ASME Letters in Dynamic Systems and Control
Online journal at:
https://asmedigitalcollection.asme.org/lettersdynsys

Yejiang Yang
School of Electrical Engineering,
Southwest Jiaotong University,
111 North Section 1,
Second Ring Road,
Chengdu, Sichuan 621000, China
e-mail: yangyejiang@my.swjtu.edu.cn

Zihao Mo
School of Computer and Cyber Sciences,
Augusta University,
1120 15th Street,
Augusta, GA 30912
e-mail: zmo@augusta.edu

Weiming Xiang[2]
School of Computer and Cyber Sciences,
Augusta University,
1120 15th Street,
Augusta, GA 30912
e-mail: wxiang@augusta.edu

Efficient Neural Hybrid System Learning and Interpretable Transition System Abstraction for Dynamical Systems[1]

This article proposes a neural network hybrid modeling framework for dynamics learning to promote an interpretable, computationally efficient method of dynamics learning and system identification. First, a low-level model is trained to learn the system dynamics, which utilizes multiple simple neural networks to approximate the local dynamics generated from data-driven partitions. Then, based on the low-level model, a high-level model is trained to abstract the low-level neural hybrid system model into a transition system that allows computational tree logic (CTL) verification to promote model's ability to handle human interaction and verification efficiency. [DOI: 10.1115/1.4066516]

Keywords: hybrid and distributed system modeling, neural networks, nonlinear system modeling, maximum-entropy partitioning, model abstraction, complex systems, modeling, nonlinear system

1 Introduction

In recent years, the development of neural networks has received particular attention in various fields, including natural language processing [1], computer vision [2], etc. The applications of neural networks in system identification hold significant promise for they provide a precise approximation of the dynamics while requiring no prior knowledge of the system's mechanism. Neural networks serve as a predominant approach in machine learning, renowned for their exceptional ability to model complex phenomena with limited prior knowledge. Their proficiency in capturing intricate patterns in data offers valuable insights for dynamical system modeling, verification, and control.

However, neural networks are opaque, limiting our ability to validate them solely from an input–output perspective. This opacity also renders neural network models vulnerable to perturbations [3,4]. When it comes to applications in safety-critical scenarios, it requires time-consuming reachability analysis of the specific trajectories for verification, which poses challenges to real-time applications. According to Ref. [5], the computational efficiency is highly related to the scale of the neural network model. To promote computational efficiency, Ref. [6] proposed a computationally efficient distributive learning structure for dynamics learning, Ref. [7] proposed a fast reachable set estimation method with a compromise in conservatism, and Ref. [8] proposed an efficient bound propagation. However, when it comes to reachability analysis, specific time-consuming trajectory generations are necessary when applying these methods for the lack of model interpretability.

This article aims to promote neural networks" interpretability and computational efficiency in dynamical system modeling by introducing a novel dual-level modeling framework. Specifically, our proposed approach will divide dynamical system modeling into two essential levels: the low-level neural hybrid system model and its high-level transition system abstraction. The low-level model is employed to precisely capture the system's local behavior and enhance the computational efficiency with a parallel set of shallow neural networks to approximate the local dynamics. Then the high-level transition model, an abstraction based on neural hybrid systems, can be obtained based on reachability analysis designed to capture relationships and transition patterns among system subspaces.

Compared with the conventional neural network modeling method, the advantages of our proposed dual-level modeling framework can be summarized as follows:

- A novel data-driven partitioning method: maximum entropy (ME) partitioning is applied to partition the system state space into multiple local subspaces, which allows analysis of the dynamics within local subspaces.
- A computationally efficient low-level model: A concept of neural hybrid systems is proposed for distributed training and verification of a set of shallow neural networks, thereby enhancing computational efficiency.
- Promoting the learning model's interpretability: A novel transition system abstraction method is proposed to investigate the transition relationships between local partitions, further enhancing model interpretability.

[1]Paper presented at the 2024 Modeling, Estimation, and Control Conference (MECC 2024), Chicago, IL, Oct. 28–30, 2024. Paper No. MECC2024-50.
[2]Corresponding author.
Manuscript received July 20, 2024; final manuscript received September 4, 2024; published online October 3, 2024. Assoc. Editor: Murat Inalpolat.

This article is organized as follows: Preliminaries and problem formulations are given in Sec. 2. The main result, the dual-level modeling framework, is given in Sec. 3. In Sec. 4, modeling of the LASA datasets is given to illustrate the effectiveness of our proposed framework.[3] Conclusions are given in Sec. 5.

2 Preliminaries and Problem Formulation

In this article, the modeling problems for the discrete-time system are discussed, i.e., we aim to model the system in the form of

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

where $x \in \mathbb{R}^{n_x}$ is the system state, $u \in \mathbb{R}^{n_u}$ is the external input, and $f : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ is the ideal mapping that precisely describes the system patterns. Due to dimensions and nonlinearity, obtaining f could be challenging, and therefore, we aim to approximate f with neural network $\Phi : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ in

$$x(k+1) = \Phi(x(k), u(k)) \tag{2}$$

In the training of Φ, approximating f means adjusting the weight and bias of Φ to minimize the error between its output and the given dataset. In this article, the given dataset consisting of input–output pairs is in the form of

$$\mathcal{D} = \{(z^{(i)}, y^{(i)}) \mid z^{(i)} \in \mathbb{R}^{n_x+n_u}, y^{(i)} \in \mathbb{R}^{n_x}\} \tag{3}$$

However, neural networks face challenges as they typically require extensive data for training and often lack an intuitive understanding of the system's behavior. To unveil the black-box model usually requires a reachability analysis of the neural network dynamical system.

