, $i = 1, \dots, N$, the reachable set $\mathcal{X}_{(k+1)}$ of neural hybrid automaton $\mathcal{H}$ at time step $k+1$ is computed by

$$\mathcal{X}_{(k+1)} = \bigcup_{i=1}^{N} \mathcal{X}_{(k+1),i} \tag{27}$$

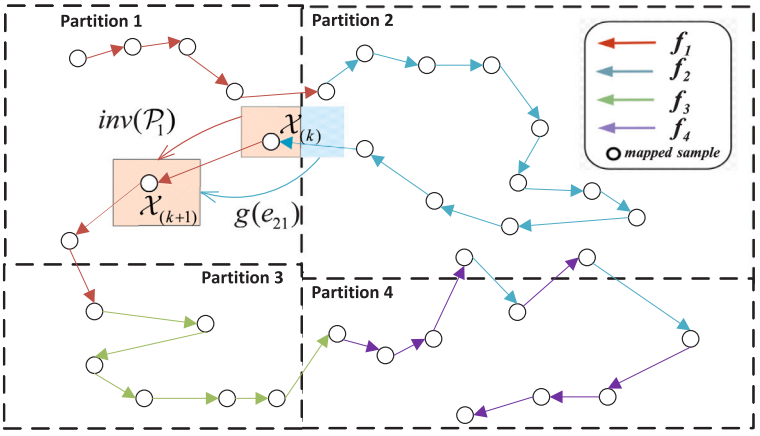


Fig. 3. The samples are segmented based on feature and partitions, in neural hybrid automaton modeling, neural networks only need to approximate $\{f_1, f_2, f_3, f_4\}$ respectively, instead of learning the global dynamics, this helps to simplify the configuration of sub-neural networks. As for reachability analysis from $\mathcal{X}_{(k)}$ to $\mathcal{X}_{(k+1)}$, Split process will serve as guard function $g(e_{21})$ and the invariant function $inv(\mathcal{P}_1)$.

Algorithm 2: Pseudo Code for Split and Combine

Input : Reachable set $\mathcal{X}_{(k)}$; Neural Hybrid Automaton $\mathcal{H}$.
Output: Output Reachable Set $\mathcal{X}_{(k+1)}$.

$i \leftarrow 1$
/* Split by Definition 8 */
while $i \leq N$ **do**
  if $\mathcal{X}_{(k)} \cap \mathcal{X}_i \neq \emptyset$ **then**
    $\hat{\mathcal{X}}_{(k),i} \leftarrow \mathcal{X}_{(k)} \cap \mathcal{X}_i$
    $\mathcal{X}_{(k+1),i} \leftarrow [\Phi_i](\hat{\mathcal{X}}_{(k),i}, \mathcal{U})$
  else
    $\mathcal{X}_{(k+1),i} \leftarrow \emptyset$
  end
  $i \leftarrow i + 1$
end
/* Combine by Definition 9 */
$\mathcal{X}_{(k+1)} \leftarrow \bigcup_{i=1}^{N} \mathcal{X}_{(k+1),i}$
return $\mathcal{X}_{(k+1)}$

by which the Combine process derives the reachable set at $k+1$ time instant.

With the Split and Combine defined above, the reachable set of $\mathcal{H}$ can be paralleling computed at time instance k if $\mathcal{X}_{(k)}$ intersects with multiple partitions in state space. The Split and Combine process is detailed in Algorithm 2.

In summary, neural hybrid automaton modeling has the advantages of simplifying the learning model and mitigating overfitting via learning the dynamics within localized subsystems. For example, the configuration of sub-neural networks can be simplified due to the idea that mapping within a specific partition contains less information, i.e., in Fig. 3 instead of approximating f neural hybrid automaton model approximates $\{f_1, f_2, f_3, f_4\}$ through segmented input–output data pair set $\mathcal{W}$ with a set of ELMs Φ. As for reachability analysis, the Split process will work efficiently as the guard and invariant function for reachable set computation, while the Combine process obtains the complete reachable set.

4. Application to modeling of complex dynamics

In this section, modeling cases of classical limit cycle and the human handwritten motions with our proposed neural hybrid automaton will be given to illustrate our framework. We will show the advantage of our proposed modeling framework by verifying our neural hybrid system model and the conventional neural network model through analysis of set-valued reachability.

