Adaptive Control via Lyapunov-Based Deep Long Short-Term Memory Networks

Xuehui Shen, Emily J. Griffis, Wenyu Wu, and Warren E. Dixon, *Fellow, IEEE*

***Abstract*—Motivated by the memory capabilities of long short-term memory (LSTM) networks and the improved function approximation power of deep learning, this article develops a Lyapunov-based adaptive controller using a deep LSTM neural network (NN) architecture. The architecture is made deep by stacking the LSTM cells on top of each other, and therefore, the overall architecture is henceforth referred to as a stacked LSTM (SLSTM). Specifically, an adaptive SLSTM architecture is developed with shortcut connections and is implemented in the controller as a feedforward estimate. Analytical adaptive laws derived from a Lyapunov-based stability analysis update the SLSTM weights in real-time and allow the SLSTM estimate to approximate the unknown drift dynamics. A Lyapunov-based stability analysis ensures asymptotic tracking error convergence for the developed Lyapunov-based stacked LSTM (Lb-SLSTM) controller and weight adaptation law. The Lb-SLSTM adaptive controller yielded an average improvement of 22.24% and 70.01% in tracking error performance, as well as 40.16% and 81.32% in function approximation error performance when compared to the baseline Lb-LSTM and Lyapunov-based deep neural network (Lb-DNN) architectures, respectively. Furthermore, the Lb-SLSTM model yielded a 96.00% and 98.75% improvement in maximum steady state error performance when compared to the Lb-LSTM and Lb-DNN models, respectively.**

***Index Terms*—Adaptive control, long short-term memory networks (LSTMs), Lyapunov methods, neural networks, nonlinear control systems.**

I. Introduction

Deep neural network (DNN)-based control methods have become increasingly popular due to the improved function approximation performance over shallow neural networks (NNs) (cf., [1] and [2]). However, the NN architectures in these results are feedforward and static, and therefore do not have access to previous state information. Results in [3] augment a feedforward NN with external memory and show that the addition of external memory to the NN architecture improves learning and overall function approximation performance. Memory in NNs significantly enhances function approximation by enabling dynamic storage and retrieval of past information [4]. This mechanism allows the network to retain key patterns over time and adapt its outputs based on historical context, improving learning efficiency and generalization. Unlike traditional NNs, which rely solely on weights to encode dependencies, memory modules provide flexibility in handling complex, nonstationary, or temporally structured data. However, the NN architecture in [3] is feedforward and the memory is external to the NN.

Received 18 December 2024; accepted 23 March 2025. Date of publication 8 April 2025; date of current version 29 August 2025. This work was supported in part by AFRL under Grant FA8651-21- F-1027 and in part by AFOSR under Grant FA9550-19- 1-0169. Recommended by Associate Editor X. Chen. *(Corresponding author: Xuehui Shen.)*

The authors are with the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesvile, FL 32601 USA (e-mail: xuehuishen@ufl.edu; emilygriffis00@ufl.edu; wuwenyu@ufl.edu; wdixon@ufl.edu).

Digital Object Identifier 10.1109/TAC.2025.3558376

Recurrent NNs (RNNs) [such as long short-term memory (LSTM) networks] have an internal memory, which significantly enhances function approximation by enabling dynamic storage and retrieval of past information. Results, such as [5] and [6], develop LSTM-based methods using pretrained LSTM networks and show that LSTMs have better function approximation capabilities when compared to feedforward NNs and traditional RNNs. LSTMs can maintain an internal memory to store and retrieve information over long sequences, and the gate mechanism allows LSTMs to decide which information to keep, which to forget, and which to output, dynamically improving the flexibility to adjust the memory based on the sequence. This internal memory and gating mechanisms can capture the inherent relationships between inputs at different time steps. However, previous works, such as [7], [8], [9], [10], [11], [12], [13], [14], further establish that deeper architectures expedite learning and improve overall function approximation capabilities. Results, such as [15], [16], [17], [18], develop stacked RNN architectures, consisting of multiple RNN cells, where each RNN cell output is used as the input to the subsequent RNN cell. Also, results in [16], [17], and [18] implement stacked LSTM (SLSTM) architectures for applications, such as natural language processing and time-series forecasting, using a series of LSTM cells, and show significant empirical improvements in function approximation performance over a single LSTM cell. These developments motivate this work to develop an SLSTM architecture that can be implemented with stability guarantees.

Results, such as [5] and [6], leverage machine learning-based offline training techniques in an open-loop manner to develop LSTM-based controllers and show improved function approximation performance over other NN models. Offline training often requires large, sufficiently rich datasets and is not robust to real-time changes in the dynamic system. Moreover, since the LSTM is implemented in an open-loop manner, the overall control architecture lacks stability guarantees. Motivated by the desire for a stability guarantee and the advantage of real-time learning without the requirement for offline training from a dataset, a class of Lyapunov-based DNNs (Lb-DNNs) have recently been developed in results, such as [12], [14], and [19], where the adaptive update laws are analytically derived from the Lyapunov-based stability analysis. The result in [20] develops a Lyapunov-based LSTM (Lb-LSTM) and comparative simulations demonstrate improved performance over adaptive shallow and deep feedforward NNs. However, the developed adaptation law and control design in [20] only considers one LSTM cell, whereas in practice, LSTM cells are often implemented stacked in a series as established in results, such as [21]. Therefore, motivation exists to develop a Lyapunov-based controller with a stability-driven adaptation method for the SLSTM architecture.

Motivated by improved function approximation with SLSTM and real-time adaptive learning with stability guarantees, a Lyapunov-based adaptive controller is developed using an adaptive SLSTM NN architecture (Lb-SLSTM). A robustifying signum term is included in the control law to compensate for uncertain terms in the closed-loop error system and facilitate the stability analysis. To address the vanishing

gradient issue with the increasing depth of LSTM cells, the developed architecture is designed to allow for shortcut connections between the LSTM cells (similar to results, such as [22]). The complexity of a stacked structure combined with the dynamical nature of the LSTM cells poses mathematical challenges that complicate the derivation of stability-driven adaptation laws. To overcome the resulting mathematical challenges, we derive the Lb-SLSTM weight adaptation laws by incorporating the Jacobians of the individual LSTM cells with shortcut connections into the adaptation law development. The Lb-SLSTM is implemented in the controller as a feedforward term to adaptively estimate the unknown system dynamics. The Lb-SLSTM is updated in real-time using a Lyapunov-based, stability-driven adaptation law, which ensures asymptotic tracking error convergence. Comparative simulations are performed on an eight-state nonlinear system using the DNN model in [14] and the LSTM model in [20]. The Lb-SLSTM adaptive controller yielded an average improvement of 22.24% and 70.01% in tracking error performance, as well as 40.16% and 81.32% in function approximation error performance when compared to the baseline Lb-LSTM and Lb-DNN architectures, respectively. Furthermore, the Lb-SLSTM model yielded a 96.00% and 98.75% improvement in maximum steady state error performance when compared to the Lb-LSTM and Lb-DNN models, respectively.