2.1 Reachability Analysis for Neural Network Dynamical System.

The reachability analysis of neural networks is useful in the neural network dynamical system verification for it can determine the range of outputs based on the interplay between the input sets and the structure of the neural network, and according to Refs. [8–10], simple neural network structure, i.e., Φ, that contains fewer layers and neurons has advantages in reachable computation.

Taking a L-layer feed-forward neural network $\Phi : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ as an example, its inter-layer propagation can be denoted as follows:

$$x_{i,k+1} = \sigma\left(\sum_j w_{ij,k+1} x_{j,k} + b_{i,k+1}\right) \tag{4}$$

where $x_{i,k+1}$ is the ith neuron output from the $k+1$th layer computed by applying the activation function σ to the weighted sum of the activations from the previous layer, plus a bias $b_{i,k+1}$ and $w_{ij,k+1}$ are the ith line and the jth row values of the weight bias $W_k \in \mathbb{R}^{n_k \times n_{k+1}}$, respectively.

Reachability analysis of neural networks goes through the inter-propagation of the neural network in Eq. (4), namely, for neural network model in Eq. (2) output reachable set computation when given the kth time-step state input set $\mathcal{X}_{(k)} \subset \mathbb{R}^{n_x}$ and external input set $\mathcal{U} \subset \mathbb{R}^{n_u}$ can be denoted as follows:

$$\mathcal{X}_{(k+1)} = \Phi^*(\mathcal{X}_{(k)}, \mathcal{U}) \tag{5}$$

where $\mathcal{X}_{(k+1)}$ is the reachable set output of Φ at the $k+1$th time-step computed by reachable set computation method Φ^* such as those presented in Refs. [7,11,12]. Reachable sets in given Kth time-steps require propagation of Eq. (5):

$$\mathcal{R}_{(K)} = \bigcup_{k=0}^{K} \mathcal{X}_{(k)} \tag{6}$$

where $\mathcal{R}_{(K)}$ is the reachable sets in K time-steps.

Due to the opacity of neural networks, the verification of Eq. (2) usually necessitates reachability computations of different trajectories to verify specific properties and can be heavily influenced by the neural network structure, posing a computational burden that challenges its application.

2.2 Maximum Entropy Partitioning.

ME partitioning proposed in Ref. [13] utilizes the Shannon entropy to partition the state space according to the data, which can be very useful in obtaining subspaces for distributed learning and prediction in neural networks.

Given a set of $N \in \mathbb{N}$ subspaces $\mathcal{P} = \{\mathcal{P}_i\}_{i=1}^N$, where $\mathcal{P}_i \subset \mathbb{R}^{n_x}$, the Shannon entropy of $\mathcal{P}$ can be denoted by

$$H(\mathcal{P}) = -\sum_{i=1}^{N} p(\mathcal{P}_i) \log p(\mathcal{P}_i) \tag{7}$$

where $p(\mathcal{P})$ denotes the probability of $\mathcal{P}_i$ occurrence in $\mathcal{P}_i$. In this data-driven process, $p(\mathcal{P}_i)$ is extrapolated by the sample set in the form of

$$p(\mathcal{P}_i) = \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \tag{8}$$

where $|\mathcal{D}|$ is the number of samples of $\mathcal{D}$, and D_i is defined by

$$\mathcal{D}_i = \{(z^{(j)}, y^{(j)}) \in \mathcal{D} \mid x \in \mathcal{P}_i, \forall [x^\top, u^\top]^\top = z^{(i)}\} \tag{9}$$

The ME partitioning employs the variation in Shannon entropy from system partitions to ascertain if the current set of partitions maximized the system's entropy after a bisecting method. Explicitly, the variation of Shannon entropy is in the form of

$$\Delta H = H(\hat{\mathcal{P}}) - H(\mathcal{P}) \tag{10}$$

where $\hat{\mathcal{P}}$ is the postbisecting set of partitions.

By setting a threshold $\varepsilon \geq 0$ as a stop condition, namely, the bisection process stops if $\Delta H < \varepsilon$, a proper set of partitions can be obtained.

2.3 Problem Formulation.

This article aims to promote the efficiency of learning and prediction of the neural network dynamical system in solving the following problem.

Problem 1. Given the dataset $\mathcal{D}$ in the form of Eq. (3), how do we model the dynamical system distributively with multiple simple neural networks?

To promote the interpretability of the learning model, the following problem will be the main concern after a distributed neural network model is obtained.

Problem 2. Given a neural network-based approximation Φ of f, how do we abstract Φ into an interpretable model that avoids real-time reachable set computation in Eq. (5)?

Solving Problem 1 allows parallel training and verification of multiple simple neural networks, which enhances the efficiency of neural network modeling while providing an accurate low-level model. On the basis of the low-level model, we can enhance the interpretability by abstracting the low-level model into a high-level model by solving Problem 2.