4.1. Modeling of limit cycle dynamics

A numerical example of the limit cycle borrowed from [46] is used to validate our approach. The dynamical system model is given in the form of

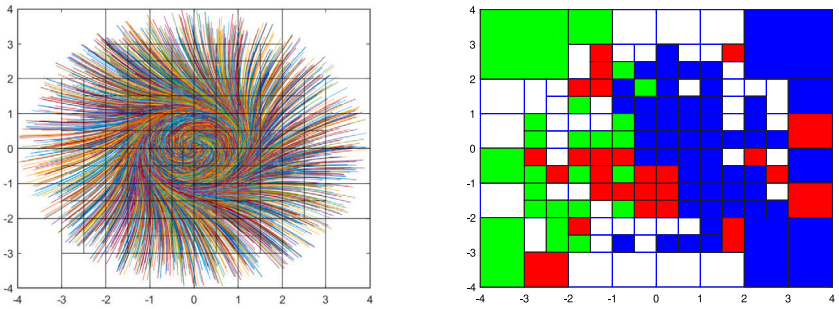
$$\begin{aligned} r(k+1) &= (1+\tau)r(k) - \tau r^3(k) + \tau u(k) \\ \theta(k+1) &= \theta(k) + \tau\omega \\ u(k) &= \mu + \delta\zeta(k) \end{aligned} \tag{28}$$

where $\omega = 2\pi/3$ and $\tau = 0.1$ are the angular velocity and time step width, respectively. The uniform random number $\zeta(k) \sim U(-1, 1)$. Namely, the external input $u(k) \sim U(\mu - \delta, \mu + \delta)$ ($\mu = 0.2$ and $\delta = 1.5$) in which U denotes uniform distribution.

We generate the samples from the given model (28) with random initial conditions when $r(0) \in [-4, 4]$, $\theta(0) \in [-\pi, \pi]$ with (28). The illustration of 150 trajectories is given in Fig. 4(a) which can be used as references for training as well as evaluation of modeling performance. Considering the samples generated from the dynamics, the input region for $\mathcal{H}$ in the form of interval is $\mathcal{X} = [\underline{\mathcal{X}}, \overline{\mathcal{X}}]$ where $\overline{\mathcal{X}} = [4, 4, 1.7]^T$ and $\underline{\mathcal{X}} = [-4, -4, -1.3]^T$. The modeling procedure of $\mathcal{H}$ is summarized as follows:

- Determine the structure of ELMs, which have 20 ReLU neurons for each ELM.
- In the Pre-processing, it generates the initial partitions with 149 lattices for the successive modeling of $\mathcal{H}$.
- Then, by the mode clustering, i.e., the Merging process, the number of partitions is reduced to 50 with a collection of 50 ELMs $\Phi = \{\Phi_1, \ldots, \Phi_9\}$.
- Determine the transitions among partitions based on the partition Merging result.

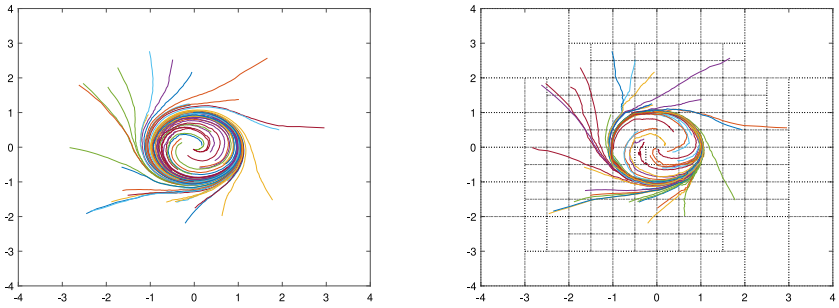
To compare our approach with a single neural network modeling approach, a single-ELM model Φ which has 200 hidden neurons with ReLU as activation functions is also trained for comparison. The trajectories with random initial conditions of the proposed hybrid automaton $\mathcal{H}$ and a single-ELM model are given in Fig. 5. There are 50 trajectories with 200 steps randomly generated from random initial states satisfying $x_1(0) \in [-4, 4]$, $x_2(0) \in [-4, 4]$, and $u(k) \in [-1.3, 1.7]$ in Fig. 6. The average training times and MSE performances of twenty times are given in Table 1. It can be explicitly observed that the neural



(a) Trajectory Samples and the pre-specified lattices.

(b) Merging of the redundant lattices.

Fig. 4. (a) Trajectories of the limit cycle with random initial condition $r(0) \in [-4, 4]$, $\theta(0) \in [-\pi, \pi]$ each of which contains 150 samples and the input $u \sim U(-1.3, 1.7)$ (b) Pre-processing the limit cycle model (blue lattices) and merged partitions (green, red and blue squares, respectively).



(a) Single Neural Network Model.

(b) Neural Hybrid Automaton.

Fig. 5. Comparison of trajectories given random initial conditions between a single-ELM-based model and our proposed hybrid automaton model, which indicates that both learning systems share a similar dynamical behavior given random initial conditions.

Table 1
Model performance and computation time comparison.

Method	MSE	Training	Reachable set
Single neural network	0.0817	5.28 s	1.0062×10^4 s
Neural hybrid automaton	0.0519	0.36 s	956.0135 s

hybrid automaton model $\mathcal{H}$ with 20 neurons in each ELM can very well approximate the original system behaviors with better accuracy and much less training time other than the single neural network model with 200 neurons.