Notations and Preliminaries: The $n \times n$-dimensional identity matrix is represented by $I_n \in \mathbb{R}^{n\times n}$. The right-to-left matrix product operator is represented by $\overset{\curvearrowleft}{\prod}$, i.e., $\overset{\curvearrowleft}{\prod_{l-1}^{m}} A_l = A_m \ldots A_2 A_1$ and $\overset{\curvearrowleft}{\prod_{l-p}^{m}} A_p = 1$ if $p > m$. The signum function is represented as sgn(). The Kronecker product and Hadamard product are denoted by $\otimes$ and $\odot$, respectively. The function composition operator is denoted as $\circ$, i.e., given functions $f(\cdot)$ and $g(\cdot)$, $f \circ g(x) = f(g(x))$. The vectorization operator is denoted by vec$(\cdot)$, i.e., given $A \triangleq [a_{i,j}] \in \mathbb{R}^{m\times n}$, vec$(A) \triangleq [a_{1,1}, \ldots, a_{m,1}, \ldots, a_{1,n}, \ldots, a_{m,n}]^\top$. The Hadamard product adheres to the following properties as [23, Fact 7.6.3] and [24, Thm. 3]. Given any $a, b \in \mathbb{R}^n$, $a \odot b = \text{diag}(a)b$ and therefore $\frac{\partial}{\partial b}(a \odot b) = \text{diag}(a)$ where diag$(a) \in \mathbb{R}^{n\times n}$ denotes a diagonal matrix with its main diagonal as vector a. Let the functions $a(c), b(c) \in \mathbb{R}^n$ be continuously differentiable with respect to $c \in \mathbb{R}^p$. Then, $\frac{\partial}{\partial c}(a \odot b) = \frac{\partial}{\partial c} a \odot b + a \odot \frac{\partial}{\partial c} b$. The vectorization operator meets the following properties [23, Prop. 7.1.9]. Given any $A \in \mathbb{R}^{n\times m}$, $B \in \mathbb{R}^{m\times p}$, and $C \in \mathbb{R}^{p\times r}$, vec$(ABC) = (C^\top \otimes A)vec(B)$, and $\frac{\partial}{\partial \text{vec}(B)}vec(ABC) = C^\top \otimes A$. The p-norm is denoted by $\|\cdot\|_p$, where the subscript is suppressed when $p = 2$. Almost all time is denoted as a.a.t.

II. System Dynamics and Control Objective

Consider the dynamical system modeled as

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

where $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the state, $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ denotes the control input, and $f : \mathbb{R}^n \to \mathbb{R}^n$ denotes an unknown differentiable drift vector field.

The control objective is to design an Lb-SLSTM adaptive controller that guarantees the state x tracks the desired state trajectory $x_d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$. As a means to quantify the tracking objective, the tracking error $e : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is defined as

$$e \triangleq x - x_d. \tag{2}$$

It is assumed that x_d is designed to be sufficiently smooth, i.e., $\|x_d(t)\| \leq \overline{x_d}$, $\|\dot{x}_d(t)\| \leq \overline{\dot{x}_d}$, and $\|\ddot{x}_d(t)\| \leq \overline{\ddot{x}_d} \forall t \in \mathbb{R}_{\geq 0}$, where $\overline{x_d}, \overline{\dot{x}_d}, \overline{\ddot{x}_d} \in \mathbb{R}_{>0}$ are known constants. Taking the time derivative of (2) and substituting the model dynamics in (1) yields

$$\dot{e} = f(x) + u - \dot{x}_d. \tag{3}$$

III. Control Development

A. LSTM NN Model

The LSTM NN can be represented in continuous-time[1] based on [20] and Euler's method [26] as

$$\begin{aligned} f^*(\eta, h, \theta) &= \sigma_g \circ \left(V_f^\top \eta + W_f^\top h\right) \\ i(\eta, h, \theta) &= \sigma_g \circ \left(V_i^\top \eta + W_i^\top h\right) \\ o(\eta, h, \theta) &= \sigma_g \circ \left(V_o^\top \eta + W_o^\top h\right) \\ c^*(\eta, h, \theta) &= \sigma_c \circ \left(V_c^\top \eta + W_c^\top h\right) \end{aligned} \tag{4}$$

where $\eta : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_1}$ denotes the LSTM input. The cell state and hidden state are denoted by $c : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_2}$ and $h : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_2}$, respectively, where $h(0) = c(0) = 0, l_1 \in \mathbb{R}_{>0}$ denotes the size of the LSTM input, and $l_2 \in \mathbb{R}_{>0}$ denotes the number of neurons. The forget gate, input gate, cell gate, and output gate are represented by $f^*(\eta, h, \theta) \in \mathbb{R}^{l_2}$, $i(\eta, h, \theta) \in \mathbb{R}^{l_2}$, $c^*(\eta, h, \theta) \in \mathbb{R}^{l_2}$, and $o(\eta, h, \theta) \in \mathbb{R}^{l_2}$, respectively. The sigmoid and tanh activation functions are represented by $\sigma_g : \mathbb{R}^{l_2} \to \mathbb{R}^{l_2}$ and $\sigma_c : \mathbb{R}^{l_2} \to \mathbb{R}^{l_2}$, respectively. The weight matrices are represented by $V_f^\top, V_c^\top, V_i^\top, V_o^\top \in \mathbb{R}^{l_2\times l_1}$, $V_h^\top \in \mathbb{R}^{l_1\times l_2}$, $W_f^\top, W_c^\top, W_i^\top, W_o^\top \in \mathbb{R}^{l_2\times l_2}$. The weights can be represented as a concatenated state vector $\theta \in \mathbb{R}^{l_3}$ defined as $\theta \triangleq [\text{vec}(V_c)^\top, \text{vec}(V_i)^\top, \text{vec}(V_f)^\top, \text{vec}(V_o)^\top, \text{vec}(V_h)^\top, \text{vec}(W_c)^\top, \text{vec}(W_i)^\top, \text{vec}(W_f)^\top, \text{vec}(W_o)^\top]^\top$, where $l_3 = 5l_1l_2 + 4l_2l_2$. The cell state and hidden state dynamics are defined as

$$\begin{aligned} \dot{c} &= -b_c c + b_c \Psi_c(\eta, c, h, \theta) \\ \dot{h} &= -b_h h + b_h \Psi_h(\eta, c, h, \theta) \end{aligned}$$

respectively, where $b_c, b_h \in \mathbb{R}_{>0}$ denote user-selected constants. The functions $\Psi_c(\eta, c, h, \theta) \in \mathbb{R}^{l_2}$, $\Psi_h(\eta, c, h, \theta) \in \mathbb{R}^{l_2}$ are defined as

$$\begin{aligned} \Psi_c(\eta, c, h, \theta) &= f^*(\eta, h, \theta) \odot c + i(\eta, h, \theta) \odot c^*(\eta, h, \theta) \\ \Psi_h(\eta, c, h, \theta) &= o(\eta, h, \theta) \odot (\sigma_c \circ \Psi_c(\eta, c, h, \theta)) \end{aligned}$$

respectively. To ensure the LSTM output is the appropriate dimension, a fully connected layer is added to the LSTM cell in (4). Thus, the output of the LSTM $\Psi(\eta, c, h, \theta) \in \mathbb{R}^{l_1}$ can be modeled as

$$\Psi(\eta, c, h, \theta) = V_h^\top \Psi_h(\eta, c, h, \theta) \tag{5}$$

where V_h denotes the output weight matrix and $l_1 = n$.

B. Lb-SLSTM Architecture

The SLSTM architecture has grown in popularity due to its ability to effectively leverage the improved function approximation performance of deep architectures and long short-term dependencies in time sequences. Stacking the LSTM cells and creating a deep architecture

[1]The LSTM cell in [25] is in discrete-time and is converted into a continuous-time model as shown in Fig. 1. This conversion of the SLSTM model in (1) to continuous-time was done to make the architecture better suited for control of continuous-time systems. Derived from the continuous-time model in (4), b_c and b_h can be fine-tuned to improve the performance of the continuous-time LSTM cell.