3 Dual-Level Modeling Framework

Before presenting the dual-level modeling framework, we make the assumption that the system training set (3) provides adequate information in the working zone for dynamical learning as follows:

[3]The developed modeling tool and code for experiments are publicly available online at: https://github.com/aicpslab/Dual-Level-Dynamic-System-Modeling

Assumption 1. *The working zone of ideal system dynamical description f in Eq. (1) is within the localized state space $x \in \mathcal{X}$, given the external input bound where $u \in [\underline{u}, \overline{u}]$.*

In most cases of neural network dynamical system modeling, Φ in Eq. (2) has a high accuracy in approximating the dynamics based on a sample set $\mathcal{D}$. On the basis of $\mathcal{D}$, we assume that the learning model applies only to a working zone with Assumption 1.

3.1 Neural Hybrid System Model and Transition System Abstraction. To solve Problem 1, we proposed the neural hybrid system model, which allows precise learning of the dynamical system through multiple small-scale neural networks. The neural hybrid system model is defined as follows:

Definition 1. *A neural hybrid system model is a tuple $\mathcal{H} = \langle \mathcal{P}, \Omega, \delta, \tilde{\Phi} \rangle$, where*

- $\Omega \subset \mathbb{R}^d$: *Working zone, with states $x(k) \in \Omega$.*
- $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{N_p}\}$: *Finite set of nonoverlapping partitions in the working zone, where: (1) $\mathcal{P}_i \subseteq \Omega$; (2) $\bigcup_{i=1}^{N} \mathcal{P}_i = \Omega$; (3) $\mathcal{P}_i \cap \mathcal{P}_j = \varnothing, \ \forall i \neq j$.*
- $\delta : \Omega \to \{1, 2, \ldots, N_p\}$: *Function mapping states to partitions $\delta(x(k)) = i$, implies $x(k) \in \mathcal{P}_i$.*
- $\tilde{\Phi} = \{\Phi_1, \Phi_2, \ldots, \Phi_{N_p}\}$: *Set of neural networks, each Φ_i models dynamics in $\mathcal{P}_i$.*

Definition 1 introduces a distributed structure of the neural networks that allow local approximations of the subspaces of state space called partitions. The dynamics of low-level model $\mathcal{H}$ is denoted as follows:

$$x(k+1) = \Phi_{\delta(x(k))}(x(k), u(k)) \tag{11}$$

The trajectory generation process in Eq. (11) is given in Fig. 1. This distributed structure aids in reducing the scale of the neural network approximation, thereby enhancing computational efficiency in both training and verification.

Compared with the conventional model, the neural hybrid system modeling has the advantages of real-time computation and verification. However, to gain insights from the neural hybrid system modeling and enhance interactivity between the learning model and human users, we can abstract the neural hybrid system into a transition system defined as follows:

Definition 2. *A transition system abstraction is a tuple $\mathcal{T} \triangleq \langle \Omega, \mathcal{Q}, \mathcal{E} \rangle$, where its elements are given as follows:*

- $\Omega \subset \mathbb{R}^{n_x}$: *Working zone, where this abstraction is applying to.*
- $\mathcal{Q} = \{\mathcal{Q}_1, \ldots, \mathcal{Q}_{N_q}\}$: *The finite set of subspaces called cells, where (1) $\mathcal{Q}_i \subseteq \Omega$; (2) $\Omega = \bigcup_{i=1}^{N_q} \mathcal{Q}_i$; and (3) $\mathcal{Q}_i \bigcap \mathcal{Q}_j = \varnothing$. With an index function $idx : \mathcal{Q} \to \mathbb{N}^{\leq N_q}$ for $idx(\mathcal{Q}_i) = i$.*
- $R : \mathbb{N}^{\leq N_q} \times \mathbb{N}^{\leq N_q} \to \mathbb{B}$: *Transition rules, if there exist a probable transition from $\mathcal{Q}_i$ to $\mathcal{Q}_j$, then $R(i, j) = 1$, else $R(i, j) = 0$.*

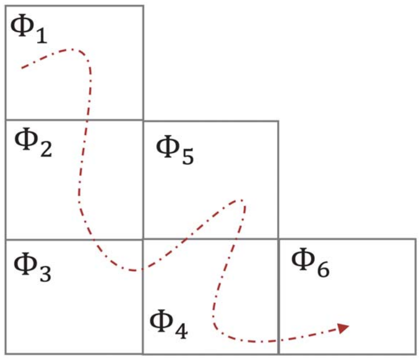

Fig. 1 Distributive low-level model $\mathcal{H}$, where $|\Phi| = 6$, and the dynamics within each partition $\mathcal{P}_i$ is iterated with subneural network $\tilde{\Phi}_i$

Illustrated by Fig. 2, a transition system abstraction unveils the interconnection of subspaces with transition rules T through abstracting the neural hybrid system $\mathcal{H}$. In this process, the data in the form of traces are generated by the neural hybrid system $\mathcal{H}$ by giving it randomized initial states, and randomized or user-specified external input for the nonautonomous dynamical systems.