The comparison of set-valued reachability analysis using NNV in [25] between $\mathcal{H}$ and the single neural network model is shown in Fig. 6. It can be observed that the neural hybrid automaton model can produce a similar reachable set with a single neural network, which is much more preferable in safety verification for real-time applications. Notably, the computation time of reachable set computation has been significantly reduced compared with the single neural network model, i.e., $\mathcal{H}$ only needs 9.5% of time consumption of the single neural network model to perform the same 200-step reachable set computation.

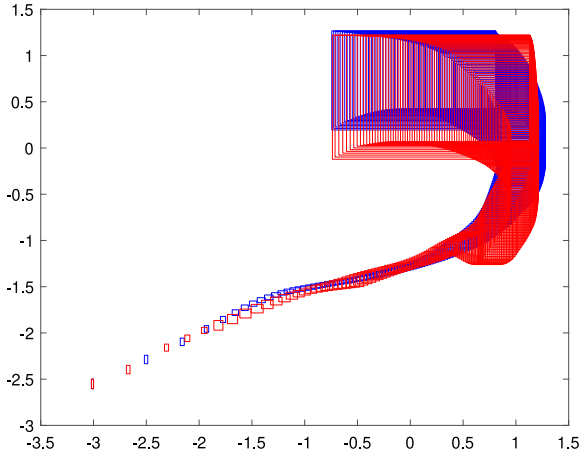


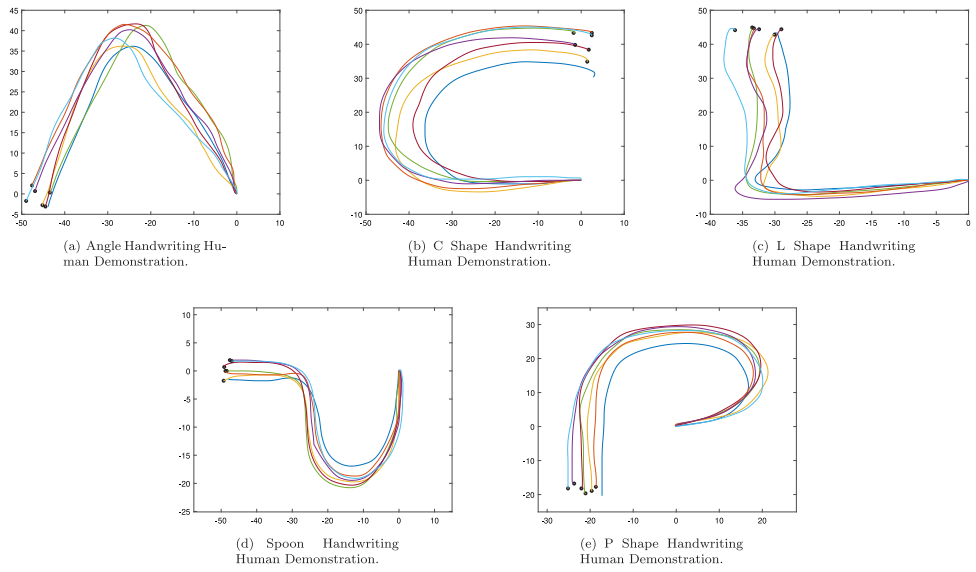
Fig. 6. Set-valued reachability analysis for 200 steps of a single neural network model (blue) and our proposed neural hybrid automaton model (red) given initial condition $x_1 \in [-3.02, -3]$, $x_2 \in [-2.603, -2.5]$.

4.2. Modeling of human handwritten motions

Learning-based methods have been promoted as an effective way to model motion dynamics [47,48]. Consider a modeling problem for Human handwritten motions, i.e., modeling human writing behaviors using LASA data set[1] [49] which contains 20 handwriting demonstrations of humans. Each handwriting consists of 7 trajectories with 1000 discrete-time samples of human demonstrations, some examples are shown in Fig. 7.

We assume that there exists a nonlinear function $f : \mathcal{X} \to \mathcal{X}$ that describes the motions of different human handwriting patterns, and thus f can be approximated by one or several neural networks that are trained based on the samples. This is a typical task in human behavior modeling that relies on the data and requires no prior system knowledge. However, approximating f is challenging due to the following specifics:

[1] The LASA dataset is a library of 2D human handwriting motions available at: https://cs.stanford.edu/people/khansari/download.html.



(a) Angle Handwriting Human Demonstration.
(b) C Shape Handwriting Human Demonstration.
(c) L Shape Handwriting Human Demonstration.
(d) Spoon Handwriting Human Demonstration.
(e) P Shape Handwriting Human Demonstration.