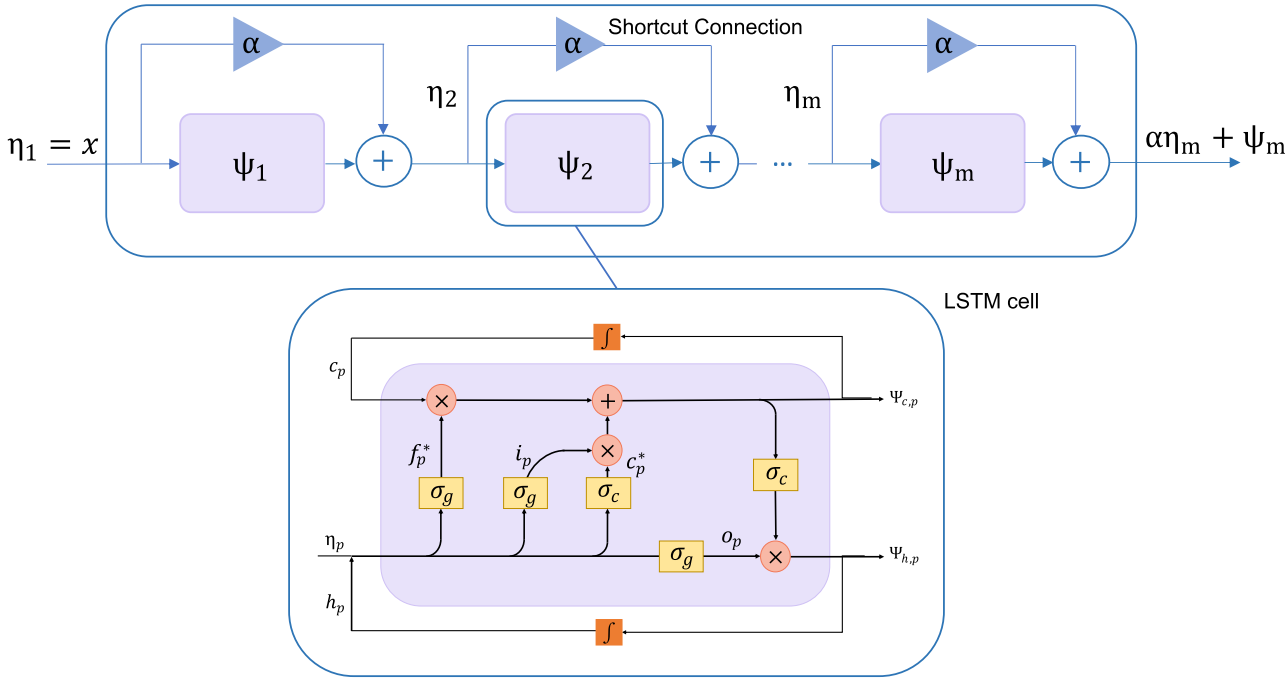

Fig. 1. SLSTM model $\Phi(\eta_1, C, H, \Theta)$ for the pth LSTM cell is denoted by $\Psi_p\ \forall p \in \{1, \ldots, m\}$, where the input of Ψ_p is denoted by η_p. Then, the output of the pth LSTM cell with a shortcut signal is represented by $\eta_p = \alpha\eta_{p-1} + \Psi_{p-1}$ for all $p \in \{2, \ldots, m\}$, and the output of the SLSTM model is $\alpha\eta_m + \Psi_m$. The symbol $\times$ in Fig. 1 denotes the Hadamard product.

yields faster learning and has been empirically shown to improve function approximation performance [2]. Therefore, the SLSTM in Fig. 1 is constructed by stacking LSTM cells such that the output of each LSTM cell works as the input to the subsequent LSTM cell. Since the SLSTM architecture is deep, a shortcut connection is implemented across each LSTM cell to address the challenges of vanishing gradients [22]. Let Ψ_p refer to the pth LSTM cell modeled in (5) defined as $\Psi_p \triangleq \Psi(\eta_p, c_p, h_p, \theta_p)\ \forall p \in \{1, \ldots, m\}$, where $\eta_p : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_1}$, $c_p : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_2}$, $h_p : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_2}$, and $\theta_p \in \mathbb{R}^{l_3}$ denote the input, cell state, hidden state, and weights vector of Ψ_p, respectively, and $m \in \mathbb{Z}_{>0}$ denotes the number of LSTM cells. The pth input η_p can be represented in a recursive relation as

$$\eta_p \triangleq \begin{cases} \alpha\eta_{p-1} + \Psi_{p-1}, & p \in \{2, \ldots, m\} \\ \eta_1, & p = 1 \end{cases} \tag{6}$$

where $\alpha \in \{0, 1\}$ is a user-selected boolean value that denotes the existence of shortcut connections across the LSTM cells. As shown in (6), when $\alpha = 0$, the SLSTM architecture has no shortcut connections, and when $\alpha = 1$, the SLSTM architecture has shortcut connections. Therefore, using (6), the SLSTM output $\Phi(\eta_1, C, H, \Theta) \in \mathbb{R}^{l_1}$ can be modeled as

$$\Phi\left(\eta_1, C, H, \Theta\right) \triangleq \alpha\eta_m + \Psi_m \tag{7}$$

where $\Theta \triangleq [\theta_1^\top \ \ldots \ \theta_m^\top]^\top \in \mathbb{R}^{ml_3}$ denotes the weights for the entire SLSTM architecture, and the concatenated vectors of the cell states and hidden states are defined as $C \triangleq [c_1^\top, \ldots, c_m^\top]^\top \in \mathbb{R}^{ml_2}$ and $H \triangleq [h_1^\top, \ldots, h_m^\top]^\top \in \mathbb{R}^{ml_2}$, respectively.

Based on the universal function approximation property [27], let $\mathcal{C}(\mathcal{X})$ represent the space of continuous functions over the set $\mathcal{X} \subseteq \mathbb{R}^{l_1}$, where $x \in \mathcal{X}$. The function space of SLSTM is dense in $\mathcal{C}(\mathcal{X})$. Therefore, for any given $\overline{\varepsilon} \in \mathbb{R}_{>0}$, there exist ideal weight matrices Θ such that $\|\varepsilon(x)\| \leq \overline{\varepsilon}\ \forall x \in \mathcal{X}$, where $\varepsilon : \mathbb{R}^{l_1} \to \mathbb{R}^{l_1}$ represents an unknown function reconstruction error. Then, the unknown drift dynamics $f(x)$ in (1) can be modeled by the SLSTM architecture in (7) as

$$f(x) = \Phi\left(x, C, H, \Theta\right) + \varepsilon(x) \tag{8}$$

where the input of the SLSTM model $\eta_1 \in \mathbb{R}^{l_1}$ is equal to the system state x and the dimension $l_1 = n$.

Assumption 1: There exists a known constant $\overline{\Theta} \in \mathbb{R}_{>0}$ such that the ideal weights Θ for (7) can be bounded as $\|\Theta\| \leq \overline{\Theta}$ [28, Assumption 1].[2]

C. Lb-SLSTM Weight Adaptation Law

Using the SLSTM model in (7), an adaptive SLSTM estimate is constructed in this section and is implemented in the controller as a feedforward estimate to approximate the unknown drift dynamics $f(x)$. Since the ideal weights are unknown, adaptive weight estimates are implemented and updated using a Lyapunov-based weight adaptation law.

Let $\widehat{\Psi}_p$ refer to the estimate of pth LSTM cell modeled in (5) defined as $\widehat{\Psi}_p \triangleq \Psi(\hat{\eta}_p, \hat{c}_p, \hat{h}_p, \hat{\theta}_p) \forall p \in \{1, \ldots, m\}$, where $\hat{\eta}_p : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_1}$, $\hat{c}_p : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_1}$, and $\hat{h}_p : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_1}$ denote the input of the LSTM estimate, cell state estimate, and hidden state estimate of $\widehat{\Psi}_p$, respectively. The weight estimates for the pth LSTM cell $\hat{\theta}_p : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_3}$ are defined as $\hat{\theta}_p \triangleq [\text{vec}(\widehat{V}_{p,c})^\top, \text{vec}(\widehat{V}_{p,i})^\top, \text{vec}(\widehat{V}_{p,f})^\top, \text{vec}(\widehat{V}_{p,o})^\top, \text{vec}(\widehat{V}_{p,h})^\top, \text{vec}(\widehat{W}_{p,c})^\top, \text{vec}(\widehat{W}_{p,i})^\top, \text{vec}(\widehat{W}_{p,f})^\top, \text{vec}(\widehat{W}_{p,o})^\top]^\top \forall p \in \{1, \ldots, m\}$. Furthermore, let the overall weight estimates for the SLSTM model $\widehat{\Theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{ml_3}$ be defined as $\widehat{\Theta} \triangleq [\hat{\theta}_1^\top \ \ldots \ \hat{\theta}_m^\top]^\top$, and let the shorthand notation for the SLSTM adaptive estimate of $f(x)$ be defined as $\widehat{\Phi} \triangleq \Phi(x, \widehat{C}, \widehat{H}, \widehat{\Theta}) \in \mathbb{R}^{l_1}$.