3.2 Efficient Dynamics Learning via Low-Level Modeling. In this article, efficient dynamics learning is achieved through our proposed low-level model, termed neural hybrid system modeling. To begin with, ME partitioning proposed in Ref. [13] is applied to bisecting the working zone Ω based on the dataset $\mathcal{D}$. In this process, Ω and $\mathcal{P}$ are in the form of the interval, e.g., $\mathcal{P}_i = [\underline{p}_{i,1}, \overline{p}_{i,1}] \times [\underline{p}_{i,2} \times \overline{p}_{i,2}] \ldots$ in which $\overline{\mathcal{P}}_i = \{\overline{p}_{i,1}, \overline{p}_{i,2}, \ldots, \overline{p}_{i,n_x}\} \in \mathbb{R}^{n_x}$, etc. Specifically, we locates the jth dimension of the ith partition to bisect via

$$(i, j) = \arg\max_{i,j} D_{i,j} \tag{12}$$

where

$$D_{i,j} = \overline{p}_{i,j} - \underline{p}_{i,j} \tag{13}$$

We can keep bisecting the $\mathcal{P}$ until $\Delta H \leq \varepsilon$. After the ME partitioning, the set of partitions $\mathcal{P}$ with N_p partitions can be obtained, which subsequently defines the segmented dataset $\{\mathcal{D}_1, \ldots, \mathcal{D}_{N_p}\}$. With the segmented dataset, we can train the set of neural networks once given a neural network structure, namely, the layers, neurons, activation function, etc., of neural networks.

To further optimize the ME partitioning and simplify the learning model, we merge the redundant partitions based on the training performance of the neural network. Merging redundant partitions is based on the mean-square error (MSE) performance of the neural network. Given $\mathcal{D}$ and a trained neural network Φ, the MSE performance of Φ is expressed as follows:

$$\text{MSE}(\Phi, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left\| \Phi(z^{(i)}) - y^{(i)} \right\| \tag{14}$$

By setting a threshold based on MSE performance $\gamma \geq 0$, we are able to identify the redundant partitions that are considered to have similar performance under the same neural network structure, namely, if

$$\text{MSE}(\Phi, \mathcal{D}_i \cup \mathcal{D}_j) \leq \gamma \tag{15}$$

for a trained Φ, the corresponding partitions $\mathcal{P}_i$ and $\mathcal{P}_j$ are considered redundant partitions, and hence, they will be merged.

Merging the redundant partitions subsequently defines the switching logic δ and the set of neural networks $\tilde{\Phi}$ for the neural hybrid system $\mathcal{H}$. The low-level neural hybrid system modeling can be summarized in pseudo-code given in Algorithm 1.

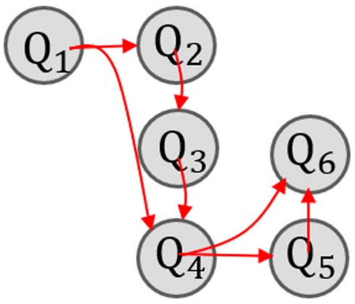

Fig. 2 High-level transition system abstraction $\mathcal{T}$ based on the low-level model $\mathcal{H}$, where the set of cells $|\mathcal{Q}| = 6$, and the transition rules R are denoted by arrows

Algorithm 1 Low-level neural hybrid system modeling

```
                                                        ▷ Maximum entropy partitioning
1: procedure ME partitioning $\Omega$, $\mathcal{D}$, $\varepsilon$
   Input: $\Omega$; $\mathcal{D}$; $\varepsilon$.
   Output: $\mathcal{P}$; $\cup\{\mathcal{D}_i\}$.
2:     $P_{save} \leftarrow \varnothing$; $\mathcal{D}_{save} \leftarrow \varnothing$;
3:     $P_1 \leftarrow \Omega$;
4:     while $\exists \Delta H_i \geq \varepsilon, \forall \mathcal{P}_i$ do
5:         $[i, j, Distance] \leftarrow \max(D_{i,j})$
6:         Obtain $\mathcal{P}_{temp1}$ and $\mathcal{P}_{temp2}$ under (12)
7:         Obtain $\mathcal{D}_{temp1}$ and $\mathcal{D}_{temp2}$
8:         if $\Delta H_i \geq entropy$ then                            ▷ Using (7)
9:             $P_i \leftarrow \{P_{temp1}, P_{temp2}\}$
10:            $\mathcal{D}_i \leftarrow \{\mathcal{D}_{temp1}, \mathcal{D}_{temp2}\}$
11:        else
12:            Add $P_i$ to $P_{save}$ and delete $P_i$
13:            Add $\mathcal{D}_i$ to $\mathcal{D}_{save}$ and delete $\mathcal{D}_i$
14:        end if
15:    end while
16:    return $\mathcal{P} \cup \mathcal{P}_{save}$; $\mathcal{D} \cup \mathcal{D}_{save}$
17: end procedure
                                                        ▷ Merging and dynamics learning
18: procedure Merge and Learn $\mathcal{P}$, $\cup\{\mathcal{D}_i\}$, $\Phi$
    Input: $\mathcal{P}$, $\cup\{\mathcal{D}_i\}$, $\Phi$
    Output: $\mathcal{P}$, $\Phi$
19:    $\ell \leftarrow |\mathcal{P}|$, $N \leftarrow 1$;                       ▷ Segmented partitions merge
20:    while $N < \ell$ do
21:        $n \leftarrow 1$;
22:        while $n \leq \ell$ do
23:            $n \leftarrow n + 1$;
24:            $\Phi_{N,n} \leftarrow \Phi$, $\mathcal{D}_{N,n} \leftarrow \mathcal{D}_N \cup \mathcal{D}_n$
25:            $\Phi_{N,n} \leftarrow \arg\min_{\Phi_{N,n}} MSE(\Phi_{N,n}, \mathcal{D}_{N,n})$
26:            if $MSE(\Phi_{N,n}, \mathcal{D}_{N,n}) \leq \gamma$ then
27:                $\mathcal{P}_N \leftarrow \{\mathcal{P}_N \cup \mathcal{P}_n\}$
28:                Delete $\mathcal{P}_n$
29:                $\ell \leftarrow \ell - 1$
30:            end if
31:        end while
32:        $N \leftarrow N + 1$;
33:    end while
34:                                           ▷ Generate neural network approximations
35:    $i \leftarrow 1$;
36:    while $i \leq N$ do
37:        $\Phi_i \leftarrow \arg\min_{\Phi_i} MSE(\Phi_i, \mathcal{D}_i)$
38:    end while
39: return $\mathcal{P} = \{\mathcal{P}_1 \ldots, \mathcal{P}_N\}$; $\Phi = \{\Phi_1, \ldots, \Phi_N\}$
40: end procedure
```