Fig. 7. Human handwriting demonstrations of Angle, C shape, L shape, Spoon, P shape from LASA data set [49], demonstrations contain seven trajectories, each of which has 1000 location samples.

Table 2
MSE and computation time of single neural network model.

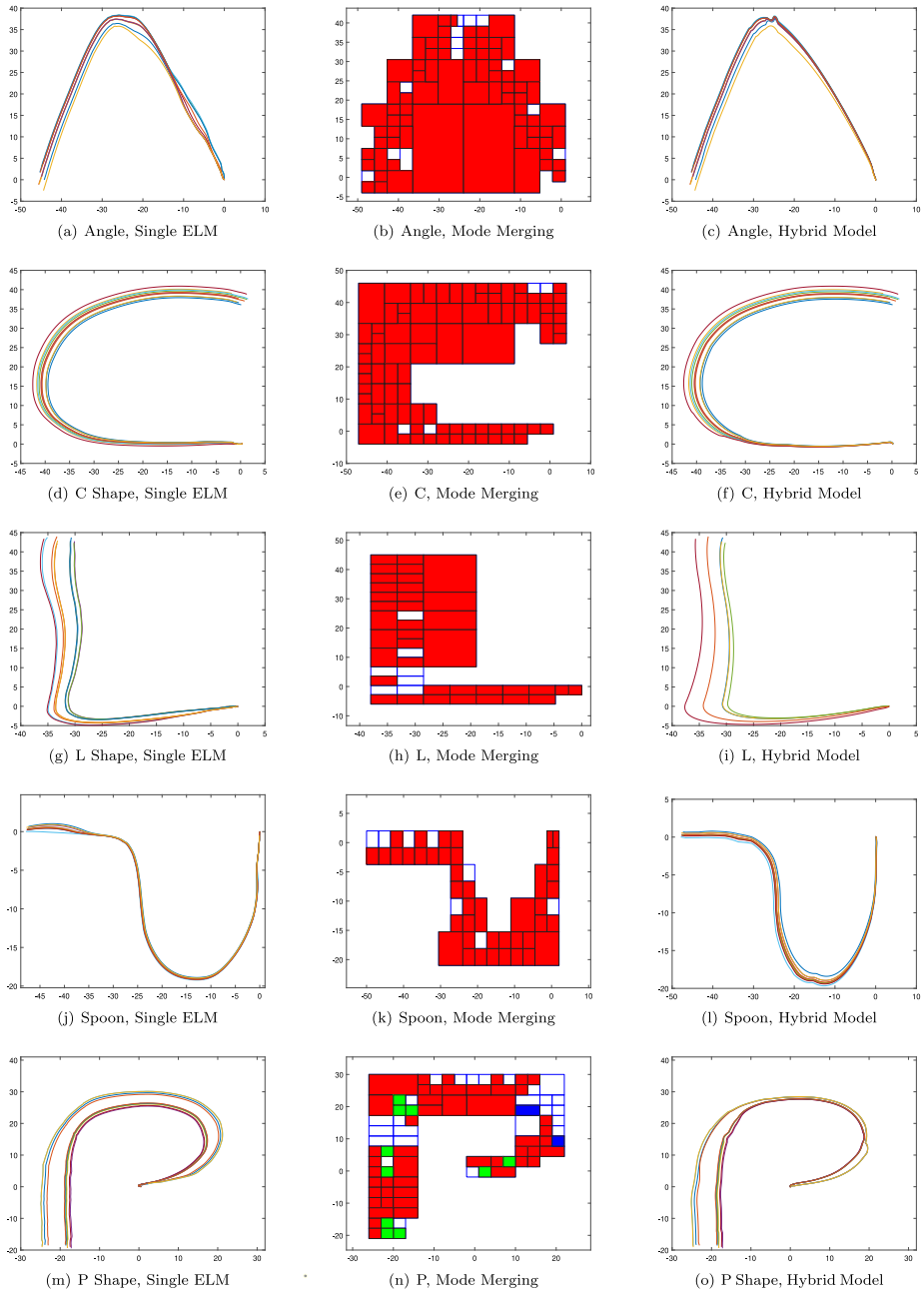
Data set	MSE	Training	Reachable set
Angle	2.739×10^{-4}	4.50×10^{-2} s	3.6466×10^{4} s
C Shape	2.375×10^{-4}	4.79×10^{-2} s	9.0068×10^{4} s
L Shape	1.972×10^{-4}	4.37×10^{-2} s	9.7783×10^{4} s
Spoon	1.767×10^{-4}	4.30×10^{-2} s	8.4465×10^{4} s
P Shape	2.301×10^{-4}	4.50×10^{-2} s	8.5201×10^{4} s

- Given the unpredictable behavior of humans, f is difficult to define mathematically, and therefore, learning-based methods may be effective in approximating f.
- f depends on demonstrations with small sample sizes, a complex neural network, such as DNN may not get well-trained.
- Trajectories of demonstrations are determined by humans with sudden changes in some local area, which suggests that one single generic model may not be able to capture these complex dynamics.