[2]The robust adaptive work in [29] could provide extensions for an unknown $\overline{\Theta}$.

The overall cell states and hidden states estimates of $\widehat{\Phi}$ are defined as $\widehat{C} \triangleq [\hat{c}_1^\top, \ldots, \hat{c}_m^\top]^\top \in \mathbb{R}^{ml_2}$ and $\widehat{H} \triangleq [\hat{h}_1^\top, \ldots, \hat{h}_m^\top]^\top \in \mathbb{R}^{ml_2}$, respectively. Therefore, the input estimate of each LSTM is designed as

$$\hat{\eta}_p \triangleq \begin{cases} \alpha\hat{\eta}_{p-1} + \widehat{\Psi}_{p-1}, & p \in \{2, \ldots, m\} \\ x, & p = 1. \end{cases} \tag{9}$$

Based on the subsequent stability analysis, the adaptation law for the weight estimates of the SLSTM in (7) is designed as

$$\dot{\widehat{\Theta}} \triangleq \Gamma \widehat{\Phi}'^\top e \tag{10}$$

where $\Gamma \in \mathbb{R}^{ml_3 \times ml_3}$ is a positive-definite adaptation gain matrix, and $\widehat{\Phi}' \in \mathbb{R}^{l_1 \times ml_3}$ is a shorthand notation denoting the Jacobian $\widehat{\Phi}' \triangleq \frac{\partial \widehat{\Phi}}{\partial \widehat{\Theta}}$. To facilitate the subsequent development, let the shorthand notation $\widehat{\Phi}'_p$ be defined as $\widehat{\Phi}'_p \triangleq \frac{\partial \widehat{\Phi}}{\partial \hat{\theta}_p} \forall p \in \{1, \ldots, m\}$. Then, the Jacobian $\widehat{\Phi}'$ can be expressed as $\widehat{\Phi}' \triangleq [\widehat{\Phi}'_1, \ldots, \widehat{\Phi}'_m]$. Using (12) and (15), the term $\widehat{\Phi}'_p$ can be expressed as

$$\widehat{\Phi}'_p = \left(\overset{\curvearrowleft}{\prod_{l=p+1}^{m}} (\alpha I_n + \Xi_l) \right) \Lambda_p \ \forall p \in \{1, \ldots, m\} \tag{11}$$

where $\Lambda_p \triangleq \frac{\partial \widehat{\Psi}_p}{\partial \hat{\theta}_p}$ and $\Xi_p \triangleq \frac{\partial \widehat{\Psi}_p}{\partial \hat{\eta}_p}$.[3]

Remark 1: The term Λ_p can be expressed as $\Lambda_p \triangleq [\Lambda_{p,\widehat{V}_{p,c}}, \Lambda_{p,\widehat{V}_{p,i}}, \Lambda_{p,\widehat{V}_{p,f}}, \Lambda_{p,\widehat{V}_{p,o}}, \Lambda_{p,\widehat{V}_{p,h}}, \Lambda_{p,\widehat{W}_{p,c}}, \Lambda_{p,\widehat{W}_{p,i}}, \Lambda_{p,\widehat{W}_{p,f}}, \Lambda_{p,\widehat{W}_{p,o}}] \forall p \in \{1, \ldots, m\}$, where $\Lambda_{p,\nu_p} \triangleq \frac{\partial \widehat{\Psi}_p}{\partial \mu_p} \forall \nu_p \in \{\widehat{V}_{p,c}, \widehat{V}_{p,i}, \widehat{V}_{p,f}, \widehat{V}_{p,o}, \widehat{V}_{p,h}, \widehat{W}_{p,c}, \widehat{W}_{p,i}, \widehat{W}_{p,f}, \widehat{W}_{p,o}\}$. Using the chain rule and properties of the vectorization operator, the terms in Λ_{p,ν_p} can be computed as

$$\begin{aligned} \Lambda_{p,\widehat{V}_{p,h}} &= I_n \otimes \widehat{\Psi}_{p,h}^\top \\ \Lambda_{p,\mu_p} &= \widehat{V}_{p,h}^\top \widehat{\Psi}'_{p,h,\mu_p} \end{aligned} \tag{12}$$

$\forall p \in \{1, \ldots, m\}$ and $\forall \mu_p \in \{\widehat{V}_{p,c}, \widehat{V}_{p,i}, \widehat{V}_{p,f}, \widehat{V}_{p,o}, \widehat{W}_{p,c}, \widehat{W}_{p,i}, \widehat{W}_{p,f}, \widehat{W}_{p,o}\}$, where $\widehat{\Psi}_{p,h} \triangleq \Psi_h(\hat{\eta}_p, \hat{c}_p, \hat{h}_p, \hat{\theta}_p) \forall p \in \{1, \ldots, m\}$, and the terms in $\widehat{\Psi}'_{p,h,\mu_p}$ in (12) are defined as $\widehat{\Psi}'_{p,h,\mu_p} \triangleq \frac{\partial \widehat{\Psi}_{p,h}}{\partial \mu_p} \forall \mu_p \in \{\widehat{V}_{p,c}, \widehat{V}_{p,i}, \widehat{V}_{p,f}, \widehat{V}_{p,o}, \widehat{W}_{p,c}, \widehat{W}_{p,i}, \widehat{W}_{p,f}, \widehat{W}_{p,o}\}$. Using the chain rule, the properties of the Hadamard product, and the properties of vectorization, the terms $\widehat{\Psi}'_{p,h,\widehat{V}_{p,o}}$, $\widehat{\Psi}'_{p,h,\widehat{W}_{p,o}}$, and $\widehat{\Psi}'_{p,h,\mu_p}$ can be computed as

$$\begin{aligned} \widehat{\Psi}'_{p,h,\widehat{V}_{p,o}} &= \operatorname{diag}\left(\sigma_c\left(\widehat{\Psi}_{p,c}\right)\right)\left(\sigma'_g\left(\widehat{V}_{p,o}^\top \hat{\eta}_p + \widehat{W}_{p,o}^\top \hat{h}_p\right)\right)\left(I_{l_2} \otimes \hat{\eta}_p^\top\right) \\ \widehat{\Psi}'_{p,h,\widehat{W}_{p,o}} &= \operatorname{diag}\left(\sigma_c\left(\widehat{\Psi}_{p,c}\right)\right)\left(\sigma'_g\left(\widehat{V}_{p,o}^\top \hat{\eta}_p + \widehat{W}_{p,o}^\top \hat{h}_p\right)\right)\left(I_{l_2} \otimes \hat{h}_p^\top\right) \\ \widehat{\Psi}'_{p,h,\mu_p} &= \operatorname{diag}\left(\sigma_g\left(\widehat{V}_{p,o}^\top \hat{\eta}_p + \widehat{W}_{p,o}^\top \hat{h}_p\right)\right)\sigma'_c\left(\widehat{\Psi}_{p,c}\right)\widehat{\Psi}'_{p,c,\mu_{p,h}} \end{aligned} \tag{13}$$

respectively, $\forall \mu_p \in \{\widehat{V}_{p,c}, \widehat{V}_{p,i}, \widehat{V}_{p,f}, \widehat{W}_{p,c}, \widehat{W}_{p,i}, \widehat{W}_{p,f}\}$ and $\forall p \in \{1, \ldots, m\}$, where $\sigma'_j \triangleq \frac{\partial}{\partial \operatorname{vec}(y)} \sigma_j(y) \in \mathbb{R}^{l_2 \times l_2} \forall y \in \mathbb{R}^{l_2}$ $\forall j \in \{c, g\}$, $\widehat{\Psi}'_{p,c,\mu_p} \triangleq \frac{\partial \widehat{\Psi}_{p,c}}{\partial \mu_p} \forall \mu_p \in \{\widehat{V}_{p,c}, \widehat{V}_{p,i}, \widehat{V}_{p,f}, \widehat{W}_{p,c}, \widehat{W}_{p,i}, \widehat{W}_{p,f}\}$, and $\widehat{\Psi}_{p,c} \triangleq \Psi_c(\hat{\eta}_p, \hat{c}_p, \hat{h}_p, \hat{\theta}_p) \forall p \in \{1, \ldots, m\}$. Similarly, the terms $\widehat{\Psi}'_{p,c,\mu_p}$ in (13) can be computed as