The low-level neural hybrid system can model the dynamical system through a distributive and computationally efficient framework, which makes it possible for parallel training in the merge and learning procedure, and distributive verification in Ref. [6]. To further exploit this distributive structure and promote the interpretability of the low-level learning model, we proposed a transition system abstraction method as the high-level model.

3.3 Interpretable Abstraction via High-Level Model. In high-level model abstraction, we intend to abstract the neural hybrid system model in Definition 1 into a transition system in Definition 2 with the help of the data generated by $\mathcal{H}$ called the set of samples, defined by

Definition 3. Set of samples $\mathcal{W} = \{w_1, w_2, \ldots, w_L\}$ of neural hybrid system (11) is a collection of sampled L traces obtained by given $\mathcal{H}$ different initial condition and randomized external input $u \in \mathcal{U}$, where for each trace w_i, $i = 1, \ldots, L$, is a finite sequence of time-steps and data $(k_{0,i}, d_{0,i}), (k_{1,i}, d_{1,i}), \ldots, (k_{M_i,i}, d_{M_i,i})$, where

- $k_{0,i} \in (0, \infty)$ and $k_{\ell+1,i} = k_{\ell,i} + 1$, $\forall \ell \in \mathbb{N}^{\leq M_i}$, $\forall i \in \mathbb{N}^{\leq L}$.
- $z_{\ell,i} = [x_i^\top(k_{\ell,i}),\ u_i^\top(k_{\ell,i})]^\top \in \mathbb{R}^{n_x+n_u}$, $\forall \ell = 0, 1, \ldots, M_i$, $\forall i \in \mathbb{N}^{\leq L}$, where $x_i(k_{\ell,i})$, $u_i(k_{\ell,i})$ denote the state and input of the system at ℓth step for ith trace, respectively.

Remark 1. It should be noted that the abstraction of the neural hybrid system is specific, meaning that different transition system abstractions can be obtained based on different control strategies. This specificity aids system designers in implementing and validating control strategies tailored to specific partitions.

After obtaining the set of samples, the set of cells $\mathcal{Q}$ is obtained via the ME partitioning method as in procedure ME partitioning in Algorithm 1 based on $\mathcal{W}$. Then, the transition relationships between cells are computed via reachability analysis in

$$\mathcal{Q}_i' = \bigcup_{j=1}^{N_p} \Phi_j^*(\mathcal{Q}_i \cap \mathcal{P}_j, \mathcal{U}) \tag{16}$$

where Φ_j^* indicates a reachable set computation method using the subneural network. Intuitively, based on Definition 2, the transition rule $R(i, j)$ is

$$R(i,j) = \begin{cases} 1, & \mathcal{Q}_i' \cap \mathcal{Q}_j \neq \varnothing \\ 0, & \mathcal{Q}_i' \cap \mathcal{Q}_j = \varnothing \end{cases} \tag{17}$$

The process of transition computation can be summarized in pseudo-code given in Algorithm 1.

Algorithm 2 Transition computation via $\mathcal{H}$

```
   Input: $\mathcal{P}$, $\mathcal{Q}$, $\Phi$, $\mathcal{U}$
   Output: $R$
1: $N_p \leftarrow |\mathcal{P}|$, $N_q \leftarrow |\mathcal{Q}|$
2: $i \leftarrow 1$; $j \leftarrow 1$;
3: while $i \leq n$ do
4:     $\mathcal{Q}_i' \leftarrow \bigcup_{l=1}^{N_p} \Phi_l^*(\mathcal{Q}_i \cap P_l, \mathcal{U})$          ▷ Using (16)
5:     $j \leftarrow 1$
6:     while $j \leq n$ do
7:         if $\mathcal{Q}_i' \cap \mathcal{Q}_j \neq \varnothing$, then
8:             $R(i, j) \leftarrow 1$
9:         else
10:            $R(i, j) \leftarrow 0$
11:        end if
12:        $j \leftarrow j + 1$
13:    end while
14:    $i \leftarrow i + 1$
15: end while
16: return $R$
```

The proposed dual-level modeling framework can be summarized as follows:

- The localized working zone of Ω, i.e., $\mathcal{P}$ can be obtained based on an ME partitioning process, which is completely data driven and can be easily tuned by adjusting the threshold.
- Partitions can be further optimized based on the MSE performance of the trained neural network to simplify the low-level model.
- The low-level model has a distributive structure consisting of simple neural networks that allow parallel training and verification, which is computationally efficient.
- The low-level model can be further abstracted into a high-level transition system, and this process can be specifically designed and allow system designers to develop and test control strategies that are specifically tailored for each distinct localized cell.
- The transitions can be of-line computed by reachability analysis and can be transferred into a transition graph, which enhances the learning model's interpretability and enables the feasibility of verifications based on logical descriptions.