After approximating f with the help of neural networks, we need to verify whether the model performs well when the initial condition is perturbed, namely, by analyzing the dynamical behavior of the model by giving an initial input set. The performance of a model from the perspective of verification, e.g., time consuming, accuracy, etc., is of great importance in set-valued reachability analysis. Based on [15,22], the scale of a neural network can significantly influence the performance of model verification. In other words, choosing localized and small-scale neural networks to approximate the dynamics will benefit the model from a verification point of view. Hence, our neural hybrid automaton modeling framework will be applied to illustrate its effectiveness.

The modeling process and comparison results are summarized as follows:

- We will demonstrate our proposed framework using the LASA data set, in which ELMs are with 20 ReLU-activated neurons with randomized input weight matrix and bias vector as the structure of the model. To demonstrate the effectiveness of our modeling framework, an ELM with a single hidden layer that contains 200 ReLU-activated neurons are trained as a single-neural-network reference model.
- Samples of different handwriting demonstrations will be modeled by both a neural hybrid automaton and a single neural network. The state space $\mathcal{X}$ for our neural hybrid automaton $\mathcal{H}$ will be an interval determined by the maximum and minimum value of training data. For the 2-dimensional state space, the feature function will be a linear function with an identity matrix. Then the Pre-processing and Merging will obtain partitions, which subsequently define invariants and transitions. The local dynamics are also learned and represented by a collection of ELMs. The partitions after Pre-processing and Merging are given in the middle of Fig. 8. For example, there are three merged partitions of P shape modeling with our proposed automaton, which are marked in green, blue, and red squares, respectively.
- After training our proposed model under partitions, different initial conditions are given to compare the trajectories between the single neural network model and neural hybrid automaton model. In Fig. 8, a detailed illustration of the 1000-step-trajectories of the single neural network models is given on the left, while the trajectories of our proposed neural hybrid automaton models are given on the right. The trajectories of the neural hybrid automaton models and single neural network models are similar by observing Fig. 8, which indicates the two models share similar patterns, but it is noted that the neural hybrid automaton model has much fewer neurons.
- Comparing with Fig. 7, both learning systems are able to reflect the patterns of the system given random initial conditions. We can perform a set-valued reachability analysis of $\mathcal{H}$ and the single neural network models as given in Fig. 9. Our proposed modeling framework has an advantage in reachability analysis compared with a single neural network model, the average MSE performance and the computation time of twenty times comparison are given in Tables 2 and 3, respectively, indicating that their computational costs of reachability analysis are significantly reduced while maintaining similar training accuracy levels as single-ELM models. The training time can be reduced as well when we use neural hybrid automaton models.
- When modeling with our neural hybrid automaton, there will be circumstances that the states of $\mathcal{H}$ in the feature space run out of the partitions, namely, $x(k) \notin \mathcal{P}$. In these cases, we will be using the single-ELM model as a backup model to handle the part of trajectories that are outside the partitions as well as for the set-valued reachability analysis for the reachable sets that are outside the partitions.



(a) Angle, Single ELM (b) Angle, Mode Merging (c) Angle, Hybrid Model

(d) C Shape, Single ELM (e) C, Mode Merging (f) C, Hybrid Model

(g) L Shape, Single ELM (h) L, Mode Merging (i) L, Hybrid Model

(j) Spoon, Single ELM (k) Spoon, Mode Merging (l) Spoon, Hybrid Model

(m) P Shape, Single ELM (n) P, Mode Merging (o) P Shape, Hybrid Model

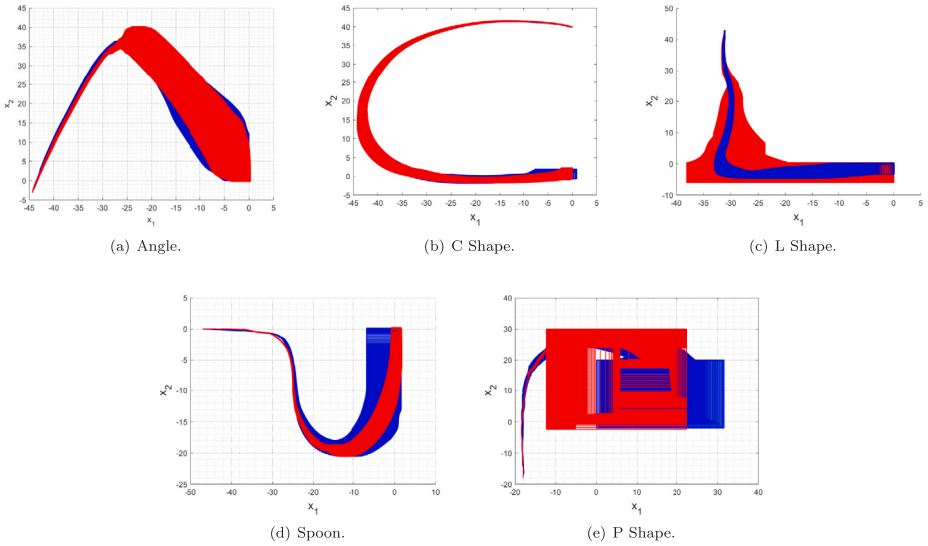
Fig. 8. Simulations of Single-ELM models are on the left. Pre-processing and Merging process is in the middle. Simulations of hybrid models are on the right.