$$\begin{aligned} \widehat{\Psi}'_{p,c,\widehat{V}_{p,f}} &= \operatorname{diag}\left(\hat{c}_p\right)\sigma'_g\left(\widehat{V}_{p,f}^\top \hat{\eta}_p + \widehat{W}_{p,f}^\top \hat{h}_p\right)\left(I_{l_2} \otimes \hat{\eta}_p^\top\right) \\ \widehat{\Psi}'_{p,c,\widehat{V}_{p,i}} &= \operatorname{diag}\left(\sigma_c\left(\widehat{V}_{p,c}^\top \hat{\eta}_p + \widehat{W}_{p,c}^\top \hat{h}_p\right)\right)\sigma'_g\left(\widehat{V}_{p,i}^\top \hat{\eta}_p + \widehat{W}_{p,i}^\top \hat{h}_p\right) \\ &\quad \left(I_{l_2} \otimes \hat{\eta}_p^\top\right) \\ \widehat{\Psi}'_{p,c,\widehat{V}_{p,c}} &= \operatorname{diag}\left(\sigma_g\left(\widehat{V}_{p,i}^\top \hat{\eta}_p + \widehat{W}_{p,i}^\top \hat{h}_p\right)\right)\sigma'_c\left(\widehat{V}_{p,c}^\top \hat{\eta}_p + \widehat{W}_{p,c}^\top \hat{h}_p\right) \\ &\quad \left(I_{l_2} \otimes \hat{\eta}_p^\top\right) \\ \widehat{\Psi}'_{p,c,\widehat{W}_{p,f}} &= \operatorname{diag}\left(\hat{c}_p\right)\sigma'_g\left(\widehat{V}_{p,f}^\top \hat{\eta}_p + \widehat{W}_{p,f}^\top \hat{h}_p\right)\left(I_{l_2} \otimes \hat{h}_p^\top\right) \\ \widehat{\Psi}'_{p,c,\widehat{W}_{p,i}} &= \operatorname{diag}\left(\sigma_c\left(\widehat{V}_{p,c}^\top \hat{\eta}_p + \widehat{W}_{p,c}^\top \hat{h}_p\right)\right)\sigma'_g\left(\widehat{V}_{p,i}^\top \hat{\eta}_p + \widehat{W}_{p,i}^\top \hat{h}_p\right) \\ &\quad \left(I_{l_2} \otimes \hat{h}_p^\top\right) \\ \widehat{\Psi}'_{p,c,\widehat{W}_{p,c}} &= \operatorname{diag}\left(\sigma_g\left(\widehat{V}_{p,i}^\top \hat{\eta}_p + \widehat{W}_{p,i}^\top \hat{h}_p\right)\right)\sigma'_c\left(\widehat{V}_{p,c}^\top \hat{\eta}_p + \widehat{W}_{p,c}^\top \hat{h}_p\right) \\ &\quad \left(I_{l_2} \otimes \hat{h}_p^\top\right) \end{aligned} \tag{14}$$

$\forall p \in \{1, \ldots, m\}$.

Furthermore, using the chain rule, the product rule, the properties of the Hadamard product, and the properties of vectorization, the term Ξ_p in (11) can be computed as

$$\begin{aligned} \Xi_p = V_h^\top \operatorname{diag}\left(\sigma_c\left(\widehat{\Psi}_{p,c}\right)\right)\sigma'_g\left(\widehat{V}_{p,o}^\top \hat{\eta}_p + \widehat{W}_{p,o}^\top \hat{h}_p\right)\widehat{V}_{p,o}^\top \\ + V_h^\top \operatorname{diag}\left(\sigma_g\left(\widehat{V}_{p,o}^\top \hat{\eta}_p + \widehat{W}_{p,o}^\top \hat{h}_p\right)\right)\sigma'_c\left(\widehat{\Psi}_{p,c}\right)\widehat{\Psi}'_{p,c,\hat{\eta}_p} \end{aligned} \tag{15}$$

$\forall p \in \{1, \ldots, m\}$, where $\widehat{\Psi}'_{p,c,\hat{\eta}_p} \triangleq \frac{\partial \widehat{\Psi}_c}{\partial \hat{\eta}_p}$. Using the chain rule, the product rule, the properties of the Hadamard product, and the properties of vectorization, the term $\widehat{\Psi}'_{p,c,\hat{\eta}_p}$ in (15) can be computed as

$$\begin{aligned} \widehat{\Psi}'_{p,c,\hat{\eta}_p} &= \operatorname{diag}\left(\hat{c}_p\right)\sigma'_g\left(\widehat{V}_{p,f}^\top \hat{\eta}_p + \widehat{W}_{p,f}^\top \hat{h}_p\right)\widehat{V}_{p,f}^\top \\ &+ \operatorname{diag}\left(\sigma_c\left(\widehat{V}_{p,c}^\top \hat{\eta}_p + \widehat{W}_{p,c}^\top \hat{h}_p\right)\right)\sigma'_g\left(\widehat{V}_{p,i}^\top \hat{\eta}_p + \widehat{W}_{p,i}^\top \hat{h}_p\right)\widehat{V}_{p,i}^\top \\ &+ \operatorname{diag}\left(\sigma_g\left(\widehat{V}_{p,i}^\top \hat{\eta}_p + \widehat{W}_{p,i}^\top \hat{h}_p\right)\right)\sigma'_c\left(\widehat{V}_{p,c}^\top \hat{\eta}_p + \widehat{W}_{p,c}^\top \hat{h}_p\right)\widehat{V}_{p,c}^\top \end{aligned} \tag{16}$$

$\forall p \in \{1, \ldots, m\}$.

To deal with technical difficulties raised by the nonlinear structure of neural networks, such as the SLSTM model in (7), results, such as [14], [31], and [32] implement first-order Taylor series approximation-based methods. For the Lb-SLSTM given by (7), we also employ a first-order Taylor series approximation-based error model given by [28, eq. (22)]

$$\widetilde{\Phi} \triangleq \widehat{\Phi}'\widetilde{\Theta} + \mathcal{O}\left(\left\|\widetilde{\Theta}\right\|\right) \tag{17}$$

where the weight estimation error $\widetilde{\Theta} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{ml_3}$ is defined as $\widetilde{\Theta} \triangleq \Theta - \widehat{\Theta}$, the term $\mathcal{O}(\|\widetilde{\Theta}\|) \in \mathbb{R}^{l_1}$ denotes the higher order terms, and the shorthand notation $\widetilde{\Phi} \in \mathbb{R}^{l_1}$ is defined as $\widetilde{\Phi} \triangleq \Phi(x, \widehat{C}, \widehat{H}, \Theta) - \widehat{\Phi}$.

[3]When $\alpha = 1$ and the SLSTM architecture is designed to have shortcut connections, the term α in (11) is nonzero and therefore, the adaptation law is less susceptible to vanishing gradient [30].