Table 1 Training time and MSE of the low-level model

Shape name	Training time (ms)	MSE (10^{-5})
Khamesh	0.7147	0.3466
LShape	0.7784	0.3745
MultiModels$_1$	0.5603	0.2858
MultiModels$_2$	0.6225	0.2892
...	...	...

Table 2 Training time and MSE for ELM model

Shape name	Training time (ms)	MSE (10^{-5})
Khamesh	14.8098	0.0923
LShape	38.4118	0.0842
MultiModels$_1$	20.8079	0.0551
MultiModels$_2$	26.6117	0.0753
...	...	...

4 Applications to Dynamical System Modeling

Regarding the modeling of complex dynamical systems like human behaviors, learning-based methodologies have garnered significant attention for their efficacy in Refs. [14,15]. However, while learning-based approaches offer advantages over mechanistic modeling, they present numerous challenges in practical applications. For instance, typical issues include:

- The limited availability of sample data may result in a deep neural network-based dynamical system model that is not adequately trained, thereby hindering its ability to capture the full spectrum of human behavioral complexities.
- The inherent nature of human demonstrations, characterized by sudden shifts, suggests that a trained neural network-based dynamical system model might exhibit discrepancies in its behavior, especially in localized regions of the operational space.
- The interpretability deficit in neural network-based dynamical system models poses a significant challenge in real-time applications for limited, and computationally intensive verification methods.

The aforementioned issues are exemplified in the LASA dataset [16] modeling, which encompasses a diverse range of handwriting motions demonstrated by human users across 30 distinct shapes. This article addresses these issues through our proposed dual-level dynamical system modeling framework. The dual-level modeling process can be summarized as follows:

- Extreme learning machines (ELMs) are employed, each comprising 20 ReLU-activated neurons. These ELMs feature a randomized input weight matrix and bias vector, forming the core structure of the model. To highlight the efficacy of our modeling approach, an ELM with a solitary hidden layer containing 200 ReLU-activated neurons is trained to serve as a single-neural network reference model.
- A threshold of $\varepsilon = 4 \times 10^{-2}$ is set for ME partitioning variation in Algorithm 1. This setting led to the generation of the set of partitions in ME partitioning for the low-level model of all

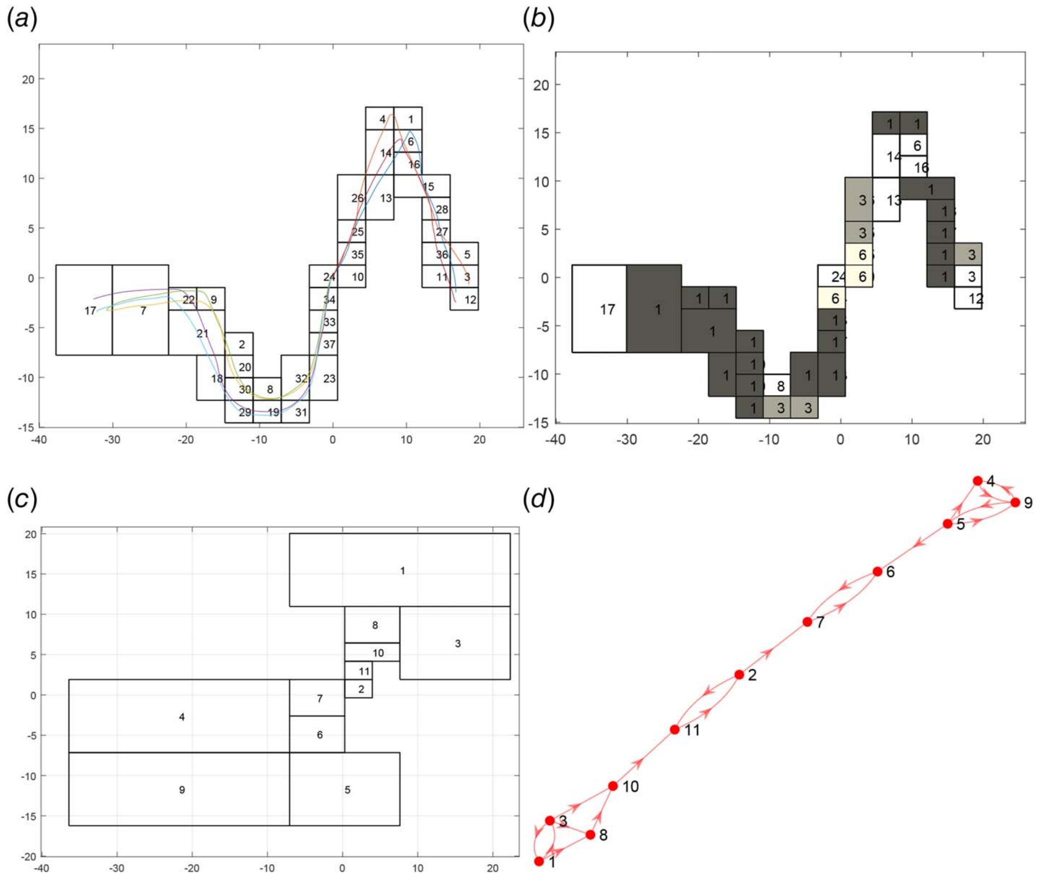

Fig. 3 Partitions, cells, and transition map abstraction of dual-level models for *MultiModels*$_2$ from LASA dataset: (*a*) *MultiModels*$_2$ handwriting human demonstration and 37 partitions obtained, (*b*) 12 partitions obtained where redundant ones are given in the same color, (c) 11 cells abstraction for $\mathcal{H}$ of *MultiModels*$_2$, and (*d*) transition map based on $\mathcal{T}$ of *MultiModels*$_2$