Table 3
MSE and computation time of neural hybrid automaton.

Data set	MSE	Training	Reachable set
Angle	4.787×10^{-4}	8.60×10^{-3} s	503.0428 s
C Shape	8.323×10^{-4}	1.25×10^{-2} s	3308.2473 s
L Shape	4.175×10^{-4}	7.05×10^{-3} s	1513.7453 s
Spoon	4.643×10^{-4}	7.57×10^{-3} s	69.9705 s
P Shape	9.484×10^{-4}	4.71×10^{-3} s	127.3636 s

In summary, the advantages of our proposed neural hybrid automaton framework are well-demonstrated by modeling examples in

- Mitigating overfitting: pre-processing procedure divides the state space into multiple pre-specified lattices based on the localized information in the form of sample numbers. According to UAT and under Assumption 1, the training of ELMs is able to mitigate overfitting. In practice, q_i in Assumption 1 can be easily tuned according to the configuration of ELMs.
- Scalability improved and computationally efficient: due to the distributed configuration of neural hybrid automaton, simple ELMs instead of a general complex one will be activated and involved in computing, which will significantly improve the model scalability. The distributed learning framework allows parallel training and verification of multiple simple ELMs will promote computation efficiency.
- Fast reachability verification: fast set-valued reachability analysis for our neural hybrid automaton is possible for the Split and Combine processes will activate simple ELMs based on invariants and allow parallel computing.



(a) Angle. (b) C Shape. (c) L Shape.

(d) Spoon. (e) P Shape.

Fig. 9. Set-valued reachability analysis of $\mathcal{H}$ (red) and the single-ELM model (blue) for 1000 time steps for the models of Angle, C shape, L shape, Spoon, and P shape.

5. Conclusion and future work

In this paper, a neural hybrid automaton modeling framework including, pre-processing, mode clustering & dynamics learning via ELM, and set-valued reachability analysis through Split and Combine is developed to model complex dynamical systems. The proposed learning framework is able to model the dynamics with multiple ELMs while maintaining accuracy in approximating the dynamics compared with the conventional neural network model. Besides, the scalability of neural hybrid automaton will improve due to parallel computing in training and verification.

This computationally efficient learning framework offers the advantage of improving the scalability of the learning model while mitigating overfitting. Further research is recommended to explore its potential in

- A novel feature function that can capture the key information of the dynamics and improve the performance of neural hybrid automaton in high-dimensional modeling.
- A novel partitioning method in pre-processing and mode clustering, to better approximate the dynamical system with a simple neural network structure, subsystems can be divided according to entropy or with prior information from the system.
- Neural hybrid automaton learning with other state-of-the-art learning methods such as DNN, radial basis function neural network, kernel ELM, etc.

CRediT authorship contribution statement

Tao Wang: Methodology. **Yejiang Yang:** Investigation, Writing – original draft, Software. **Weiming Xiang:** Conceptualization, Formal analysis, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Weiming Xiang reports financial support was provided by National Science Foundation. Weiming Xiang reports a relationship with National Science Foundation that includes: funding grants.

Data availability

Data will be made available on request.

Acknowledgments

This work was partially supported by the National Science Foundation, under NSF CAREER Award no. 2143351, NSF CNS Award no. 2223035, and NSF IIS Award no. 2331938.