D. Control Design

Based on the following stability analysis and using the developed adaptive SLSTM estimate, the control input $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is designed as

$$u \triangleq -\widehat{\Phi} - k_e e - k_s \text{sgn(e)} + \dot{x}_d \tag{18}$$

where $k_e, k_s \in \mathbb{R}_{>0}$ denote user-selected constants, and sgn(e) denotes a discontinuous sliding mode term, which is designed based on stability analysis to compensate for system uncertainties. Substituting the control input in (18) into (3) and adding and subtracting the term $\Phi(x, \widehat{C}, \widehat{H}, \Theta)$ yields the closed-loop error system

$$\dot{e} = f_e + \widetilde{\Phi} + \varepsilon(x) - k_e e - k_s \text{sgn(e)} \tag{19}$$

where $f_e \triangleq \Phi(x, C, H, \Theta) - \Phi(x, \widehat{C}, \widehat{H}, \Theta)$. Substituting (17) into (19) yields

$$\dot{e} = f_e + \widehat{\Phi}'\widetilde{\Theta} + \mathcal{O}\left(\left\|\widetilde{\Theta}\right\|\right) + \varepsilon(x) - k_e e - k_s \text{sgn(e)}. \tag{20}$$

IV. Stability Analysis

In this section, a Lyapunov-based stability analysis is performed for the developed controller and adaptive SLSTM architecture. Consider the Lyapunov function candidate $\mathcal{V}_L : \mathbb{R}^{l_4} \to \mathbb{R}_{\geq 0}$ defined as

$$\mathcal{V}_L(z, t) \triangleq \frac{1}{2} e^\top e + \frac{1}{2} \widetilde{\Theta}^\top \Gamma^{-1} \widetilde{\Theta} \tag{21}$$

where $z : \mathbb{R}_{\geq 0} \to \mathbb{R}^{l_4}$ is defined as $z \triangleq [e^\top, \quad \widetilde{\Theta}^\top]^\top$ and $l_4 \in \mathbb{R}_{>0}$ is defined as $l_4 \triangleq n + ml_3$. The candidate Lyapunov function in (21) can be bounded as $\beta_1 \|z\|^2 \leq \mathcal{V}_L(z) \leq \beta_2 \|z\|^2$, where $\beta_1 \triangleq \min\{\frac{1}{2}, \frac{1}{2}\lambda_{\min}\{\Gamma^{-1}\}\}$ and $\beta_2 \triangleq \max\{\frac{1}{2}, \frac{1}{2}\lambda_{\max}\{\Gamma^{-1}\}\}$. Since the signum term is discontinuous, a generalized time-derivative is used in the stability analysis. The Filippov set-valued map defined in [33, eq. (2b)] is denoted by $K[\cdot]$. Consider a Lebesgue measurable and locally essentially bounded function $h : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$. Then, the function $z(t) : \mathcal{I} \to \mathbb{R}^n$ is called a Filippov solution of $\dot{y} = h(z)$ on the interval $\mathcal{I} \subseteq \mathbb{R}_{\geq 0}$ if y is absolutely continuous on $\mathcal{I}$ and $\dot{y} \in K[h](z)$ for almost all $t \in \mathcal{I}$. Let the state $z(t)$ denote a Filippov solution to the differential equation $\dot{z} \in K[h^*](z)$, where the state $h^* : \mathbb{R}^{l_4} \to \mathbb{R}^{l_4}$ is defined as $h^*(z) \triangleq [\dot{e}, \quad \dot{\widetilde{\Theta}}]^\top$.

The generalized time-derivative of $\mathcal{V}_L$ via the Filippov trajectories of $\dot{z} = h^*(z)$ can be expressed as $\dot{\tilde{\mathcal{V}}}_L(z) \triangleq \bigcap_{z \in \partial V_L(z)} z^\top [K[h^*]^\top(z), 1]^\top$, where $\partial \mathcal{V}_L(z)$ denotes the Clarke generalized gradient of $\mathcal{V}_L(z)$. Since $\mathcal{V}_L(z)$ is continuously differentiable in z, $\partial V_L(z) = \{\nabla V_L(z)\}$, where ∇ denotes the gradient operator. Taking the generalized time derivative of $\mathcal{V}_L$ yields

$$\dot{\tilde{\mathcal{V}}}_L \subseteq \begin{bmatrix} e^\top, & \widetilde{\Theta}^\top \end{bmatrix} K \begin{bmatrix} \dot{e} \\ \Gamma^{-1}\dot{\widetilde{\Theta}} \end{bmatrix}. \tag{22}$$

Then, the term $\dot{\mathcal{V}}_L$ yields

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\in} e^\top K[\dot{e}] - \widetilde{\Theta}^\top K\left[\Gamma^{-1}\dot{\widehat{\Theta}}\right].$$

Substituting the weight adaptation law in (10) and the closed-loop error system in (20) into (22) yields

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\in} e^\top \left(f_e + \widehat{\Phi}'\widetilde{\Theta} + \mathcal{O}\left(\left\|\widetilde{\Theta}\right\|\right) + \varepsilon(x) - k_e e - k_s K\left[\text{sgn}\right](e)\right) - \widetilde{\Theta}^\top \Gamma^{-1}\left(\Gamma \widehat{\Phi}'^\top e\right).$$

Applying $K[\cdot]$ and canceling out the last term yields

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\leq} e^\top \left(f_e + \widehat{\Phi}'\widetilde{\Theta} + \mathcal{O}\left(\left\|\widetilde{\Theta}\right\|\right) + \varepsilon(x) - k_e e - k_s \text{sgn}(e)\right) - \widetilde{\Theta}^\top \widehat{\Phi}'^\top e.$$

Canceling corresponding terms yields

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\leq} -k_e e^\top e - k_s \|e\|_1 + e^\top \left(\varepsilon(x) + \mathcal{O}\left(\left\|\widetilde{\Theta}\right\|\right) + f_e\right). \tag{23}$$

To facilitate the subsequent analysis, let the open and connected compact sets $\mathcal{D} \subset \mathbb{R}^{l_4}$ and $\mathcal{S} \subset \mathbb{R}^{l_4}$ be defined as $\mathcal{S} \triangleq \{\varsigma \in \mathbb{R}^{l_4} : \|\varsigma\| \leq \sqrt{\frac{\beta_1}{\beta_2}}\omega\}$ and $\mathcal{D} \triangleq \{\varsigma \in \mathbb{R}^{l_4} : \|\varsigma\| \leq \omega\}$, respectively, where $\omega \in \mathbb{R}_{>0}$ denotes a user-selected bounding constant. Lemma 1 in [20] states that the norms of the hidden and cell states, can be bounded by known constants. By the design of LSTM and the use of sigmoid and tanh activation functions, the norms of C, H, $\widehat{C}$, and $\widehat{H}$ can also be bounded by known constants. Using this and Assumption 1, there exist known constants $\overline{\mathcal{O}} \in \mathbb{R}_{>0}$ and $\overline{f_e} \in \mathbb{R}_{>0}$ such that $\mathcal{O}(\|\widetilde{\Theta}\|)$ and f_e can be bounded as $\|\mathcal{O}(\|\widetilde{\Theta}\|)\| \leq \overline{\mathcal{O}}$ and $\|f_e\| \leq \overline{f_e}$, respectively, when $z \in \mathcal{D}$. Substituting these bounds into (23) yields

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\leq} -k_e e^\top e - \|e\|\left(k_s - \overline{\varepsilon} - \overline{\mathcal{O}} - \overline{f_e}\right) \tag{24}$$

when $z \in \mathcal{D}$. Since the universal function approximation property is only on the compact set $\mathcal{X}$, the following analysis has to guarantee $x \in \mathcal{X}\ \forall t \geq 0$. It is shown that $x \in \mathcal{X}\ \forall t \geq 0$ by proving z lies in a compact domain, specifically that $z \in \mathcal{D}\ \forall t \geq 0$ when z is initialized such that $z(0) \in \mathcal{S}$. Then, it can be shown that $x \in \mathcal{X}\ \forall t \geq 0$, and therefore, the universal function approximation property holds.