Table 3 Verification results of CTL formula: $\mathcal{T}_{\text{MultiModels}_2}$ with $\mathcal{Q}_9$ as the initial cell

CTL formula	$\mathcal{T}_{\text{MultiModels}_2}$
$\phi_1 = \exists \diamond \mathcal{Q}_2$	*True*
$\phi_2 = \forall \bigcirc \mathcal{Q}_4$	*False*
$\phi_3 = \exists(\mathcal{Q}_6) \wedge (\exists \bigcirc \mathcal{Q}_7)$	*True*

30 shapes, some results are given in Tables 1 and 2, and an illustration of *MutiModels*$_2$[4] is given in Fig. 3(*a*).

- By setting a threshold $\gamma = 1.5 \times 10^{-5}$ in merging, we simplify the low-level model by training fewer neural networks while maintaining accuracy.
- We obtain the abstraction data from randomly generated trajectories in the working zone Ω, where $\forall M_i = 400$, $\forall i \in \mathbb{N}^{\leq 400}$ under Definition 3. By applying the threshold $\varepsilon = 4 \times 10^{-2}$, a set of cells is then generated, as shown in Fig. 3(*c*).
- On the basis of the set of cells, we employ the Algorithm 2 to compute the transition relationships of the high-level model abstraction. The transition from Fig. 3(*c*) to Fig. 3(*d*) allows for the interpretation of transition relationships between local working zones.
- We verify the transition system abstraction via computation tree logic (CTL) formulae [17], where $\exists$ or $\forall$ denote the for *some* or *all* traces, and $\bigcirc$ denotes the next step. The formulae and results of *MultiModel*$_2$ are given in Table 3 as examples, ϕ_1 indicates the possibility of the neural hybrid system model being in $\mathcal{Q}_2$, ϕ_2 specifies that for every possible next step, the system is in $\mathcal{Q}_4$, and ϕ_3 signifies whether there exists a trajectory that reaches $\mathcal{Q}_7$ immediately after passing through $\mathcal{Q}_6$, given the initial condition is $\mathcal{Q}_9$.

4.1 Discussion. As observed in Tables 1 and 2, our proposed low-level modeling process can maintain the same level of the training error with smaller-scale neural networks, and the training time can be reduced due to parallel training of simple subneural networks. For example, in the modeling of *MutiModels*$_2$, our low-level model can be trained in only 0.6225 ms with multiple 20-ReLu-activated ELM, while the MSE is 0.2892×10^{-5}, where baseline conventional ELM model takes 26.6117 ms in training, with an MSE being 0.0753×10^{-5}.

The ME partitioning demonstrated in Figs. 3(*a*) and 3(*b*) shows that information-rich areas are allocated more partitions, indicating that this process can efficiently partition the state space based on Entropy.

Figure 3(*d*) and Table 3 show that our high-level transition system abstraction can unveil the transition between cells via the transition map. On the basis of high-level transition system abstraction, we can verify the system's properties against CTL formulae, and the verification results can be found in the modeling processes.

5 Conclusion and Future Works

In this article, a dual-level dynamical system learning framework is proposed to promote computational efficiency and interpretability in system identification. This framework utilizes a data-driven ME partitioning process to bisect the working zone, which makes parallel training and local analysis possible. Merging the partitions based on the training performance is proposed to simplify the learning model. The low-level model can then learn the dynamics precisely while only consisting of simple neural networks. A high-level model is proposed to promote interpretability through reachability analysis. This high-level model provides valuable insights into the transition relationship within the working zone with the transition map and allows user-specified verification against CTL formulae.

While the proposed framework shows promise, several limitations to our approach must be acknowledged. First, the data-driven ME partitioning process may result in an excessive number of partitions in high-dimensional state spaces, which may lead to a transition map that is too complex to be described using LTL language. Second, the abstraction process relies on the data-driven partitioning results rather than being guided by verification considerations, which suggests that the resulting partitions often do not align with the user's ideal verification regions. Additionally, the transition system we generate is nondeterministic, meaning that transitions cannot be quantified probabilistically. This means we cannot perform quantitative analysis on the probability of meeting target conditions, nor can we design control strategies to maximize the likelihood of achieving these conditions.

To address these limitations and further enhance the framework, several avenues for future research are identified. One potential direction is to develop model reduction techniques to reduce the dimension of the state space. Another area of improvement is applying the counterexample-guided abstraction refinement approach such as proposed in Ref. [18] to promote the partitioning process in high-level transition system abstraction. Furthermore, probability estimation for each transition will make the high-level model applicable in control synthesis. Finally, extending the framework to include more comprehensive verification tools and exploring its application to real-world industrial systems could provide valuable insights and validate its practical utility.

Funding Data

- National Science Foundation CAREER (Award No. 2143351).
- National Science Foundation CNS (Award No. 2223035).
- National Science Foundation IIS (Award No. 2331938).