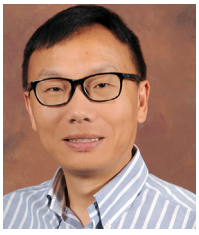
References

[1] J.E. Sierra-García, M. Santos, Switched learning adaptive neuro-control strategy, Neurocomputing 452 (2021) 450–464.
[2] Y. Qi, X. Zhao, J. Huang, Data-driven event-triggered control for switched systems based on neural network disturbance compensation, Neurocomputing 490 (2022) 370–379.
[3] N. Kumar, M. Rani, Neural network-based hybrid force/position control of constrained reconfigurable manipulators, Neurocomputing 420 (2021) 1–14.
[4] X. Li, X. Wang, X. Zheng, Y. Dai, Z. Yu, J.J. Zhang, G. Bu, F.-Y. Wang, Supervised assisted deep reinforcement learning for emergency voltage control of power systems, Neurocomputing 475 (2022) 69–79.
[5] S.-W. Fu, Y. Tsao, X. Lu, SNR-aware convolutional neural network modeling for speech enhancement, in: Interspeech, 2016, pp. 3768–3772.
[6] A. Mollalo, K.M. Rivera, B. Vahedi, Artificial neural network modeling of novel coronavirus (COVID-19) incidence rates across the continental United States, Int. J. Environ. Res. Public Health 17 (12) (2020) 4204.
[7] M. Gridach, Hybrid deep neural networks for recommender systems, Neurocomputing 413 (2020) 23–30.
[8] J. Zhang, Y. Luo, Z. Tao, J. You, Graphic-processable deep neural network for the efficient prediction of 2D diffractive chiral metamaterials, Appl. Opt. 60 (19) (2021) 5691–5698.
[9] C.P. Chen, Z. Liu, Broad learning system: An effective and efficient incremental learning system without the need for deep architecture, IEEE Trans. Neural Netw. Learn. Syst. 29 (1) (2017) 10–24.
[10] Y. Yang, T. Wang, J.P. Woolard, W. Xiang, Guaranteed approximation error estimation of neural networks and model modification, Neural Netw. 151 (2022) 61–69.
[11] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[12] O.A. Beg, H. Abbas, T.T. Johnson, A. Davoudi, Model validation of PWM DC-DC converters, IEEE Trans. Ind. Electron. 64 (9) (2017) 7049–7059.
[13] M. Poli, S. Massaroli, L. Scimeca, S. Chun, S.J. Oh, A. Yamashita, H. Asama, J. Park, A. Garg, Neural hybrid automata: Learning dynamics with multiple modes and stochastic transitions, Adv. Neural Inf. Process. Syst. 34 (2021).
[14] X. Yang, O.A. Beg, M. Kenigsberg, T.T. Johnson, A framework for identification and validation of affine hybrid automata from input-output traces, ACM Trans. Cyber-Phys. Syst. (TCPS) 6 (2) (2022) 1–24.
[15] W. Xiang, H.-D. Tran, X. Yang, T.T. Johnson, Reachable set estimation for neural network control systems: A simulation-guided approach, IEEE Trans. Neural Netw. Learn. Syst. 32 (5) (2020) 1821–1830.
[16] W. Xiang, H.-D. Tran, T.T. Johnson, Output reachable set estimation and verification for multilayer neural networks, IEEE Trans. Neural Netw. Learn. Syst. 29 (11) (2018) 5777–5783.