Theorem 1: The controller in (18) and the weight adaptation law in (10) ensure asymptotic tracking error convergence in the sense that $\|x - x_d\| \to 0$ as $t \to \infty$ provided $z(0) \in \mathcal{S}$ and the following gain condition is satisfied

$$k_s > \overline{\varepsilon} + \overline{\mathcal{O}} + \overline{f_e}. \tag{25}$$

Proof: Consider the candidate Lyapunov function in (21). Provided the sufficient gain condition in (25) is met, (24) can be further bounded as

$$\dot{\mathcal{V}}_L \overset{a.a.t.}{\leq} -k_e e^\top e \tag{26}$$

when $z \in \mathcal{D}$. To show $z \in \mathcal{D}\ \forall t \geq 0$, using (21) and the fact that $\dot{\mathcal{V}}_L(z(t)) \overset{a.a.t.}{\leq} 0$ when $z \in \mathcal{D}$ implies $z(t)$ can be bounded as $\|z(t)\| \leq \sqrt{\frac{\beta_2}{\beta_1}}\|z(0)\|$. Thus, if $\|z(0)\| \leq \omega\sqrt{\frac{\beta_1}{\beta_2}}$, then $\|z(t)\| \leq \omega\ \forall t \geq 0$, and $e, \widetilde{\Theta} \in \mathcal{L}_\infty$. Therefore, if z is initialized such that $z(0) \in \mathcal{S}$, then $z \in \mathcal{D}\ \forall t \geq 0$. To show $x \in \mathcal{X}$, let the open and connected set $\Upsilon \subseteq \mathcal{X}$ be defined as $\Upsilon \triangleq \{\varsigma \in \mathcal{X} : \|\varsigma\| \leq \omega + \overline{x}_d\}$. Using the fact that $\|z(t)\| \leq \omega\ \forall t \geq 0$, it can be shown that $\|e(t)\| \leq \omega\ \forall t \geq 0$. Hence, using (2), x can be bounded as $\|x\| \leq \omega + \overline{x}_d$. Therefore, if $z(0) \in \mathcal{S}$, then $x \in \Upsilon \subseteq \mathcal{X}$. Using (21) and $\dot{\mathcal{V}}_L \overset{a.a.t.}{\leq} 0$, $e, \widetilde{\Theta} \in \mathcal{L}_\infty$. Θ is bounded due to Assumption 1 and x_d is bounded by design. Using $e, \widetilde{\Theta}, \Theta, x_d \in \mathcal{L}_\infty$ implies $x, \widehat{\Theta} \in \mathcal{L}_\infty$. Using [20, Lemma 1], $\widehat{C}, \widehat{H} \in \mathcal{L}_\infty$. Using (10), the fact that $\widehat{C}, \widehat{H}, \widehat{\Theta}, x \in \mathcal{L}_\infty$ and the fact that $\widehat{\Phi}$ is smooth implies $\dot{\widehat{\Theta}} \in \mathcal{L}_\infty$. Using (18) and the fact that $\widehat{C}, \widehat{H}, \widehat{\Theta}, x, e \in \mathcal{L}_\infty$ implies $u \in \mathcal{L}_\infty$. Then, using the extension of LaSalle–Yoshizawa theorem for nonsmooth systems [34], it can be shown that $\lim_{t\to\infty}\|e(t)\| = 0$, resulting in asymptotic tracking error convergence. ■

TABLE I
MEAN AND STANDARD DEVIATION OF PERFORMANCE COMPARISON RESULTS FOR 50 RUNS

NN Architecture	$\|e\|$	$\|f(x)-\widehat{\Phi}\|$	$\|u\|$
SLSTM	0.1238±0.0153	0.8252±0.1244	3.7974±0.1175
LSTM	0.1592±0.0153	1.3791±0.1591	3.9769±0.1392
DNN	0.4128±0.0173	4.4170±0.1978	4.7179±0.1863

V. SIMULATIONS

The Lb-SLSTM adaptive controller is implemented in simulations on an eight-state nonlinear system. The two comparative baselines implement the DNN in [14] and the LSTM in [20] under the same architecture of controllers, adaptive laws, and dynamic systems. The Lyapunov-based adaptation eliminates the need for a training set.

For the simulations, the unknown drift vector field in (1) is modeled as $f(x) = Ad(x)$, where $A \in \mathbb{R}^{8\times 64}$ is a random matrix with all elements distributed uniformly as $U(0, 0.05)$, and $d(x) \triangleq [x^\top, \cos(x)^\top, \sin(x)^\top, \tanh(x)^\top, (x \odot x)^\top, (x \odot x \odot x)^\top, (x \odot x \odot x + x \odot x + x)^\top, (\sin(x) + \tanh(x) + x)^\top]^\top$. The simulations are performed for 100 s with a 0.001 s step size. The initial state $x(0)$ is selected from a uniform distribution of $U(-3, 3)$. The initial weights are generated from a uniform distribution of $U(-0.3, 0.3)$ for each iteration. The desired trajectory is designed as $x_d(t) = [\sin(\frac{\pi t}{2}), \ldots, \sin(\frac{\pi t}{2})]^\top$. The SLSTM in (7) is designed with five LSTM cells, with a shortcut connection across each LSTM cell (i.e., $\alpha = 1$) and $l_2 = 5$ neurons. The gains are selected as $b_c = 1$, $b_h = 1$, $k_e = 11$, $k_s = 0.01$, and $\Gamma = 0.3I_{1500}$.

The LSTM and DNN controllers were constructed by replacing the Lb-SLSTM term in (18) with the adaptive Lb-LSTM model in [20] and the adaptive Lb-DNN model in [14], respectively. For a fair comparison, the same learning gain and robust gains were selected for all three controllers. The Lb-DNN architecture was designed with four hidden layers with five neurons each and tanh activation functions. Compared to the Lb-SLSTM architecture in (7), the Lb-LSTM model in [20] implements only one LSTM cell with the same number of neurons (i.e., $l_2 = 5$ neurons).

Since the Lb-SLSTM model and baseline models are sensitive to weight initialization, a Monte Carlo approach was used to initialize the weight estimates. In this method, 1000 simulations are performed, where the initial weights in each simulation are selected from $U(-0.3, 0.3)$, and the cost $J = \int_0^{100} e^\top(t)e(t)dt$ is evaluated in each simulation. The weights with the lowest cost are selected.

Table I provides the mean and standard deviation of the norm of the root mean square (rms) tracking error, function approximation error, and control input for 50 iterations. All three adaptive NN architectures compensate for the system uncertainty and accomplish trajectory tracking. Based on a t-test, the Lb-SLSTM is statistically significant ($p < 0.001$) from Lb-LSTM and Lb-DNN for tracking error, function approximation error, and control input. Based on a representative run as shown in Figs. 2 and 3, all three models converge to their final error bounds within approximately 0.5 s. However, the Lb-SLSTM demonstrates better tracking performance with improved function approximation performance and less control effort. In the steady state period, the Lb-DNN and Lb-LSTM models yield periodic peaks in tracking and function approximation error due to the repetitive design of the desired trajectory. This behavior is eliminated with the Lb-SLSTM model. During the steady state period, the error bound for the Lb-LSTM gradually decreases over time (from approximately 0.25 to 0.08), showing continual learning of the Lb-LSTM model throughout the entire simulation run time. Conversely, the Lb-DNN controller

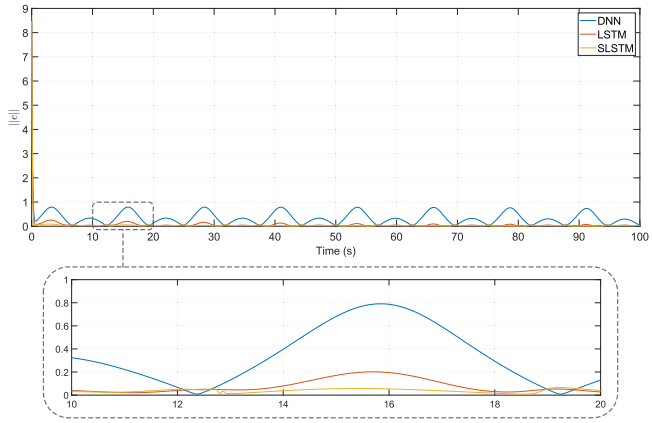

Fig. 2. Plots of the norms of rms tracking error $\|e\|$ over time for the Lb-SLSTM, Lb-LSTM, and Lb-DNN adaptive controllers for one representative run. The zoomed-in view shows characteristic plot features from 10 to 20 s.