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data and information that support the findings of this article are freely available.[5]

Nomenclature

$\mathbb{B}$	=	the set of Boolean variables
$\mathbb{N}$	=	the set of natural number sets
$\mathbb{R}$	=	the set of real numbers
$\mathbb{N}^{\leq n}$	=	the set of natural numbers less than or equal to n
$\mathbb{R}^n$	=	the vector space of n-tuples of real numbers
$\underline{X}$	=	the lower bound of an interval X
$\overline{X}$	=	the upper bound of an interval X

References

[1] Wang, Y., Wang, W., Chen, Q., Huang, K., Nguyen, A., De, S., and Hussain, A., 2023, "Fusing External Knowledge Resources for Natural Language Understanding Techniques: A Survey," Inf. Fusion, **92**, pp. 190–204.

[2] Stefenon, S. F., Corso, M. P., Nied, A., Perez, F. L., Yow, K. -C., Gonzalez, G. V., and Leithardt, V. R. Q., 2022, "Classification of Insulators Using Neural

[4]Complete results include modeling for all 30 shapes can be found on our GitHub repository on dual-level dynamical system modeling at https://github.com/aicpslab/Dual-Level-Dynamic-System-Modeling/tree/main/Results

[5]See Note 4.

Network Based on Computer Vision," IET Generation, Transm. Distrib., **16**(6), pp. 1096–1107.

[3] Zhang, X., Zheng, X., and Mao, W., 2021, "Adversarial Perturbation Defense on Deep Neural Networks," ACM Comput. Surv. (CSUR), **54**(8), pp. 1–36.

[4] Yang, Y., Wang, T., Woolard, J. P., and Xiang, W., 2022, "Guaranteed Approximation Error Estimation of Neural Networks and Model Modification," Neural Netw., **151**, pp. 61–69.

[5] Brix, C., Müller, M. N., Bak, S., Johnson, T. T., and Liu, C., 2023, "First Three Years of the International Verification of Neural Networks Competition (VNN-COMP)," Int. J. Softw. Tools Technol. Transf., **25**(3), pp. 1–11.

[6] Wang, T., Yang, Y., and Xiang, W., 2023, "Computationally Efficient Neural Hybrid Automaton Framework for Learning Complex Dynamics," Neurocomputing, **562**, p. 126879.

[7] Xiang, W., Tran, H.-D., and Johnson, T. T., 2018, "Output Reachable Set Estimation and Verification for Multilayer Neural Networks," IEEE Trans. Neural Netw. Learn. Syst., **29**(11), pp. 5777–5783.

[8] Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z., 2021, "Beta-CROWN: Efficient Bound Propagation With Per-Neuron Split Constraints for Neural Network Robustness Verification," Adv. Neural Inf. Process. Syst., **34**, pp. 29909–29921.

[9] Tran, H.-D., Musau, P., Lopez, D. M., Yang, X., Nguyen, L. V., Xiang, W., and Johnson, T. T., 2019, "Parallelizable Reachability Analysis Algorithms for Feed-forward Neural Networks," IEEE/ACM 7th International Conference on Formal Methods in Software Engineering (FormaliSE), Montreal, QC, Canada, May 27, IEEE, pp. 51–60.

[10] Feng, R., Leung, C.-S., and Sum, J., 2018, "Robustness Analysis on Dual Neural Network-Based *k* WTA With Input Noise," IEEE Trans. Neural Netw. Learn. Syst., **29**(4), pp. 1082–1094.

[11] Lopez, D. M., Choi, S. W., Tran, H. -D., and Johnson, T. T., 2023, "NNV 2.0: The Neural Network Verification Tool," International Conference on Computer Aided Verification (CAV), Paris, France, July 17–22, Springer, pp. 397–412.

[12] Vincent, J. A., and Schwager, M., 2021, "Reachable Polyhedral Marching (RPM): A Safety Verification Algorithm for Robotic Systems With Deep Neural Network Components," IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, May 30–June 5, IEEE, pp. 9029–9035.

[13] Yang, Y., and Xiang, W., 2023, "Modeling Dynamical Systems With Neural Hybrid System Framework via Maximum Entropy Approach," 2023 American Control Conference (ACC), San Diego, CA, May 31–June 2, pp. 3907–3912.

[14] Reinhart, R. F., and Steil, J. J., 2011, "Neural Learning and Dynamical Selection of Redundant Solutions for Inverse Kinematic Control," 11th IEEE-RAS International Conference on Humanoid Robots, Bled, Slovenia, Oct. 26–28, IEEE, pp. 564–569.

[15] Kanazawa, A., Zhang, J. Y., Felsen, P., and Malik, J., 2019, "Learning 3D Human Dynamics From Video," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, June 15–20, pp. 5614–5623.

[16] Khansari-Zadeh, S. M., and Billard, A., 2011, "Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models," IEEE Trans. Rob., **27**(5), pp. 943–957.

[17] Pan, H., Li, Y., Cao, Y., and Ma, Z., 2016, "Model Checking Computation Tree Logic Over Finite Lattices," Theor. Comput. Sci., **612**, pp. 45–62.

[18] Hajdu, Á., and Micskei, Z., 2020, "Efficient Strategies for CEGAR-Based Model Checking," J. Autom. Reason., **64**(6), pp. 1051–1091.