[17] Y. Yang, W. Xiang, Robust optimization framework for training shallow neural networks using reachability method, in: 2021 60th IEEE Conference on Decision and Control (CDC), 2021, pp. 3857–3862.
[18] S. Singh, V. Sindhwani, J.-J.E. Slotine, M. Pavone, Learning stabilizable dynamical systems via control contraction metrics, in: International Workshop on the Algorithmic Foundations of Robotics, Springer, 2018, pp. 179–195.
[19] H. Tsukamoto, S.-J. Chung, Neural contraction metrics for robust estimation and control: A convex optimization approach, IEEE Control Syst. Lett. 5 (1) (2020) 211–216.
[20] K. Neumann, A. Lemme, J.J. Steil, Neural learning of stable dynamical systems based on data-driven Lyapunov candidates, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 1216–1222.
[21] M. Fazlyab, A. Robey, H. Hassani, M. Morari, G. Pappas, Efficient and accurate estimation of Lipschitz constants for deep neural networks, in: Advances in Neural Information Processing Systems, 2019, pp. 11427–11438.
[22] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, J.Z. Kolter, Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification, 2021, arXiv.
[23] O. Thapliyal, I. Hwang, Approximating reachable sets for neural network-based models in real time via optimal control, IEEE Trans. Control Syst. Technol. (2023) 1–8, http://dx.doi.org/10.1109/TCST.2023.3234248.
[24] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, L. Daniel, Efficient neural network robustness certification with general activation functions, Adv. Neural Inf. Process. Syst. 31 (2018).
[25] H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L.V. Nguyen, W. Xiang, S. Bak, 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.
[26] M. Fazlyab, M. Morari, G.J. Pappas, Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming, IEEE Trans. Automat. Control (2020).
[27] Y. Wan, W. Zhou, J. Fan, Z. Wang, J. Li, X. Chen, C. Huang, W. Li, Q. Zhu, POLAR-express: Efficient and precise formal reachability analysis of neural-network controlled systems, 2023, arXiv preprint arXiv:2304.01218.
[28] M. Alajanbi, D. Malerba, H. Liu, Distributed reduced convolution neural networks, Mesop. J. Big Data 2021 (2021) 26–29.
[29] V. Dwivedi, N. Parashar, B. Srinivasan, Distributed learning machines for solving forward and inverse problems in partial differential equations, Neurocomputing 420 (2021) 299–316.
[30] B. Liu, Z. Ding, A distributed deep reinforcement learning method for traffic light control, Neurocomputing 490 (2022) 390–399.
[31] Y. Kwak, W.J. Yun, J.P. Kim, H. Cho, J. Park, M. Choi, S. Jung, J. Kim, Quantum distributed deep learning architectures: Models, discussions, and applications, ICT Exp. (2022).
[32] Z. Gao, F. Gama, A. Ribeiro, Wide and deep graph neural network with distributed online learning, IEEE Trans. Signal Process. 70 (2022) 3862–3877.
[33] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H.W. Chung, C. Sutton, S. Gehrmann, et al., Palm: Scaling language modeling with pathways, 2022, arXiv preprint arXiv:2204.02311.
[34] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1) (1991) 79–87.
[35] J. Zhang, Y. Li, W. Xiao, Z. Zhang, Non-iterative and fast deep learning: Multilayer extreme learning machines, J. Franklin Inst. B 357 (13) (2020) 8925–8955.
[36] G.A. Kale, C. Karakuzu, Multilayer extreme learning machines and their modeling performance on dynamical systems, Appl. Soft Comput. 122 (2022) 108861.
[37] Y. Wang, K.-Y. Lin, S. Cheng, L. Li, Variational quantum extreme learning machine, Neurocomputing 512 (2022) 83–99, URL https://www.sciencedirect.com/science/article/pii/S0925231222011225.
[38] J. Zhang, W. Xiao, Y. Li, S. Zhang, Residual compensation extreme learning machine for regression, Neurocomputing 311 (2018) 126–136.
[39] P. Mitra, C. Murthy, S.K. Pal, Unsupervised feature selection using feature similarity, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 301–312.
[40] M. Wang, C. Qi, H. Yan, H. Shi, Hybrid neural network predictor for distributed parameter system based on nonlinear dimension reduction, Neurocomputing 171 (2016) 1591–1597.
[41] W. Xiang, Data-driven modeling of switched dynamical systems via extreme learning machine, in: 2021 American Control Conference (ACC), IEEE, 2021, pp. 852–857.
[42] M. Khansari, E. Klingbeil, O. Khatib, Adaptive human-inspired compliant contact primitives to perform surface–surface contact under uncertainty, Int. J. Robot. Res. 35 (13) (2016) 1651–1675.
[43] S. Skelboe, Computation of rational interval functions, BIT Numer. Math. 14 (1) (1974) 87–95.
[44] W. Xiang, D.M. Lopez, P. Musau, T.T. Johnson, Reachable set estimation and verification for neural network models of nonlinear dynamic systems, in: Safe, Autonomous and Intelligent Vehicles, Springer, 2019, pp. 123–144.
[45] X. Ying, An overview of overfitting and its solutions, in: Journal of Physics Conference Series, Vol. 1168, 2019, 022022.
[46] S.H. Strogatz, Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering, CRC Press, 2018.
[47] R.F. Reinhart, Z. Shareef, J.J. Steil, Hybrid analytical and data-driven modeling for feed-forward robot control, Sensors 17 (2) (2017) 311.
[48] R.F. Reinhart, J.J. Steil, Neural learning and dynamical selection of redundant solutions for inverse kinematic control, in: 2011 11th IEEE-RAS International Conference on Humanoid Robots, IEEE, 2011, pp. 564–569.
[49] S.M. Khansari-Zadeh, A. Billard, Learning stable nonlinear dynamical systems with gaussian mixture models, IEEE Trans. Robot. 27 (5) (2011) 943–957.

Tao Wang is currently a Professor of Electrical Engineering at Southwest Jiaotong University, Chengdu, Sichuan, China. He received Ph.D. degree in Traffic Information Engineering and Control from Southwest Jiaotong University in 2007. His research interests include: electric traction control system; computer control system; control theory and applications.

Yejiang Yang is a Ph.D. candidate in Control Science and Engineering in the School Electrical Engineering at Southwest Jiaotong University. He is also a visiting scholar at the School of Computer and Cyber Sciences, Augusta University. His research interests are in system dynamics learning, hybrid systems.



Weiming Xiang received his B.S. degree in Automation from East China Jiaotong University, China, in 2005, M.S. degree in System Engineering from Nanjing University of Science and Technology, China, in 2007, and Ph.D. degree in Transportation Management and Planning from Southwest Jiaotong University, China, in 2014. He is currently an Associate Professor in the School of Computer and Cyber Sciences at Augusta University, USA. Dr. Xiang's current research centers on formal methods on safety, security and reliability of learning-enabled cyber–physical systems. He is the recipient of National Science Foundation CAREER Award, 2022. Dr. Xiang is a Senior Member of IEEE.