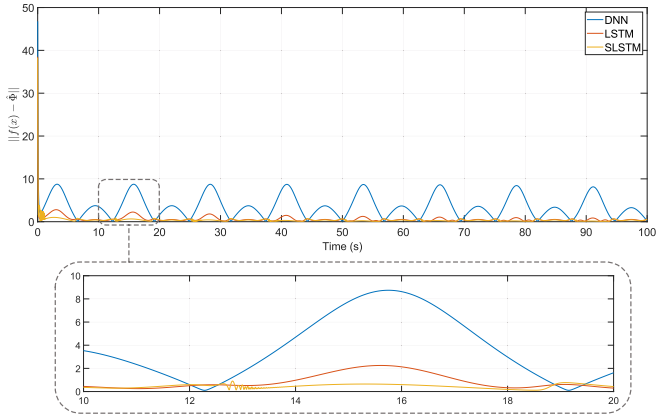

Fig. 3. Plots of the norms of rms function approximation error $\|f(x) - \widehat{\Phi}\|$ over time for the Lb-SLSTM, Lb-LSTM, and Lb-DNN adaptive controllers for one representative run. The zoomed-in view shows characteristic plot features from 10 to 20 s.

maintains roughly the same tracking error bound through the simulation run time (approximately 0.8). The maximum steady state errors for the Lb-DNN, Lb-LSTM, and Lb-SLSTM architectures were approximately 0.8, 0.25, and 0.01, respectively. Moreover, the Lb-SLSTM model yielded a 96.00% and 98.75% improvement in maximum steady state error performance, when compared to the Lb-LSTM and Lb-DNN models, respectively.

Overall, the SLSTM-based controller and developed weight adaptation law resulted in improved tracking error performance. When compared to the Lb-LSTM and Lb-DNN controllers in Table I, the Lb-SLSTM controller yielded an average improvement of 22.24% and 70.01% in tracking error performance, as well as 40.16% and 81.32% in function approximation error performance, respectively, with comparable control effort.

VI. CONCLUSION

This article introduces an adaptive control framework that leverages a deep LSTM NN architecture to address real-time learning challenges in dynamic systems with unknown drift dynamics. The developed Lb-SLSTM architecture incorporates stacked LSTM cells and shortcut

connections and is implemented in the controller as a feedforward estimate to adaptively learn model uncertainties. Stability-driven adaptation laws are developed for the weights of the Lb-SLSTM architecture using Lyapunov stability-driven techniques. A Lyapunov-based stability analysis ensures asymptotic tracking error convergence for the developed adaptive Lb-SLSTM control framework. In comparative simulations on an eight-state nonlinear system, the Lb-SLSTM model yielded an improvement of 96.00% and 98.75% in maximum steady state error performance, and an average improvement of 22.24% and 70.01% in tracking error performance, as well as 40.16% and 81.32% in function approximation error performance when compared to the Lb-LSTM and Lb-DNN models, respectively.

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsoring agencies.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] D. Rolnick and M. Tegmark, "The power of deeper networks for expressing natural functions," 2017, *arXiv:1705.05502*.
[3] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. Int. Conf. Mach. Learn*, 2016, pp. 1842–1850.
[4] D. Muthirayan and P. P. Khargonekar, "Memory augmented neural network adaptive controllers: Performance and stability," *IEEE Trans. Autom. Control*, vol. 68, no. 2, pp. 825–838, Feb. 2023.
[5] L. Heindel, P. Hantschke, and M. Kästner, "A data-driven approach for approximating non-linear dynamic systems using LSTM networks," *Procedia Struct. Integr.*, vol. 38, pp. 159–167, 2022.
[6] J. Li, Y. Huang, Q. Li, and Y. Li, "Closed-LSTM neural network based reference modification for trajectory tracking of piezoelectric actuator," *Neurocomputing*, vol. 467, pp. 379–391, 2022.
[7] S.-L. Dai, C. Wang, and M. Wang, "Dynamic learning from adaptive neural network control of a class of nonaffine nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 111–123, Jan. 2014.
[8] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans, "Nonlinear systems identification using deep dynamic neural networks," 2016, *arXiv:1610.01439*.
[9] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 543–548, Jul. 2018.
[10] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3866–3878, Sep. 2020.
[11] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3050–3057, Apr. 2020.
[12] R. Sun, M. Greene, D. Le, Z. Bell, G. Chowdhary, and W. E. Dixon, "Lyapunov-based real-time and iterative adjustment of deep neural networks," *IEEE Control Syst. Lett.*, vol. 6, pp. 193–198, 2022.
[13] D. Le, M. Greene, W. Makumi, and W. E. Dixon, "Real-time modular deep neural network-based adaptive control of nonlinear systems," *IEEE Control Syst. Lett.*, vol. 6, pp. 476–481, 2022.
[14] O. Patil, D. Le, M. Greene, and W. E. Dixon, "Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network," *IEEE Control Syst Lett.*, vol. 6, pp. 1855–1860, 2022.
[15] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," 2013, *arXiv:1312.6026*.
[16] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems," *Sci. Rep.*, vol. 9, no. 1, 2019, Art. no. 19038.
[17] T. A. Farrag and E. E. Elattar, "Optimized deep stacked long short-term memory network for long-term load forecasting," *IEEE Access*, vol. 9, pp. 68511–68522, 2021.
[18] R. Azzam, Y. Alkendi, T. Taha, S. Huang, and Y. Zweiri, "A stacked LSTM-based approach for reducing semantic pose estimation error," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 2502614.
[19] D. M. Le, O. S. Patil, C. F. Nino, and W. E. Dixon, "Accelerated gradient approach for deep neural network-based adaptive control of unknown nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 4, pp. 6299–6313, 2025, doi: 10.1109/TNNLS.2024.3395064.
[20] E. Griffis, O. Patil, Z. Bell, and W. E. Dixon, "Lyapunov-based long short-term memory (Lb-LSTM) neural network-based control," *IEEE Control Syst. Lett.*, vol. 7, pp. 2976–2981, 2023.
[21] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process*, 2013, pp. 6645–6649.
[22] J. Wang, B. Peng, and X. Zhang, "Using a stacked residual LSTM model for sentiment intensity prediction," *Neurocomputing*, vol. 322, pp. 93–101, 2018.
[23] D. S. Bernstein, *Matrix Mathematics*. Princeton, NJ, USA: Princeton Univ. Press, 2009.
[24] P. Bentler and S.-Y. Lee, "Matrix derivatives with chain rule and rules for simple, Hadamard, and Kronecker products," *J. Math. Psychol.*, vol. 17, no. 3, pp. 255–262, 1978.
[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–80, 1997.
[26] E. Kosmatopoulos, M. Polycarpou, M. Christodoulou, and P. Ioannou, "High-order neural network structures for identification of dynamical systems," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 422–431, Mar. 1995.
[27] P. Kidger and T. Lyons, "Universal approximation with deep narrow networks," in *Proc. Conf. Learn. Theory*, 2020, pp. 2306–2327.
[28] F. L. Lewis, A. Yegildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 388–399, Mar. 1996.
[29] V. Stepanyan and A. Kurdila, "Asymptotic tracking of uncertain systems with continuous control using adaptive bounding," *IEEE Trans. Neur. Netw.*, vol. 20, no. 8, pp. 1320–1329, Aug. 2009.
[30] O. S. Patil, D. M. Le, E. Griffis, and W. E. Dixon, "Deep residual neural network (ResNet)-based adaptive control: A Lyapunov-based approach," in *Proc. IEEE Conf. Decis. Control*, 2022, pp. 3487–3492.
[31] F. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural net robot controller: Structure and stability proofs," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 388–399, 1996.
[32] S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, *Stable Adaptive Neural Network Control*. Boston, MA, USA: Kluwer Academic Publishers, 2002.
[33] B. E. Paden and S. S. Sastry, "A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators," *IEEE Trans. Circuits Syst.*, vol. 34, no. 1, pp. 73–82, Jan. 1987.
[34] N. Fischer, R. Kamalapurkar, and W. E. Dixon, "LaSalle-Yoshizawa corollaries for nonsmooth systems," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2333–2338, Sep. 2013.